An OPENID Identity Service for Android, Based on USIM Secure Elements

Pascal Urien

Telecom ParisTech.,
23 Avenue d'Italie, 75013, Paris, France
Pascal.Urien@Telecom-ParisTech.fr

Abstract. This paper presents a new identity platform based on the OPENID standard for Android mobile and working with the Open Mobile API, which has been recently defined by the SIM Alliance committee. This innovative service comprises four components, an embedded USIM TLS stack, an Android proxy application (PS), an identity provider (IdP) server, and an OPENID authentication server. USIM modules are issued with a device certificate and may thereafter download other certificates from the IdP server. All HTTP exchanges are secure by the USIM TLS stack. This platform may work with about one million of WEB sites, which are today compatible with the OPENID framework.

Keywords: Security, OPENID, USIM, Android, Secure Element.

1 Introduction

The mobile applications rocket is fuelled by the development of the Android devices sales. According to [1], this open operating system could power about 50% of smartphones by 2015. Android [8] was originally created by the company Android Inc, bought in 2005 by Google. The first mobile (the HTC G1) was commercialized in 2008. Some Android mobile are open, such as the Nexus S, i.e. quite all code sources (except some libraries imported under a binary format) are available in dedicated repositories managed by the GIT and REPO tools [7]. It is therefore possible to compile and load OEM versions for some models. Furthermore, Android works with a UNIX kernel; there are many tools over the WEB enabling to root the system, which consequently may be modified without manufacturers or operators agreement. We recently made a reverse engineering of mobile banking applications, which shows that most of them do not rely on Android libraries, but embed their own cryptographic resources, typically used for TLS sessions with online account servers. A way to solve trust issues in mobile is to run some parts of applications in tamper resistant components, usually referred as Secure Elements (SE). A SE is a secure microcontroller protected by various physical and logical countermeasures. There are four classes of such devices in today mobiles: USIM [11] chips mandatory for 3G/4G/LTE authentication, NFC (Near Field Communication) controllers [14], micro SD cards [9][10] including secure microcontrollers, and extern contactless smartcard feed by NFC reader. In this paper we focus on USIM device because we target an operator service.

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 355–366, 2013.

