Smart Card based Solution for Non-Repudiation in GSM WAP Applications

CRISTIAN TOMA, MARIUS POPA, CATALIN BOJA
Economic Informatics Department
Academy of Economic Studies
Romana Square No. 6, Bucharest
ROMANIA
cristian.toma@ie.ase.ro marius.popa@ase.ro catalin.boja@ie.ase.ro
http://www.ase.ro

Abstract: The paper presents security issues and architectures for mobile applications and GSM infrastructure. The article also introduces the solution for avoiding denial of service from WAP applications using WIM features. The first section contains the structure of GSM network from voice and data point of view. The security in GSM network is presented in second section. The third section presents a solution for realizing mobile subscriber non-repudiation. The solution is based on the HTTP protocol over WAP.

Key-words: mobile security, m-application security, SAWNR - Secure Application for Wireless Non Repudiation.

1. GSM Overview

The GSM – Global System for Mobile Communication includes many technologies for voice and data transmission. For GSM there are few distinctive types of applications:

- **Pull** typical applications – Web/WAP applications over HTTP (the web browser of the mobile device is requesting a resource and the service provider responses with the resource using usually HTTP); For more information please see [9], [5], [6];
- **Push** typical applications (the subscribers have signed up for a service and the information is going to the mobile device using SMS – Short Message Service, MMS – Multimedia Message Service, Push SI – Service Indication, SL – Service Location and CO – Cache Operation); For more information please see [6];
- **SIM** Toolkit– Subscriber Identity Module Toolkit applications that are running in the SIM Smart Card using native code or Java Card technology; For more information please see [3], [4], [5], [6];
- **Native applications** which are running on the top of operating system of the mobile device. Usually these applications are developed in C/C++ for mobile OS such as: Symbian OS, Linux ALP and Microsoft Mobile OS;
  Applications written in **Java Micro-Edition** or in **C# for .NET Compact Framework** that are running in proper virtual machines on the mobile device. Please see [5], [6];
- **Hybrid Applications** that provide complex services such as SIM Sentry for Multimedia Mobile Content Digital Rights Management, Midlets with Java Smart Card technology solution for mobile banking or electronic purses, Web WAP applications for mobile streaming over RTP and RTSP in GSM networks. For more information please see [3], [4], [5], [6], [7], [8].

In this paper we focus on Web WAP – Wireless Application Protocol applications security and in particular we focus on the solution to avoid repudiation. The basic GSM network architecture is presented in figure 1. The main components involved in voice and data transmission are the following:

- **BSS** – Base Station Subsystem. It controls the quality of the links from GSM radio interface and contains BTS and BCS.
- **BTS** – Base Transceiver Station. Controls the "antennas" and maintains the communication through a duplex radio channel. It supports configurations for: electro-magnetic power, radio channel used for broadcasting, BSIC – Base Station Identity Code. Its main functionalities are: message encryption, channel coding and modularization.
  BCS– Base Station Controller administrates and controls base stations and radio channels. It provides the coding implementation for voices and messages and it manages the data localization.

