Generic, Secure and Modular (GSM) Methodology for Design and Implementation of Secure Mobile Applications

Feng Zhang[1], Ioannis Kounelis[1,2], and Sead Muftic[1]
[1]Communication Systems
School of Information and Communication Technology
Royal Institute of Technology, Stockholm, Sweden
[2]Institute for the Protection and Security of the Citizen,
Joint Research Centre (JRC), European Commission, Ispra (VA), Italy
{fengz, kounelis, sead}@kth.se

***Abstract*—The generic, secure and modular methodology, described in this paper, provides a generic approach for the design and development of secure mobile applications. It is applicable to multiple mobile phone platforms and mobile operating environments. This approach treats a mobile application in a holistic way and structures it into four groups of modules: user interface modules, communication modules, security modules, and business logic modules. These four groups of modules can be designed and implemented independently and finally be integrated together. This approach not only simplifies the process of design and development of mobile applications, but also improves the reusability and robustness of mobile applications. In addition, this paper proposes a trusted layer model for designing the security modules of mobile applications, which provides generic application interfaces and comprehensive data protection. The paper finally gives an example of a secure mobile application, called SAFE Mobile Wallet, which was designed and implemented using GSM methodology.**

Keywords - mobile; generic; secure; modular.

I. INTRODUCTION

With the significant growth of mobile technologies, mobile phones have already become one of the most important accessories in people's everyday lives all over the world. Hundreds of thousands mobile applications exist today and their number is largely increasing every day. Until March 2012, only Apple's App Store offered over 500,000 iPhone applications [1]. There are various platforms, such as Android, Research In Motion (RIM), Symbian, Windows Phone, iOS (Apple), etc., allowing third parties to easily develop, test and deploy various mobile applications. Based on the reasons mentioned above, a lot of researchers, developers and fans of mobile technologies make great efforts today to design and develop various mobile applications.

At the moment, there are numerous hardware and software platforms used by different mobile devices. Therefore, the methodology of designing and implementating mobile applications varies. In addition, even though there are some generic approaches for building mobile applications [2], [3], these approaches have not taken security into account. Many security flaws [2] violate users' privacy [3] and data security and eventually affect wider deployment of those applications.

In order to solve the issues described above, this paper proposes the, here called, Generic, Secure and Modular (GSM) methodology for the design and implementation of mobile applications. The distinguished features of our GSM methodology are the following:

- It treats mobile applications in a holistic way, meaning that our methodology can be applied to develop applications for any platform, using any programming language;
- It uses a modular approach and structures mobile applications into several groups of modules. Each module is independently designed and implemented, making the implementation results reusable;
- It provides an approach for the design and implementation of security modules, which provides comprehensive security features for the mobile applications. The details are described in Section 3.4.

Therefore, we expect that the GSM methodology is a significant contribution to the process of designing, developing and deploying secure mobile applications.

The next section continues with related work and then our methodology is described in detail, covering all four modules. Finally, there is a conclusion section.

II. RELATED WORK

This section gives an overview of what has been already published related to secure mobile applications. Most of these contributions indicate how to design or implement a mobile application and how to integrate some specific features into it. Some describe suggestions for mobile application development, while others point out the challenges for the development of secure mobile applications.

Ueyama, Pinto and Madeira proposed a generic approach for constructing mobile device applications [4]. It is a middleware solution, based on generic components approach, to build adaptive applications, which can be deployed across heterogeneous devices with minimum resources overhead. It effectively solved the adaptability and flexibility issues.

Andre and Segarra proposed another generic approach, called MoleNE [5]. It works in such a way that it provides a

 ISBN: 978-1-61208-209-7

set of generic base services as the abstraction of concepts to the developers so that they could use it to create customized applications.

Serhani, Benharref, Dssouli, and Mizouni listed the main issues to consider when developing mobile applications and mobile Web services [6]. A framework, which includes a mobile client environment and backend server/mobile Web services, is described. Finally, an example of a mobile application, developed by using the proposed framework, is given.

Ray Gonzales gives some suggestions on how to design mobile applications in [7]. These suggestions include how to structure user interfaces based on the size of a screen, some pre-defined design conventions, etc. in order to achieve convenient and comfortable interactions between users and mobile devices.

Christien Rioux presented challenges for developing secure mobile applications in [8]. He also analyzed security features for three popular mobile platforms: Windows Mobile, RIM Blackberry, and Google Android and listed some security vulnerabilities and malicious code for these platforms.