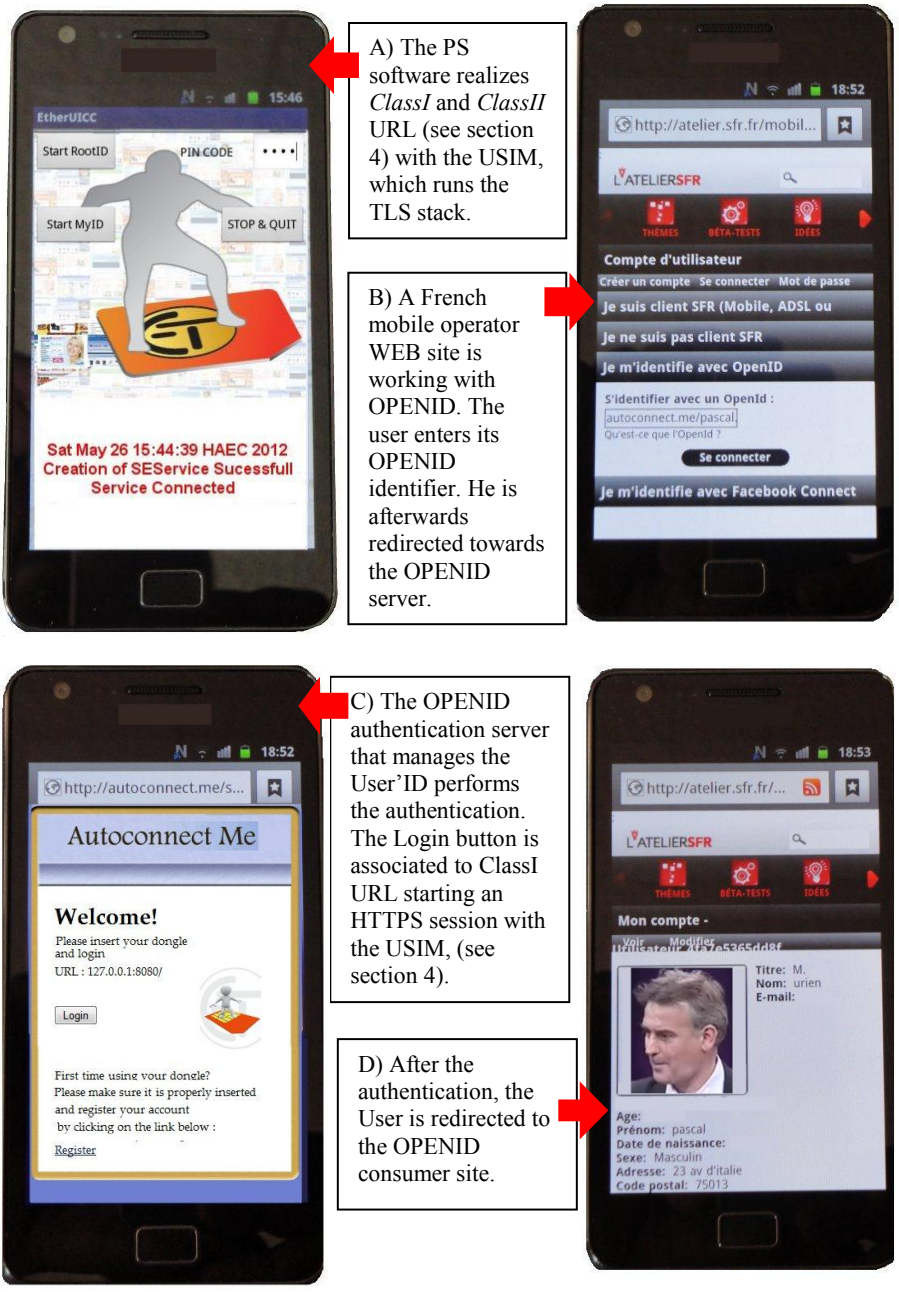

Fig. 1. Illustration of the User's experience OPENID Identity Service

A TLS [18] stack is running in the USIM it performs the booting of TLS sessions, i.e. until the reception of the server finished message for the TLS full mode, and the transmission of the client finished message for the abbreviated (or resume) mode. We use TLS with certificates for both client and server sides, i.e. there is a strong mutual authentication based on PKI between these two entities. The TLS session is afterwards transferred to the android mobile application which collects the CipherSuite and the associated KeysBlock values needed by the record layer for encryption and integrity procedures.

In android, telephony operations (ingoing calls, outgoing calls…) are controlled by the RIL (Radio Layer Interface) software component [6], acting as a TCP daemon. It is used for packets exchange between the application processor (in which is running the operating system) and the baseband processor that handles radio interfaces and controls the USIM. For security reasons the RIL forbids information exchange between the application processor and the USIM. Nerveless the SIM Alliance has defined an Open Mobile API [4] that unifies SE access in mobile environments, and which allows interactions with applications embedded in the USIM. For that purpose the baseband manufacturer delivers a special RIL version that enables communication with the USIM, typically via AT-CSIM [25] commands. Furthermore this API manages an "Access Control Rules Database & Engine", which checks that the application running in the SE (the USIM in our case) authorizes interactions with mobile applications, whose rights and integrity are checked by a certificate. Elements of this security policy are stored according to a PKCS#15 scheme in the USIM file system [5]. In this context we have developed an identity platform based on the OPENID standard [19], which according to [2] is supported by about one million of WEB sites. The basic idea is that USIM are issued with a TLS stack and a device certificate. A mobile proxy application establishes the glue between the browser and two classes of remote servers. An Identity Provider (IdP) server delivers new identities (i.e. certificate and private key) to the USIM. An OPENID authentication server that trusts these identities (i.e. the Certification Authority -CA- issuing the certificates) authenticates the USIM, via TLS sessions with mutual authentication.

The user experience is illustrated by figure 1. The mobile bearer starts the proxy application that establishes the glue between the browser and the USIM. He then works with a WEB site that requires an OPENID identifier for login operation. He is thereafter identified by an HTTPS session with the OPENID authentication server.