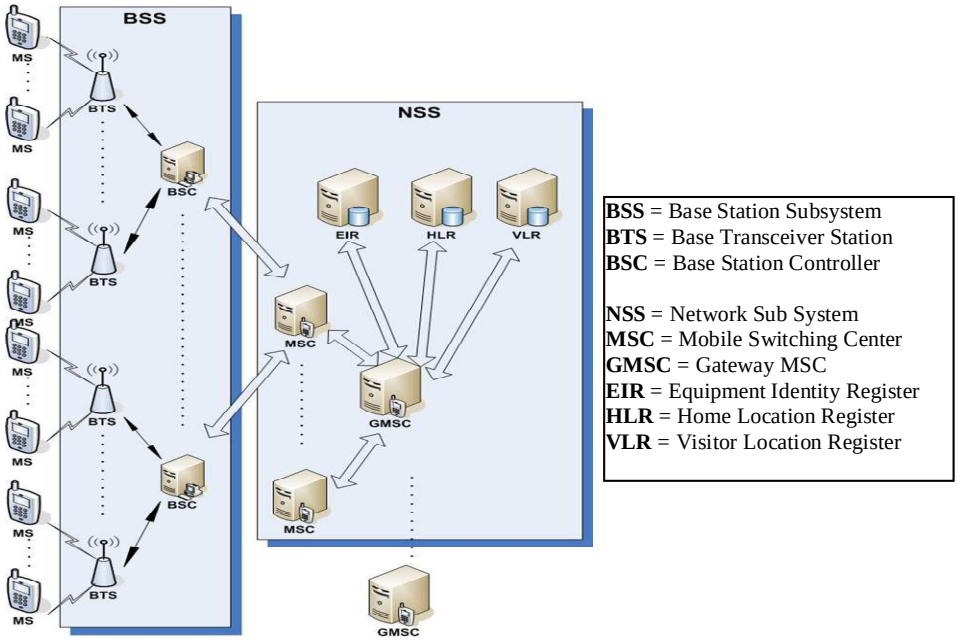

Fig. 1. GSM Network Architecture

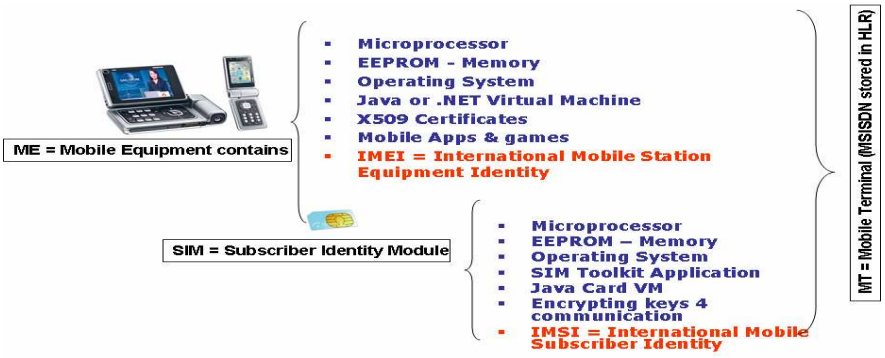

Fig. 2. GSM Mobile Equipment Structure.

- **NSS** – Network Sub System. It contains MSCs, Databases such as VLR, HLR, EIR and AuC and adaptation modules such as XC, IWF, EC. NSS provides the following functionalities: management of communication link with other mobiles, land and satellite networks, management of mobile subscribers from other BSCs, and the management of the subscribers using data from AuC, EIR, VLR and HLR databases
- **MSC** – Mobile Switching Center. It contains switching subsystems (e.g. for PBX signaling and for communication signaling over SS7 with other MSCs) and control subsystems
- **HLR** – Home Location Register. It stores the identity - TMSI (Temporary Mobile Subscriber) and the subscribers' parameters including the MSISDN and the service type (e.g. 10 SMS, 80 minutes for 1 month).
- **VLR** – Visitors Location Register. It is a mirroring database of HLR for temporary subscribers of another VLR area.
- **EIR** – Equipment Identity Register. It is the centralized database with IMEI (unique number for each provider-device) for each mobile device.
- **AuC** – Authentication Center. The following functionalities and responsibilities are included: authorization process for the subscriber access into mobile network for encrypting transmission on radio path and for assign of the temporary voice transmissions. Figure 3 presents the overall processes for authentication and

Figure 2 presents the main concept used in GSM for end-user device: the mobile is a two in one computer. The first computer is represented by the SIM. Actually, the SIM is a smart card with a microcontroller, three types of memory area (ROM, EEPROM and RAM) and I/O ports for outside communication (usually in half duplex mode).

The mobile device itself is the second computer. It also includes a microprocessor, different types of memory areas and an operating system.
The GSM evolution is based on this infrastructure and the security of the entire system depends on many factors.

2. Architectures for Voice and Data Security in GSM

This section focuses on the security mechanisms involved within the mobile data and the confidentiality of mobile communication.

Fig. 4 and 5 present a more detailed technical view of the systems' security process. Fig. 4 shows the main network items involved in voice and data security, and fig. 5 presents the logical flow of the data. AuC – Authentication Center plays the main part in voice and data authentication and integrity.

Its main functionalities and responsibilities consist in: authorizing the subscribers' access into the mobile network; encrypting transmission on radio path and assign of the temporary identity - TMSI (Temporary Mobile Subscriber Identity).