Our conclusion is that today there are not so many research and development results in the area of the generic methodology for developing secure mobile applications. Specially, few results concentrate on the security during the design and implementation phases. The methodology proposed in this paper addresses this shortage.

III. GSM METHODOLOGY

As shown in Figure 1, our GSM methodology structures mobile applications into four groups of modules: User Interface (UI) modules, communication modules, security modules, and business logic modules. The former three modules are generic, meaning that these three modules can be reused. The last module is specific, different for each mobile application. The reason for this approach is that these four groups of modules are necessary for most of the secure mobile applications. In other words, in order to create a secure mobile application, at least these four groups of modules should be included.

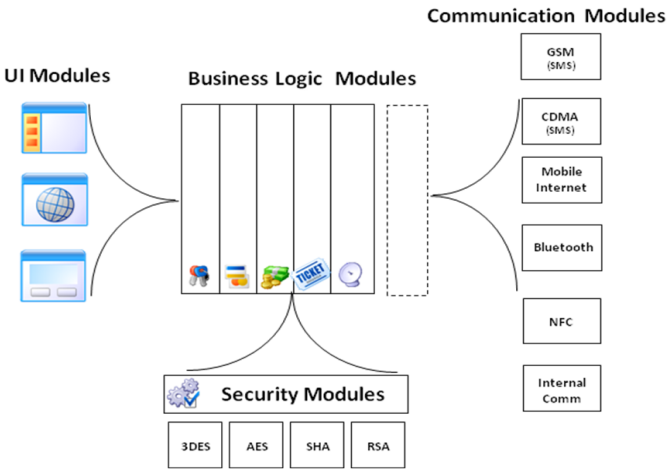

Figure 1. Internal structure of secure mobile application

Every mobile application must have user interfaces for displaying information to users and interacting with them. Some of them are simple and some are complex; this depends on the hardware and operating system of mobile phones. All mobile applications need also to communicate either with open networks or with internal modules of mobile devices. Communication modules exchange messages with networks, with other devices, and/or with other internal modules of mobile devices. Next, the security of mobile applications is also very important. Therefore, security modules that provide security features, such as generation of RSA keys, encryption, digital signatures, etc. are also necessary. These three groups of modules are not sufficient to create a complete secure mobile application, since they are separated and independent of each other and they do not perform any function specific to an application. Hence, business logic modules are needed to link all other modules together and coordinate them based on the services or functionalities provided by a specific mobile application.

GSM methodology also organizes the process of designing and creating mobile applications into three steps: (a) analyzing target platforms and devices, (b) designing and implementing the four groups of modules described above, and finally, (c) integrating all the modules and performing tests on the actual devices. As Greg Nudelman said, "*One of the challenges of mobile application design is understanding both the capabilities and limitations of each platform*" [9]. Therefore, the first step is to analyze target platforms in which the mobile application will be loaded. It includes technical details of the target platform, such as which programming language should be used, which libraries and Application Programming Interfaces (APIs) are supported, etc. The next step for developers is to design and implement the generic and specific modules as described in this paper and the final step is testing the application on the mobile devices.

A. *User Interface (UI) Modules*

Each mobile application must provide user interfaces for getting input from a user and for displaying output to a user. The design of user interfaces directly influences user experience. The UI may be different for different mobile devices or platforms, depending on the hardware and APIs. Therefore, the design and implementation of UI modules may also be different. There are lots of guidelines and suggestions for the design of UI modules for various devices and platforms. One of the approaches is to implement a generic object, the `UIProvider`, which provides all the necessary UI components, such as `TextField`, `Buttons`, `RadioBox`, etc. that may be used by all mobile applications running on the same platform. Even though the implementation of the `UIProvider` depends on the platform and the corresponding APIs, it should be implemented as generically as possible, so that it can be reused by mobile applications running on the same platform. In this way, various mobile applications could share one `UIProvider`, which saves development time and makes UI components easy and flexible to modify.

 ISBN: 978-1-61208-209-7

From a security point of view, the design of the application should always prompt the user when a security feature is not used. For example, a warning should be used when mail is sent over an insecure connection. Moreover, it is important, when having a dialogue that requires the user's interaction, to mark as default the response that will be on the user's favor, always in respect to his/her security and privacy. For instance, if an application asks the user if it is ok to sign on a page although there is no encryption, the default button should be no.