Finally he is redirected to the previous site that displays the success of the authentication operation.

This paper is constructed according to the following outline. Section 2 introduces the android architecture, more specially the Nexus S device. Section 3 presents the "Open Mobile API" and its policy access rules for the USIM. Section 4 describes our new identity service that comprises four components, an embedded USIM application, a mobile android application, an identity provider server, and an OPENID authentication server. Finally section 5 concludes this paper.

This work has been done with the collaboration of the EtherTrust Company. It has been partially funded by the STREP Project 288349, SecFuNet.

2 About Android

The Android system [8] is mainly powered by a Linux kernel and a Dalvik virtual machine. The figure 2 illustrates the hardware and software architecture for a Nexus S mobile. There are two processors, an application processor that runs the operating system, a baseband processor that manages all the radio packet exchange. Two secure elements are available, the USIM and the NFC controller.

The Linux kernel manages memories and processes. It also handles the IO drivers (/dev/devices) needed for the access to numerous peripheral devices that are controlled by the application processor. Two low level drivers handle components giving access to Secure Elements: the baseband driver; and the NFC controller driver embedding an optional tamper resistant microcontroller.

The kernel is built with a tool named GIT and is produced under the zImage format. The Android system is usually referred as the ROM, which is made of three parts. 1) The boot image (boot.img), which stores the kernel image. 2) The system image (system.img), which store libraries and applications. 3) The recovery image (revovery.img), a minimum version of the system used to download new ROMs.

The ROM is produced from several logical elements: 1) the kernel imported in binary format; 2) the Dalvik virtual machine code; 3) the libraries code, some of them such as the RIL or NFC components are proprietary and imported in binary format and required a license agreement; 4) a JAVA framework based on JINI interfaces with native libraries; 5) applications offering facilities such as telephony or NFC.

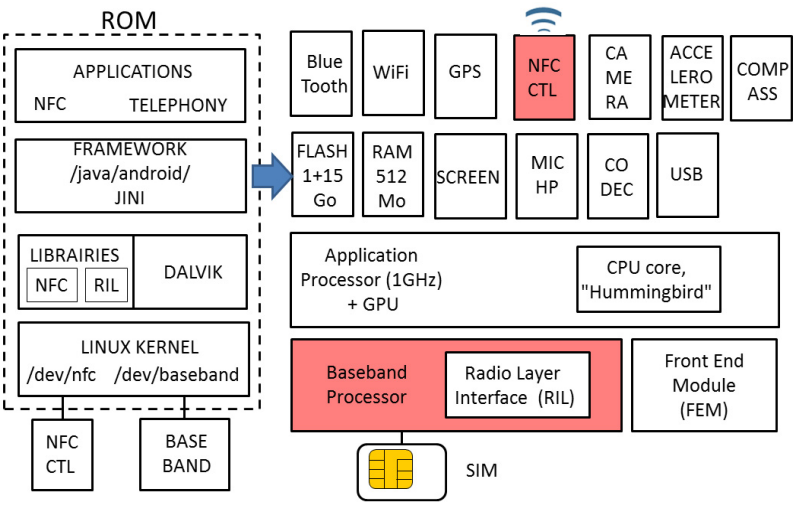

Fig. 2. Android operating system, hardware and software components

A mobile is released with an initial ROM code, which is modified by signed update files (update.zip). Nerveless it is possible to root a mobile phone, i.e. to get root privileges. This allows modifying parts of the ROM stored in the FLASH memory. As illustrated in the previous section some Android platforms are open, i.e. it is possible to build (OEM) ROM from source codes, and to flash it.

For these reasons most of android applications that need trust, work with they own cryptographic resources.

Secure Elements (USIM, NFC Controller, Secure SD...) are processors whose code can't be loaded/modified without specific authorizations. The USIM chip is a tamper resistant device mandatory for 3G/4G and LTE networks. The communication with the application processor is managed by the RIL (Radio Layer Interface) entity, which unfortunately in the current Android model does not provide access to the USIM.