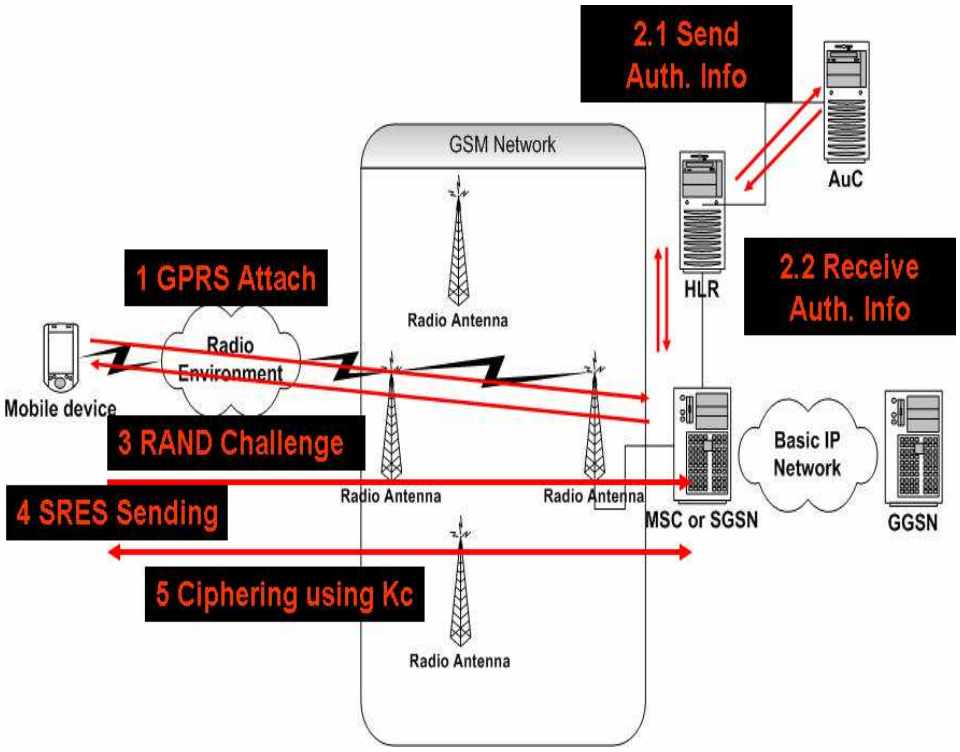

Fig. 3. Processes for authentication and confidentiality of mobile voice and data transmission

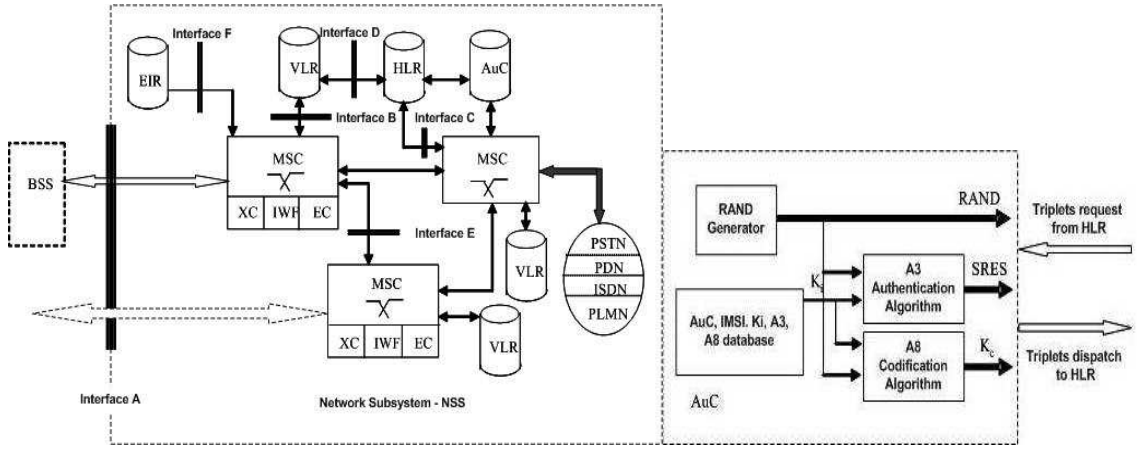

Fig. 4. The details of the security mechanisms

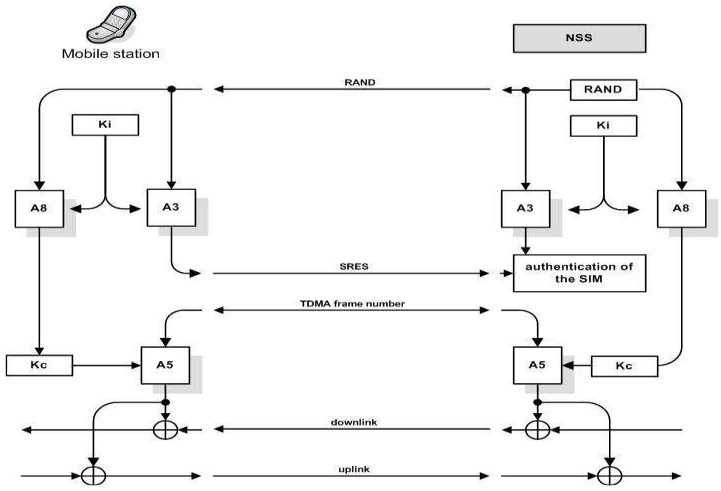

Fig. 5. Overview of data workflow for authentication and confidentiality [3]