B. Communication Modules

Mobile applications must always access local or global networks. Therefore, network communication functions, such as sending/receiving Short Message Service (SMS), establishing connections through Bluetooth [10], Wi-Fi [11], 3G [12], etc. should be designed and implemented. As a result, communication modules should perform communications with open networks.

For the communication with open networks, we designed a generic object, the `CommunicationProvider`, which is convenient to manage all the communication channels, such as Bluetooth, General Packet Radio Service (GPRS) [13], Wi-Fi, SMS, etc. The `CommunicationProvider` works as a "coordinator" to support these communication channels, as required. For example, it dynamically and transparently selects the most convenient channel, controls the priority of the channels, adds/removes channels to/from the communication, etc. The priority of communication channels can be configured, depending upon usage requirements. Each specific communication channel is implemented individually and should be tested for the connections with the `CommunicationProvider`. Communication channels should be independent of each other and the procedure of establishing connections should be transparent to users.

Moreover, secure channels should be given priority over insecure ones. So, if there is an option to connect to a page with both http and https the second should be used by default, without requiring user intervention.

C. Security Modules

As explained in previous Sections, security is more and more important for mobile applications. In order to provide comprehensive security services for mobile applications, we designed a `SecurityProvider`, based on our proposed trusted layer model.

As shown in Figure 2, our trusted layer model structures the security modules into four trusted layers: Secure Element (Chip), applets, middleware and applications. The trusted layer model is equivalent with the Open Systems Interconnection (OSI) seven layer model; each layer provides services to its upper layer while receiving services from the layer below.

Layer 1: Secure Element (Chip)

This layer represents the chips used by mobile devices for mobile transactions, such as the Subscriber Identity Module (SIM)/Universal Integrated Circuit Card (UICC) chip, Micro SD chip, etc. The chip contains a microprocessor and a small memory, used in the mobile phone. It is secure and tamper resistant, so that it works as a Secure Element (SE), storing important information, such as credit card numbers, private keys, etc. Moreover, because of the compatibility and mobility of these chips, using them for storage of data ensures that users can easily migrate their data between different mobile devices. The chip layer defines how data is stored in the SE. It also defines the format of the application protocol data unit (APDU) that is used for upper layer to invoke the functionalities provided by the chips.

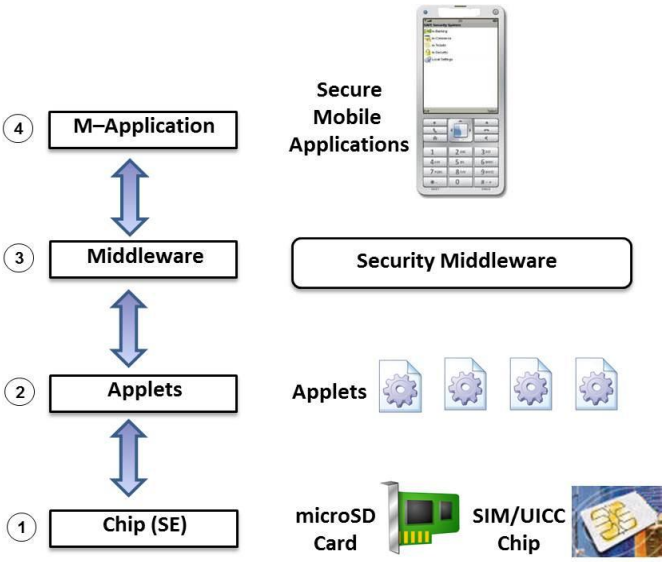

Figure 2. Trusted layer model

Layer 2: Applets

This layer represents the applets installed and functioning in the hardware of a chip. Applets interact with the chips using APDU commands, invoking functions provided by chips and providing those functions to the upper layer.

Layer 3: Middleware

Mobile applications use high-level programming APIs, but applets in the chips understand only APDU commands. Therefore, in order to make the communication between mobile applications and chips as smooth as possible, a middleware is needed. This approach is also suggested in the FIPS 201 (Personal Identity Verification - PIV) standard [14]. The middleware works as a "translator" between a high level application and an applet loaded in the chips. On one hand, it provides APIs to high level applications, so that developers should not need any knowledge about chips and applets loaded in them. As a result, it makes the invocation of the functions, which are provided by the applet in the chips, transparent to mobile applications. On the other hand, the middleware communicates with the chips using APDU commands.