The Open Mobile API detailed in the next section builds a generic framework for the management of secure elements, and works with RILs providing access to the USIM, typically via AT-CSIM commands [25].

3 About the Open Mobile API

3.1 About Secure Elements

Secure elements are tamper resistant devices based on secure microcontrollers. Security is enforced by multiple hardware and software countermeasures. They comprise CPU (8, 16 or 32 bits) with modest computing power, nonvolatile memory (FLASH or E^2PROM, about 100KB), ROM (a few hundred KB) and RAM (about 10KB). Most of them are equipped with a java virtual machine (JVM) and run applications written in javacard [13], a subset of the java language. Applications are identified by an AID (Application Identifier) whose length is 16 bytes at the most.

SE exchange commands, whose maximum size is about 256 bytes, which are called APDUs and defined by the ISO7816 standards [12]. A request APDU comprises a five bytes header, CLA the class of operation (for example 00 for ISO request, A0 for GSM), INS the instruction (B0 for reading, D0 for writing…), two bytes P1 P2, and a P3 parameter either the length of information to be written or the expected length of information to be read.

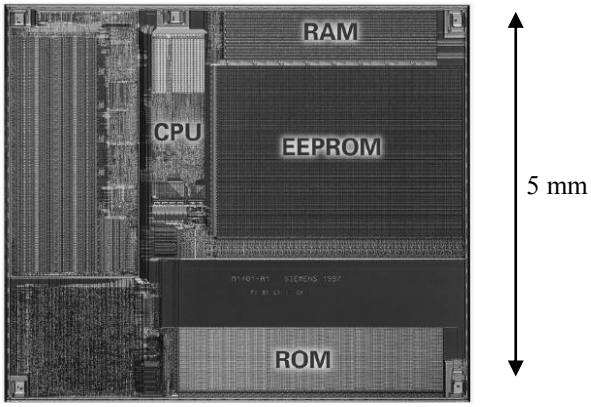

Fig. 3. Illustration of an USIM chip from Siemens ([11])

The Card Manager [15] is an application identified by its AID, which can download and activate software in Secure Elements. These procedures require a mutual authentication based on symmetric cryptographic key. In other words a SE is managed by the entity that controls the Card Manager.

3.2 The Open Mobile API Model

This API [4] defines a generic framework for the access to Secure Elements in a mobile environment. It is based on four main objects.

- The ***SEService*** is the abstract representation of all SEs that are available for applications running in the mobile phone. A service is usually associated to a callback function, invoked by the operating system when the software environment has been successfully created.
- The ***Reader*** is the logical interface with a Secure Element. It is an abstraction from electronics devices which are needed for contact (ISO 7816) and contactless (ISO 14443) smartcards. A *SEService* object gives of list of available Readers.
- The ***Session*** is opened and closed with a Reader. It establishes the logical path with the SE managed by the Reader.
- The ***Channel*** is associated with an application identified by an AID and running in the SE.

There are two kinds of channels: basic and logical. According to the ISO7816 standards SE operating system can manage up to four concurrent applications

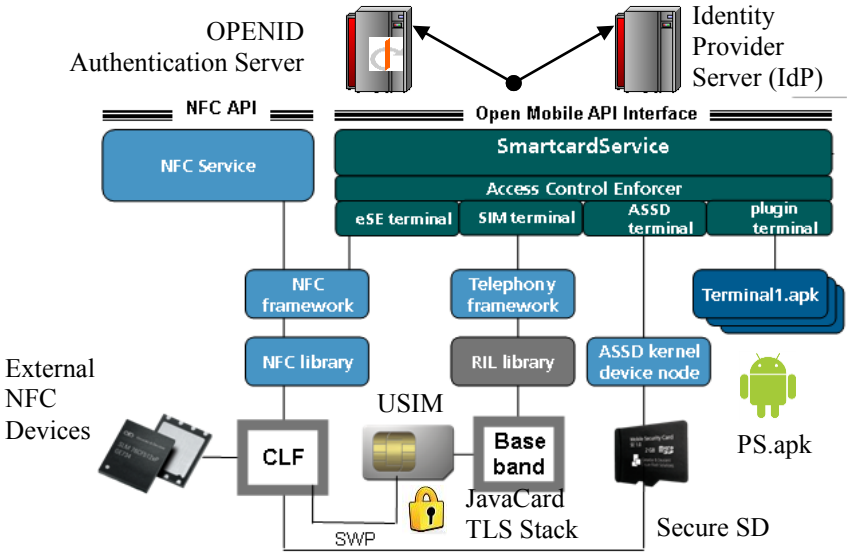

Fig. 4. An illustration of the Open Mobile API environment according to [5], with our identity platform components: USIM TLS stack, Proxy Software -PS-, OPENID and IdP Servers