The authentication process of the mobile subscriber is described as it follows:

- The mobile station sends the SIM's IMEI to the HLR through the "Net Attach" (or GPRS Attach).
- A triplets request is sent to the AuC from HLR (all HLR, AuC and SIM suppose to store same Ki) through the "Send Auth. Info" message (contains SIM's IMSI)
- The AuC generates a response that contains:
  - RAND (random number – challenge)
  - Kc – encryption key that is a result of the A8 algorithm. The A8 algorithm uses the stored identity key – Ki from AuC corresponding to the received IMEI.
  - SRES – Signed RESponse generated through A3 Authentication Algorithm with RAND and Ki as input
- The HLR receives the triplets and sends to the mobile only RAND
- The mobile device must be enabled using Ki from SIM, A3 and A8 algorithms in order to reconstruct Kc and SRES. It then sends the SRES to the HLR via MSC or SGSN (in GPRS only).
- If the SRES received by the HLR from the mobile device is the same with the one that is received from the AuC, then the authentication is done and Kc will be used for ENCRYPTION.

The described architecture had been considered at the beginnings of the GSM and has been used since (including in GPRS, EDGE, UMTS networks). So in terms of authentication and encryption the path from mobile device to the WAP Gateway is secured. In figure 6 is a typical GSM GPRS architecture.

Basically the voice is going via MSC over SS7 protocol and the data traffic goes over SGSN and GGSN. Some certain features of the SGSN and GGSN equipments must map and relate the TCP/IP protocols stack over radio from the mobile phone with the TCP/IP protocols stack over Ethernet from the service provider. In practical approaches the hardware and software components which map and relate TCP/IP protocols over the radio and Ethernet are generically called WAP Gateway. Figure 6 is very simple to understand if it is analyzed together with the previous sections and figures from this paper. If the link between the WAP Gateway and service provider is secured using SSL/TLS or IPSec and there are secure policies implemented at router level, then the problem of encryption in minimal terms of security requirements is fulfilled.

The problem is about the mobile subscriber non-repudiation. How a mobile subscriber can be indubitable linked with a web HTTP request? How a mobile subscriber can not sustain that the mobile device disappeared and someone used its device for requesting services? For the answer at these kinds of questions, we provide a solution – SAWNR – Secure Application for Wireless Non Repudiation – which uses J5EE Servlets in order to send WML – Wireless Markup Language and WMLS – WML Script code [9], [10] to the mobile device. The WML and WMLS code helps the mobile Internet browser to digitally sign the data with the private key from the WIM – Wireless Identification Module.