The middleware layer defines how the upper layer software applications invoke the functions provided by applets. It provides APIs, for upper layer mobile applications to invoke the functions provided by the applets. The middleware layer contains the implementation of these APIs,

 ISBN: 978-1-61208-209-7

the role of which is to translate the upper layer application data calls into APDUs and vice versa.

An implementation of the middleware can be seen in [15]. In this paper, we have created a middleware that allows the flow of data from a UICC to the upper layer applications and eventually the user of the mobile device. The core principle of this implementation was to make the whole process transparent to the application developers and even more to the end user. As Saltzer and Schroeder [16] stated in the 1970s, if the security mechanisms make the product or device harder to use, then the users will simply avoid using them.

Layer 4: Applications

The application layer is the one closest to the end-user. It comprises the Graphical User Interface (GUI), interacting with and providing security functionalities for users.

A `SecurityProvider` was designed in order to provide security functionalities. It is designed as follows:

The application layer comprises the interfaces to access the implementation of each security service. And it provides the `SecurityProvider` as an interface to get the instance of these security services. For example, the following pieces of code are used to create an instance of the `GenerateKeyPair` service:

```
SecurityProvider secprovider = new
SecurityProvider();
IGenerateKeyPair genkp =
secprovider.GenerateKeyPair();
genkp.generateKeys(AlgorithmID);
genkp.saveKeys(keypairs);
```

Security services are implemented in the security applets, functioning on the chips. Therefore, during the execution of the codes listed above, the application needs to communicate with a specific applet that implements the invoked security service through the middleware. The user is required to enter a correct Personal Identification Number (PIN) in order for the middleware to access the applet. The PIN-based authentication mechanism is one of the security properties of the applets. Each applet has a default PIN and the user is able to set his/her own PIN. If the user authentication is successful, a secure connection is established between the middleware and the applet. Meanwhile, the middleware translates the service request into corresponding APDU commands, according to the rules and formats defined by the applet. The middleware sends the APDU commands to the applet via the established secure connection. The applet executes the required function and returns the execution result in the form of APDU response code to the middleware, which then translates the code to an application-layer service response.

By this approach, the `SecurityProvider` can provide as many security services as the chip supports. In addition, the application-level security mechanisms can always be added in order to provide higher security. For example, the data exchanged between the mobile application and the applet is encrypted using AES with the hash value of the applet authentication PIN as the symmetric key. In this way, the data is stored/fetched in/from the chips in encrypted form, which prevents disclosure to the middleware so that the data is not revealed if the middleware is compromised.

D. Business Logic Modules

Business logic modules are the core of each mobile application. They process application data, messages, etc., coordinate various other modules and functions, and provide specific application services to mobile users. The design and implementation of business logic modules depends on the functions that need to be provided by a mobile application. A good approach is to structure them in a modular form, so that they are individually as simple as possible. In other words, these business logic modules should utilize modules from other three groups as much as possible. The process of implementing business logic modules includes also integration with all the other three groups of modules.

E. Example application: SAFE Mobile Wallet

This section describes an example of the design and implementation of one secure mobile application using our GSM approach. The mobile application is called SAFE Mobile Wallet, which is one of the components of our Secure Applications for Financial Environments (SAFE) System [17], [18]. It was implemented by using Java Platform Micro Edition (Java ME) and supports various financial transactions, such as mobile banking, mobile payment Over-the-Counter (OTC) / Over-the-Air (OTA), mobile ticketing, mobile parking, etc. SAFE Mobile Wallet is continuously expanding with new functionalities.

The UI of SAFE Mobile Wallet comprises two levels of menus and several data handling forms. The hierarchy of UI panels is shown in Figure 3. Each panel is designed and developed using `MenuItemsSelection` template and then all UI objects are linked with each other by buttons or actions. One generic object, `UIProvider`, was implemented, which provides UI related functions, such as `createFormImage`, `addTextBox`, `addStringItem`, etc.