associated to an index ranging from 0 to 3. This index is included in the two LSB bits of first byte (CLA) of every (APDU) request command. The basic channel is associated to the zero index, other values are refereed as logical channels. In an USIM device the basic channel is allocated to the application that manages authentication (i.e. the AKA algorithm) with the radio network. In other words the Open Mobile API model is application oriented. It manages information exchange with embedded applications (running in the SEs) identified by their AID. This paradigm was previously introduced by the JSR177 API [16] available in Symbian operating systems. Figure 4 illustrates the Open Mobile API services in a mobile environment, which are represented by the green blocks. In this figure CLF means "Contactless Front-end", and is the name used by in ETSI standards [17] for NFC controllers. Three classes of SE are managed. The USIM is accessed thanks to a modified RIL version. A second SE is embedded in a secure SD memory (ASSD standing for Advanced Security SD cards). External contactless smartcards are managed by the NFC (CLF) controller. The idea of trusted application running in mobile is to execute some sensitive procedure in a secure element The Access Control entity (see next section) manages/enforces authorizations for mobile (Android) applications that intend to exchange information with SE.

3.3 Access Control for USIM

In order to protect the USIM from a non-authorized Android application, an access control (AC) mechanism based on the PKCS#15 standard, has been defined in the annex 6 of [3]. According to the ISO 7816 standards repositories (named DF, Dedicated File) and files (named EF, Elementary File) stored in the USIM are identified by two bytes number. The root file (named MF, master file) identifier is 3F00.

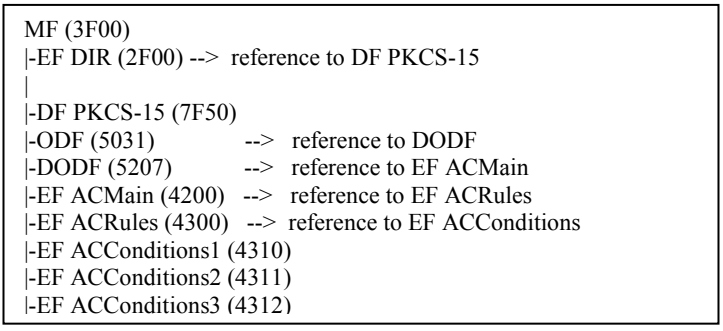

Fig. 5. PKCS#15 files structure for access control

The PKCS#15 repertory (DF-PKCS-15) identifier is listed in the EF-DIR file. Three files defined the access rules (see figure 5):

- The Access Control Main File (EF-ACMF) gives a reference to the Access Control Rules File (EF-ACRF)

- The Access Control Rules File (EF-ACRF) stores a list of Access Control Conditions File (EF-ACCF), each of them being associated to a particular AID.
- Each Access Control Conditions File (EF-ACCF), contains a SHA1 digest of the mobile application whose access to embedded software running in the Secure Element (and identified by its AID) is authorized.

Thanks to this structure the mobile operator defines access rules for mobile application that intend to use services provided by USIM applications.

4 The OPENID Identity Platform

4.1 About OPENID

The OPENID concept was initially developed by Brad Fitzpatrick [21] in 2005, and is based on a specific identity server that *"… is responsible for telling the rest of the world if you're you or not, and digitally signing a receipt saying that you said so, but only if you've told your identity server if you want to*". According to [2] about one million of WEB sites are compatible with this standard in 2012.

OPENID [19] is a Single Sign On (SSO) system made of three entities, the consumer site requiring a user authentication, the authentication server (the OPENID server) performing the user authentication, and an internet user equipped with an IP terminal.