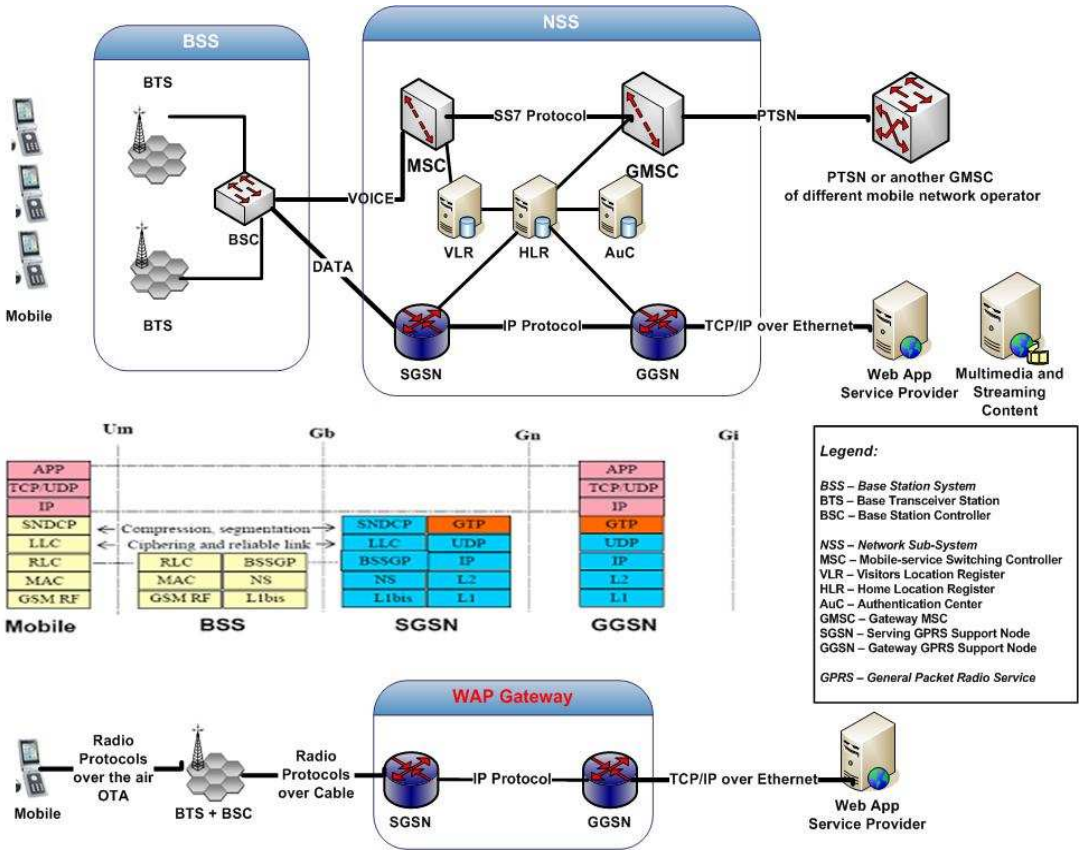

Fig. 6. WAP Gateway features in a GPRS network

3. Smart Card Communication Technology for SIM

The mobile browser in order to get certificate information or even to sign the text that is send as response over HTTP, will communicate with Identity Module using messages passing model (fig. 2). The communication between host application and the applets/applications from smart card suppose to transmit some APDU – Application Protocol Data Unit from host (mobile Internet Browser application) to CAD – Card

Acceptance Device (the physical connection SIM/WIM – Subscriber Identity Module/Wireless between SIM and mobile device), and then the same bytes strings are sent from the CAD to the card applet/application. The applet/application receives those bytes strings, is parsing the bytes and then will send back following the reverse path: Applet-CAD-Host. An APDU is composed from standard bytes blocks conform ISO/IEC 7816-3 and 7816-4. Respecting the standards the applet receives directly from CAD, *APDU Commands* and sends back to CAD, *APDU Responses*. The communication between the card reader and the card is physically realized through data link protocol.

This protocol is likely data link level protocol from protocol stack ISO/OSI. The link protocol, defined in ISO/IEC7816-4, has four alternatives: T=0 – byte oriented, T=1 – bytes arrays oriented, T=USB – oriented Universal Serial Bus or T=RF – radio wave oriented, Radio Frequencies.

APDU Commands

The general template for an ADPU Command is shown in table 1:

Table 1. The structure for an APDU Command

APDU Command						
CLA	INS	P1	P2	Lc	Data Field	Le
Header-mandatory				Body-optional		

There are other four specific structures for an APDU Command, but these structures are used only in data link protocol T=0.

The explication for the fields from the APDU command is the following:

- **CLA** – is one byte – 2 hexadecimal digits, and has different predefined values conform standard ISO7816. For instance, between the value 0x00 and 0x19 are values for accessing file system and security operations, from 0x20 to 0x7F are reserved for future using, and from 0x80 to 0x99 can be used for applets' specific instructions implemented by developers but between 0xB0 and 0xCF are specific instructions for all applets and not for a particular one. As matter of fact the most used value for this field is 0x80;
- **INS** – is one byte, and the standard defines a specific instruction in the field CLA. For instance, when CLA has the value between 0x00 and 0x09, but INS has the value 0xDC – means the field INS could have predefined values established by developers but according with the standard. For example, the developer chooses for this field the value 0x20 for signing the sent text from card if and only if the CLA field is 0x80;
- **P1** – this represents the first parameter for an instruction and has one byte. This field is used when the developers want to send some parameters to the applet or want to qualify the INS field;
- **P2** – this is the second parameter for an instruction and has one byte. Is used for the same scope like P1;
- **Lc** – has one byte, is optional and represents the bytes length for the field Data Field;
- **Data Field** – is not fixed and has a bytes' length equal with the value from the field's value Lc. In this field are stored data and parameters which are send from host application to applet;
- **Le** – stores the maxim number of bytes that should have Data Field from APDU Response (the number of bytes from response could be any value from the range 0 and the value from this field).

Practically a host application sends to the CAD but the CAD sends to the applet the same APDU commands with structures and values which respect the standards.

APDU Responses

The structure for an APDU Response is simple and is shown in the table 2:

Table 2. The structure for an APDU Response

APDU Response		
Data Field	SW1	SW2
Body-optional	Trailer-mandatory	

The fields' explication for APDU Response is the following:

- **Data Field** – has variable length which is determined by the value of the byte field Le from the APDU Command;
- **SW1** – has one byte and represent the status word 1;
- **SW2** – has one byte and represent the status word 2.