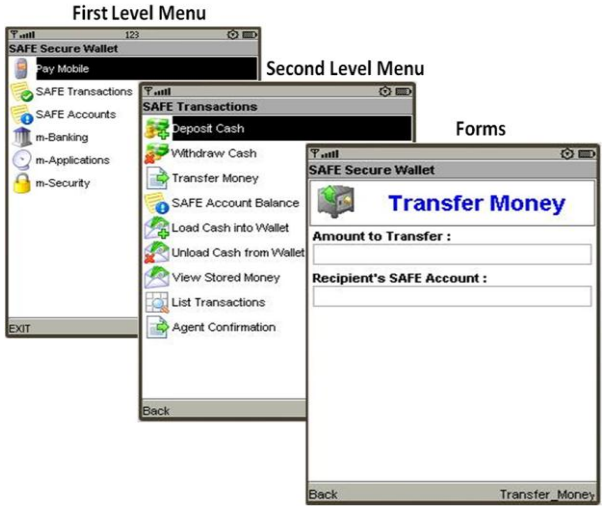

Figure 3. Hierarchy of UI Panels of SAFE Mobile Wallet

The SAFE Mobile Wallet uses three types of protocols to communicate with back-end servers: GPRS, Bluetooth, and SMS. Therefore, there are three corresponding generic communications objects: `BluetoothConnection`, `GPRSConnection`, and `SMSConnection`. Each of them handles one of the corresponding protocols. In addition, we implemented a generic `CommunicationProvider` object to manage these three communication objects. Each communication object returns its corresponding connection as an object to the `CommunicationProvider`. So, the business logic modules just need to invoke the `CommunicationProvider` to send out or receive messages through a specific communication protocol.

For communications with the UICC, SAFE Mobile Wallet uses UICC middleware in order to securely store and retrieve data from the UICC. This means that SAFE Mobile Wallet is able to invoke methods provided by the middleware internally without any separate communication method, as shown in Figure 4. In order to read and save data on the UICC, the Security and Trust Services API (SATSA) [19] was used.

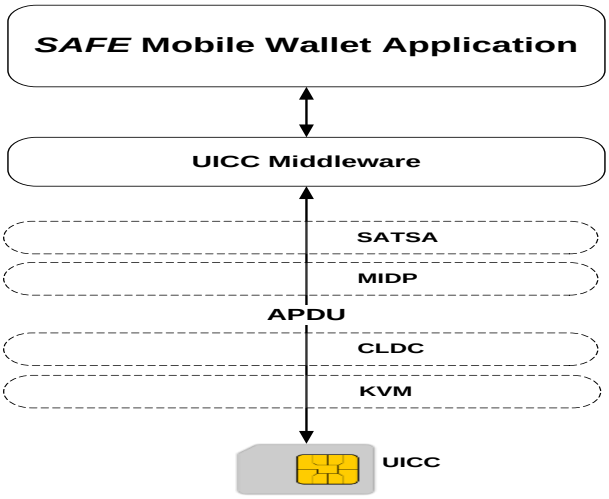

Figure 4. Communication between SAFE Mobile Wallet and UICC

The UICC middleware is a transparent, lightweight, secure and autonomous software module used by the SAFE Mobile Wallet. It was designed and implemented in such a way that the process of communicating with the UICC is transparent to the SAFE Mobile Wallet, which does not distinguish any difference between fetching and storing data from/to the smart card and manipulating data as files on its own system. Figure 5 shows an example of storing data on the UICC.

The SAFE Mobile Wallet provides several security services for its financial transactions, including user authentication, message confidentiality and integrity, non-repudiation, authorization, etc. [17]. For those services, `MobileSecurityProvider` module was designed and implemented. It provides various security functions, such as `generateRSAKeys`, `AESEncrypt/AESDecrypt`, `generateMessageDigest`, etc. All security services,

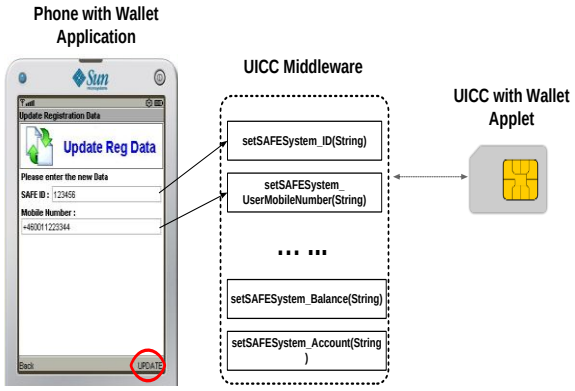

Figure 5. Example of the process for storing data on the UICC

provided by the Wallet, are based on these functions. Therefore, whenever an application requires new security features, a developer just needs to extend the `MobileSecurityProvider` module for new functions and cryptographic algorithms. Implementation of the `MobileSecurityProvider` object can be different, depending on the platform, programming language, APIs, etc. `MobileSecurityProvider` was implemented using Java ME, so it can be used on any mobile phone that supports Java.