We previously introduced OPENID support for smartcards in [20][24].

The OPENID identifier is an URL that comprises two parts the name of the OPENID server and the user alias. A security association is statically or dynamically established between the consumer and the authentication server, i.e. a secret key is shared between these two entities. It is used for message authentication, performed by an HMAC procedure.

The user gives its identifier on the consumer site, which realizes if necessary a security association with the authentication server. Thereafter a redirection transfers via the user terminal, an OPENID authentication request toward the OPENID server. This last makes the proof of user identity, and returns to the consumer site a list of attributes authenticated by the shared secret.

There are many ways to perform authentication between the user and the OPENID server. The most popular is the simple password mechanism.

In our identity service we use a strong mutual authentication based on a TLS session running in the USIM, in which both client and server are identified by their X509 certificates and associated private keys.

4.2 Components of the OPENID Platform

The Identity platform comprises four elements:

- The USIM equipped with a javacard applet running a TLS stack, and supporting secure facilities for identities downloading.

- A mobile (Android) proxy software (PS), which realizes the logical glue between the browser, the USIM and internet servers. It provides two services, TLS session booting and identity management in the USIM.

- An OPENID server. The mobile is authenticated thanks to TLS sessions running in the USIM. An optional PIN code may enforce a dual form factor procedure.
- An Identity Provider server (IdP). USIM are produced with a device identity. The IdP, whose access are controlled via TLS sessions, generates identities that are afterwards downloaded in the USIM.

4.2.1 The USIM

The USIM stores an application (identified by its AID) and the files needed by the Open Mobile API for the PS authorization.

The TLS services are made available thanks to the EAP-TLS protocol (RFC 2716), introduced in 2000 by the Microsoft Company. These latter transports TLS packets in a datagram like ways, and does not required TCP/IP flavors. The binary encoding rules over ISO7816 APDU are detailed by an IETF draft [26]. This application performs "TLS booting", i.e. it fully handles TLS until the reception of the server finished message for the full mode, and the transmission of the client finished message for the abbreviated (or resume) mode. Afterwards the CipherSuite and the associated KeysBlock values are transferred to the proxy (PS), which handles record layer operations (i.e. encryption/decryption and integrity procedures).

An identity is a set of the following attributes: an alias name; a certification authority (CA) X509 certificate; an X509 certificate; and a RSA private key. The USIM ID is the subject of its current certificate. Each USIM is manufactured with a "root" identity (the device identity or RootID) that cannot be deleted. Nerveless it may store other identities generated by the IdP server, refereed as UserID, and identified by an index greater than zero. They are written in the USIM thanks to ISO7816 commands, and protected by a secure structure named Container. A Container is made of three parts:

- A header, which is the encrypted value of a symmetric AES key, with the RootID public key, according to the PKCS#1 standard.
- A body, which is the encrypted value of all identity attributes, according to an AES-CBC procedure.
- A trailer, which is the signature by a trusted authority (identified by its public key) of the header concatenated to the body, according to the PKCS#1 standard.

Upon downloading, the signature is checked, the AES key is recovered with the USIM RootID private key, and the identity is decrypted and stored. A set of ISO7816 commands realizes the basic operations for identity management, i.e. identity inventory, activation and deletion.

4.2.2 The Android Proxy Software

This Android application realizes the logical glue between the mobile browser, the USIM and internet servers. Its User Interface is illustrated by figure 1. It is possible to enter an optional PIN code and then to start the USIM application with the RootID or the current identity activated in the chip. The PS is a TCP daemon, opened on the loopback address (127.0.0.1), usually with the port 8080. Its interface is made by two kinds of URLs, used for network exchange and USIM commands. The first class of URLs (***ClassI***), such as

http://127.0.0.1:8080/~url=www.server.com/path/file.html

opens an HTTPS session the remote server (i.e. is equivalent to https://www.server.com/path/file.html). The TLS session is booted from the USIM and then transferred back to the PS. As a result the browser opens HTTP session with remote servers that are authenticated via the USIM. The second class of URLs (***ClassII***), such as

http://127.0.0.1:8080/reader/apdu.xml

sends ISO7816 commands to the USIM application. These commands are located in the body of an HTTP POST request. There are encoded as the content of form inputs:

=CMD1&=CMD2&=CMDi

The returned response is encoded according to the XML format.