The fields SW1 and SW2 are parsed and interpreted together, but a communication process is called *complete* if there were no problems

(SW1=0x61 and SW2=0x90 or any-0xnn) or if there were only warnings (SW1=0x62 or SW1=0x63 and SW2 contains the warning code). A communication process is called *failed* if there were execution errors (SW1=0x64 or SW1=0x65 and SW2 has the error code for execution) or checking errors (SW1=from 0x67 to 0x6F and SW2 has the code for checking error).
In message passing model there is all the time a selected applet/application in the card and when is receiving an APDU Command. For signing purpose there is a standard Java card applet that runs within Java Card Virtual Machine from SIM/WIM. The study of developing Java Card applications for GSM SIM/WIM is included by the authors in various research projects and papers [5],[6],[7].

4. SAWNR – Secure Application for Wireless Non Repudiation

In this section we present SAWNR – Secure Application for Wireless Non Repudiation in terms of network traffic analysis and in terms of business flow. In terms of communications between the mobile browser application and the SIM smart card application will be presented in future papers. In figure 7 from the mobile browser is generate a HTTP GET Request to the Java servlet 'SignDeck' from URL Context 'NokiaServletSecurity' - the complete URL in our tests were:

http://10.2.4.244:8090/NokiaServletSecurity/SignDeck

The web server (Apache Tomcat) is running on machine with IP address 10.2.4.244 and listen port 8090. The Java web server servlet generates the HTTP Response from figure 7, starting with header 'HTTP/1.1 200 OK'. The response is a WML code which contains two pairs of <card>…</card> tags (two decks).
The second phone screen from figure 7 is obtained using 'Confirm' option from first deck. Because the mobile web browser display only one pair of <card>…</card> tag (a deck) in a certain period of time, the <go href='#Sign' /> tag instructs the mobile web browser to go in the deck <card id='Sign'>…</card>.

HTTP Network Analysis

```
GET /NokiaServletSecurity/SignDeck HTTP/1.1
Host: 10.2.4.244:8090
user-agent: Nokia Mobile Browser 4.0
Accept: application/x-wap-prov.browser-bookmarks, application/x-wap-prov.browser-settings, applic
vnd.wap.hashed-certificate, application/vnd.nokia.ringing-tone, application/vnd.wap.cert-response
vnd.wap.wmlscriptc, application/vnd.wap.xhtml+xml, application/vnd.syncml+wbxml, application/x-no
vnd.wap.wmlc, application/xhtml+xml, image/vnd.wap.wbmp, text/x-vcalendar, text/x-co-desc, text/x
multipart/*, text/vnd.wap.wmlscript, text/vnd.wap.wml, text/plain
accept-charset: ISO-8859-1, US-ASCII, UTF-8; Q=0.8, ISO-10646-UCS-2; Q=0.6
accept-language: en, fi
Via: Nokia Activ Server 2.0 Professional (build 2451A)
X-Network-Info: UDP,127.0.0.1,security=0
Accept-Encoding:
Connection: Close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/vnd.wap.wml;charset=ISO-8859-1
Content-Length: 757
Date: Thu, 15 Nov 2007 10:31:15 GMT
Connection: close

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml><card id='order' title='Confirm Order'>
<p>
You have ordered<br/>
Java Card Programming Handbook : $$99.50<br/>
Smart Card Handbook: $$199.50<br/>
<do type='options' label='Confirm'>
<go href='#Sign' />
</do>
</p>
</card>
<card id='Sign' title='Authenticate order'>
<p>
Please sign the order<br/>
Java Card Programming Handbook : $$99.50<br/>
Smart Card Handbook: $$199.50<br/>
<do type='options' label='Sign'>
<go href='CryptoScript#sign()' />
</do>
<do type='options' name='a' label='Submit'>
<go href='SignDeck' method='post'>
<postfield name='signed' value='$(signed)'/>
</go>
</do>
</p>
</card>
```

Mobile Web Page

Fig. 7. HTTP Network Analysis for requesting two books using a mobile Web WAP application

HTTP Network Analysis

```
GET /NokiaServletSecurity/CryptoScript HTTP/1.1
Host: 10.2.4.244:8090
user-agent: Nokia Mobile Browser 4.0
Accept: application/x-wap-prov.browser-bookmarks, application/x-wap-prov.brows
settings, application/vnd.wap.signed-certificate, application/vnd.wap.hashed-
certificate, application/vnd.nokia.ringing-tone, application/vnd.wap.cert-
response, application/vnd.wap.mms-message, application/vnd.wap.wmlscriptc,
application/vnd.wap.xhtml+xml, application/vnd.syncml+wbxml, application/x-
nokiagamedata, application/vnd.wap.wbxml, application/vnd.wap.wmlc, applicati
xhtml+xml, image/vnd.wap.wbmp, text/x-vcalendar, text/x-co-desc, text/x-vcard,
image/gif, text/html, text/css, application/*, multipart/*, text/
vnd.wap.wmlscript, text/vnd.wap.wml, text/plain
accept-charset: ISO-8859-1, US-ASCII, UTF-8; Q=0.8, ISO-10646-UCS-2; Q=0.6
accept-language: en, fi
Via: Nokia Activ Server 2.0 Professional (build 2451A)
X-Network-Info: UDP,127.0.0.1,security=0
Accept-Encoding:
Connection: Close

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/vnd.wap.wmlscript;charset=ISO-8859-1
Content-Length: 206
Date: Thu, 15 Nov 2007 10:31:20 GMT
Connection: close

extern function sign() {
var signed;
signed = Crypto.signText('Java Card Programming Handbook: $99.50, Smart Card
Handbook: $199.50',5,0,'');
WMLBrowser.setVar('signed',signed);
WMLBrowser.refresh();}
```

Mobile Web Page

Fig. 8. HTTP Network Analysis for obtain the function 'sign()' from Java Servlet 'CryptoScript'

When the mobile subscriber click 'Sign' option in mobile screen from figure 8, then the mobile web browser request the 'CryptoScript' resource and run the 'sign()' function received in figure 8 HTTP traffic analysis section (because of tag: <go href='CryptoScript#sign()' />). The figure 8 represents the HTTP Request and Response for the resource: (complete URL = http://10.2.4.244:8090/NokiaServletSecurity/CryptoScript). The mobile browser receives the WML Script code which contains only the 'sign()' function. The mobile browser runs 'sign()' function and it is obtained the behavior from figure 9. The mobile subscriber is instructed to choose a proper digital X509 Certificate which is stored in WIM.

After that the end-user inserts the password '12345' and the WML Script variable 'signed' contains now the text 'Java Card Programming Handbook: $99.50, Smart Card Handbook: $199.50' which is now digitally signed.

In the last mobile screen from the figure 9 the browser returns in the last deck (<card> tag pair). The last deck was obtained through HTTP GET request from figure 7. Now the mobile subscriber will choose option 'Submit', after the signing process in order to avoid repudiation. The mobile web browser do a HTTP POST Request accordingly with the WML code (<do type='options' name='a' label='Submit'><go href='SignDeck' method='post'><postfield name='signed' value='$(signed)'/></go></do>).

Fig. 9. Behavior for running the 'sign()' function by the mobile web browser

HTTP Network Analysis

```
POST /NokiaServletSecurity/SignDeck HTTP/1.1
Host: 10.2.4.244:8090
Content-Type: application/x-www-form-urlencoded
user-agent: Nokia Mobile Browser 4.0
Accept: application/x-wap-prov.browser-bookmarks, application/x-wap-prov.browser-
settings, application/vnd.wap.signed-certificate, application/vnd.wap.hashed-
certificate, application/vnd.nokia.ringing-tone, application/vnd.wap.cert-response,
application/vnd.wap.mms-message, application/vnd.wap.wmlscriptc, application/
vnd.wap.xhtml+xml, application/vnd.syncml+wbxml, application/x-nokiagamedata,
application/vnd.wap.wbxml, application/vnd.wap.wmlc, application/xhtml+xml, image/
vnd.wap.wbmp, text/x-vcalendar, text/x-co-desc, text/x-vcard, image/gif, text/html,
text/css, application/*, multipart/*, text/vnd.wap.wmlscript, text/vnd.wap.wml, text/
plain
accept-charset: ISO-8859-1, US-ASCII, UTF-8; Q=0.8, ISO-10646-UCS-2; Q=0.6
accept-language: en, fi
Via: Nokia Activ Server 2.0 Professional (build 2451A)
X-Network-Info: UDP,127.0.0.1,security=0
Accept-Encoding:
Date: Thu, 15 Nov 2007 12:38:25 GMT
Content-Length: 1685
Connection: Close

signed=AQEAgMR568nU%2Bwf%2BsKLoLxLrkBVYYiNE%2FJBXlzfF5G%2FEUEflbD%
2FOAhqHIFNXTstKBgEec0%2BoU8yF7%0Dnh8Pr0%2B1mqaNwQMLFaE%2FFU%
2FGPR2GuxE93PyWTx3LrTFwVR62rcRaFaTTVaC83MKmZxENmgKUSA4h%0DYJu1HQH%2BzB9qJsV%2BAhza%
2FvxA7cDA7QwggOwMIIDGaADAgECAgEAMA0GCSqGSIb3DQEBBQUAMIGOM%
0DQswCQYDVQQGEwJVUzELMAkGA1UECBMCTUExDzANBgNVBAcTBkJvc3RvbjEOMAwGA1UEChMFTm9ra%
0DWExDDAKBgNVBAsTA05NUDEkMCIGA1UEAxMbTW9iaWxlIEludGVybmV0IFRvb2xraXQgMy4xMR0wG%

....

0Dp4MaqmekLVhxiGkoxAB%2BtgZKGwHd1cFsQGPRthLIA9ievH%2FaYN3hgbq6hdhbKG8P%2BZupFnJwPj23%
0D6sBAGoBAERKYXZhIENhcmQgUHJvZ3JhbW1pbmcgSGFuZGJvb2s6ICQ5OS41MCwgU21hcnQgQ2FyZ%
0DCBIYW5kYm9vazogJDE5OS41MA0BB9cACwAPAAwAJgAUHTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/vnd.wap.wml;charset=ISO-8859-1
Content-Length: 220
Date: Thu, 15 Nov 2007 10:31:30 GMT
Connection: close

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/
wml_1.1.xml">
<wml><card id='sign' title='Sign-Result'>
<p>
Thank you for the order<br/>
</p>
</card>
</wml>
```