4.2.3 The Identity Provider (IdP) Server

The IdP server manages the USIM identities. It is in charge of the following operations

- *Subscriber registration/recovering*. Each USIM owns a RootID. The registration process establishes a logical link between the subscriber identity and its RootID. Should the UICC be lost, the subscriber will recover its previously generated IDs, by proving its knowledge of some registration parameters.
- *Identity generation/deletion*. A registered subscriber is always identified / authenticated by its RootID. It may afterwards generate identities for personal use, which are stored in the IdP database.
- *Identity loading/removing*. Identities previously generated may be downloaded in the USIM or removed from the USIM.
- *Identity activation*. Once an identity has been downloaded in the USIM it may be activated, and is afterwards referred as the CurrentID. All further operations dealing with OPENID services work with this identity.

The interactions between the IdP server and the USIM are based on the AJAX [22] technology. AJAX (shorthand for Asynchronous JavaScript and XML) is a set of technologies for executing applications on the browser side, triggered by user's interactions, typically mouse clicks associated with HTML forms, and processed by JavaScript procedures.

AJAX pages are downloaded by the mobile, via ClassI URLs, authenticated by the USIM. These pages include XMLHttpRequest [23] objects that send ClassII HTTP requests to the proxy server. These requests transport ISO7816 commands, executed afterwards by the USIM. The PS returns ISO 7816 responses, embedded in XML pages. These XML contents are parsed and thanks to the DOM (Document Object Model for JavaScript) the initial page is modified, typically with a success status. Thanks to these mechanisms the IdP server downloads container in the USIM, and performs identity management operations (inventory, activation, deletion). All these procedures are available from the mobile WEB browser. Figure 6 illustrates AJAX interaction with the USIM. An ISO7816 command is hidden in an HTML form. When the user clicks on the SetRootID button this form is sent to the proxy software and processed by the USIM. The response encoded in the XML format is afterwards parsed and a modified page is displayed by the browser.