Mobile Web Page

Fig 10. HTTP POST Request for sending the digitally signed text in order to avoid repudiation.

The HTTP POST Request contains the variable signed = AQEAg…AJgAU , which represents the text signed with the public key RSA – Rivest, Shamir, Adleman cryptographic algorithm (for full details please see [1], [2]). The RSA algorithm uses the private key stored in WIM – Wireless Identification Module and it access with the PIN – Personal Identification Number.

The 'SignDeck' servlet through HTTP POST Request (generated by the mobile browser) receives the 'signed' parameter and verifies its validity searching in a database. The validity of the signature is processed using the public key from X509 Certificate of the mobile subscriber. If everything is ok, the Java servlet generates HTTP Response which contains a WML code (mobile web page) that informs the end-user that everything was successfully processed.

The Java servlets were developed in optimal mode in order to generate as few as possible HTTP Request (two HTTP GET Request and a HTTP POST Request). There are only two Java servlets: 'SignDeck' which have different implementations for HTTP GET and POST Requests, and the 'CryptoScript' servlet which generates the function 'sign()' in WML Script code.

5. Conclusions

Based on the analysis that were made in lab using an Nokia N95 device and a NetFront mobile browser it has been highlighted that the difference between processing time and access time are not significant. On average the process of digital signing takes at most three seconds. Future research will be conducted on beta testing the smart electronic signature solution that has been embedded in a SIM Java cardlet. This approach is developed to be accessed by a Java Micro-Edition Midlet, defined by JSR177.

The solution presented here has many utilities in the public online services which are extended on mobile devices because it allows developing secure environments with fewer resources because everything is taking place at software level and it doesn't interfere with the existing infrastructure. Regarding the impact on mobile business solutions and on public mobile services, the authors are continuing this research in two important research contracts that financed by the Romanian Government through two complex development research projects, module MATNANTECH (Materials and

Nanotechnologies) and module AMCSIT of Romanian Excellence Research Program.

References

[1] William Stallings, Cryptography and Network Security, 3/E, Prentice Hall, 2003.

[2] Bruce Schneier, Applied Cryptography 2nd Edition: protocols, algorithms, and source code in C, John Wiley & Sons, Inc. Publishing House, New York 1996.

[3] Wolfgang Rankl & Effing, "Smart Card Handbook 3rd Edition", John Wiley & Sons Publishing House, USA 2004

[4] Zhiqun Chen, "Java Card Technology for Smart Cards – Architectures and Programmer's Guide", Addison Wesley, 2004

[5] Cristian TOMA, "Tutorial on Java Smart Card electronic Wallet Application", Informatics Security Handbook, AES Publishing House, Romania 2006

[6] Cristian TOMA, "Secure Patterns and Smart-card Technologies used in e-Commerce, e-Payment and e-Government", Informatics Security Handbook, AES Publishing House, Romania 2006

[7] Ion IVAN, Cristian TOMA, Catalin BOJA, Marius POPA, "Secure Architecture for the Digital Rights Management of the M-Content", ISP'06 of the WSEAS Conference in Venice, Nov. 2006.

[8] Ion IVAN, Cristian TOMA, Catalin BOJA, Marius POPA, "Secure Platform for Digital Rights Management Distribution", WSEAS Transaction on Computers, 2006.

[9] http://www.forum.nokia.com – the links for WAP, Browsing, Symbian OS, Java Micro Edition

[10] http://www.w3schools.com – the links for WAP, WML, XHTML and XML