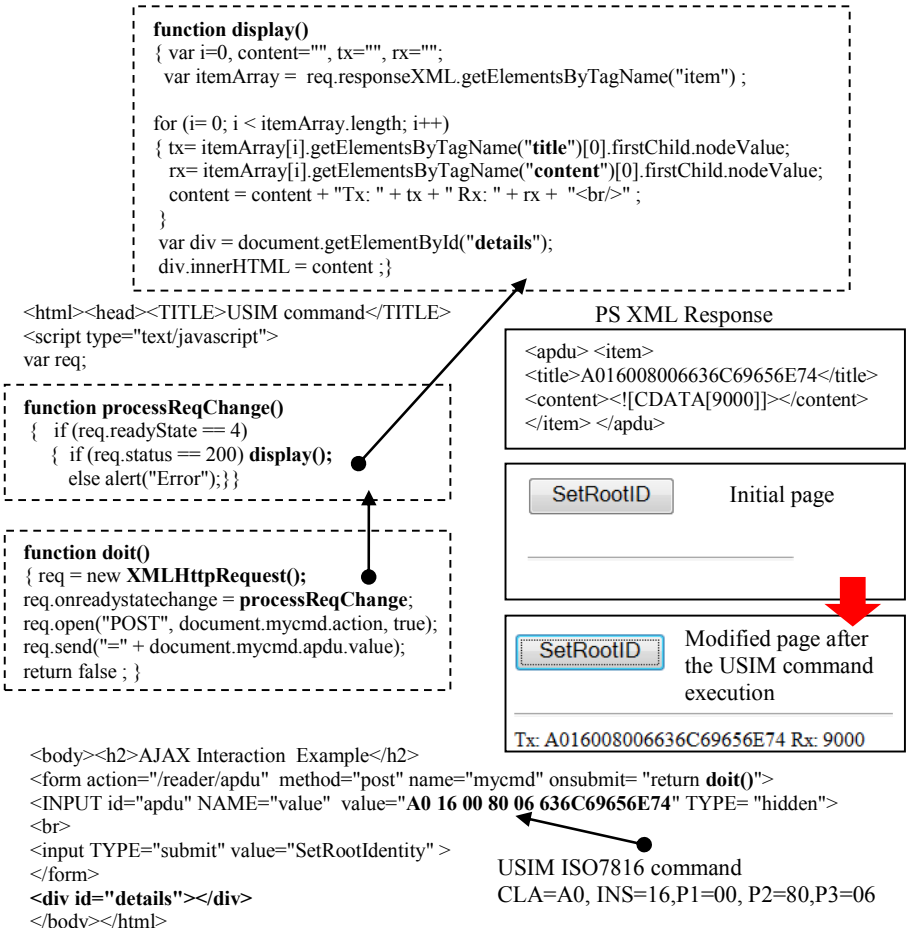

Fig. 6. Example of AJAX interaction with the USIM

4.2.4 The OPENID Server

The OPENID server authenticates the user thanks to its USIM device. It uses ClassI URLs for that purpose. An example of authentication page is shown by figure 1 (part C). This login page is usually downloaded without any security (i.e. with a simple HTTP request). It includes a form typically associated with an action such as:

http://127.0.0.1:8080/~url=www.openid.com/login?Cookie=sid

This URL authenticates the USIM thanks to a TLS session with PKI mutual authentication. For WEB programming issues, it may be necessary to include session id (i.e. the Cookie used by the OPENID server, if any).

5 Conclusion

In this paper we have presented an innovative Identity platform based on OPENID for mobile operator. It has been successfully experimented with Galaxy SII Android phones.

References

1. http://www.gartner.com/it/page.jsp?id=1622614
2. http://trends.builtwith.com/docinfo/OpenID
3. GSM Association, NFC Handset APIs & Requirements Version 2.0 (2011)
4. SIM Alliance, Open Mobile API specification V2.02 (2011)
5. Secure Element Evaluation Kit for the Android platform - the 'SmartCard API', http://code.google.com/p/seek-for-android/
6. http://www.kandroid.org/online-pdk/guide/telephony.html
7. http://source.android.com/source/initializing.html
8. What is android?, http://developer.android.com/guide/basics/what-is-android.html
9. http://www.devicefidelity.com/
10. http://www.tyfone.com/
11. Vedder, K.: "Smart Cards", ETSI Security Workshop (2006), http://www.etsi.org/WebSite/document/Workshop/Security2006/Security2006S1_3_Klaus_Vedder.pdf
12. ISO 7816, Cards Identification - Integrated Circuit Cards with Contacts. The International Organization for Standardization (ISO)
13. Chen, Z.: Java Card™ Technology for Smart Cards: Architecture and Programmer's (The Java Series). Addison-Wesley (2002)
14. NFC Forum, http://www.nfc-forum.org
15. Global Platform, http://www.globalplatform.org/
16. JSR 177, Security and Trust Services API (SATSA) for JavaTM Platform. Micro Edition
17. TS 102 613, Technical Specification Smart Cards; UICC - Contactless Front-end (CLF) Interface; Part 1: Physical and data link layer characteristics (Release 7)
18. RFC 2246, The TLS Protocol Version 1.0, IETF (1999)
19. http://openid.net/
20. Urien, P.: An OpenID Provider based on SSL Smart Cards. In: 7th IEEE Consumer Communications and Networking Conference, CCNC 2010, January 9-12 (2010)
21. http://community.livejournal.com/lj_dev/683939.html
22. Garrett, J.J.: Ajax: A New Approach to Web Applications (February 2005), http://www.adaptivepath.com/ideas/essays/archives/000385.php
23. XMLHttpRequest Level 2, W3C Working Draft 17 (January 2012)
24. Urien, P.: Collaboration of SSL smart cards within the WEB2 landscape. In: CTS 2009 (2009)
25. 3GPP TS 27.007 standard
26. Urien, P., Pujolle, G.: EAP support in smartcards. IETF Draft (2002-2012)