The UICC middleware was also designed to function in a secure way. The middleware has no semantic knowledge of any data that it parses and passes in both directions. This gives mobile applications an option to store encrypted data in the UICC. Also, the middleware has no knowledge of passwords or keys used for authentication or cryptography. Another security feature is that the middleware keeps no data in its internal memory. All changes and modifications of data are saved directly in the UICC, which provides strong protection of data and also prevents synchronization issues (mostly when dealing with online, real-time transactions). Finally, the middleware follows the authentication mechanism required by the UICC: strong authentication protocol and secure channel between an application and UICC [20]. As a result, mobile users must authenticate themselves to the Wallet applet before being able to read and write data. If a user fails to provide the correct password after a predefined number of attempts, the UICC locks and can only be unlocked by the administrator.

As one of the components of our SAFE System, the SAFE Mobile Wallet is used to perform several types of mobile applications, such as mobile banking, mobile payment, mobile ticketing, etc. Therefore, business logic modules for those applications are designed as follows:

1. When a user starts SAFE Mobile Wallet, the Wallet asks the user to specify his/her PIN, if it is initial activation of the Secure Mobile Wallet. The PIN is then encrypted using AES algorithm, with hash value of the PIN as an encryption key, and is stored locally. Later, when a user starts the Wallet, the correct PIN is required in order to authenticate the user. After that, the Wallet initiates detection of available communication protocols. If a SAFE PoS Station supporting Bluetooth protocol is detected, the Wallet

 ISBN: 978-1-61208-209-7

will use Bluetooth to communicate with the Station and pass it through with other SAFE Servers. Otherwise, the Wallet tries to activate GPRS protocol. If GPRS is not supported, Wallet will use SMS as the communication channel. The described detection procedure is performed automatically and in the background;

2. After successful authentication, the user can perform all Wallet transactions [17]. All SAFE transactions are executed in a request-response way. The user enters the required data for a transaction. The Wallet creates a transaction request message and sends it to the SAFE Servers through an available communication protocol. SAFE Servers process the transaction and send the transaction result back as a response.

IV. CONCLUSION AND FUTURE WORK

This paper achieves several goals. First, it describes our proposed GSM methodology for the design and development of mobile applications. This methodology structures mobile application into four groups of modules and each of these modules is independently developed and reused, which makes the design and development of secure mobile applications more efficient, flexible and expandable. Second, it describes our proposed trusted layer model. This model provides a framework for designing secure mobile applications and more importantly, it represents a way of integrating the security mechanisms, provided by chips, with mobile applications. Finally, it introduces the approach of using a middleware for the communication between mobile applications and chips. This approach effectively protects the data, since the data is stored in the chips instead of mobile devices. It is much more difficult to compromise the chips than mobile applications.

Future research activities can be conducted in the following aspects: a. make the middleware more generic so that it is not dependent on upper layer application and lower layer chips; b. improve the trusty and security of middleware; c. extend the security modules to use biometric information, such as finger prints.

REFERENCES

[1] The App Store, There's an app for that. Over 500000, actually, Apple. In: http://www.apple.com/iphone/built-in-apps/app-store.html [Accessed March 16, 2012].

[2] P. Hornshaw: Mobile Apps Can Create Security Issues, Oct. 5 2010. In: http://www.appolicious.com/tech/articles/3388-mobile-apps-can-create-security-issues [Accessed March 16, 2012].

[3] iPhone Applications that transmit credentials using 'unsafe' protocols, Oxolab Blog, Apr. 26, 2010. In: http://blog.0x0lab.org/2010/04/unsafe-iphone-applications/ [Accessed March 16, 2012].

[4] J. Ueyama, V. Pinto and E. Madeira, "Exploiting a Generic Approcah for Constructing Mobile Device Applications", Proceedings of the Fourth International ICST Conference on COMmunication System software and middleware, Verona, Italy, July, 2011.

[5] F. Andre and M. Segarra, "A Generic Approach to Satisfy Adaptability Needs in Mobile Environments", Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Maui, Hawaii, Januare, 2000.

[6] M.A. Serhani, A. Benharref, R. Dssouli, and R. Mizouni: Toward an Efficient Framework for Designing, Developing, and Using Secure Mobile applications, International Journal of Human and Social Sciences (IJHSS), World Academy of Science Engineering and Technology, Volume 5 Number 4, pp. 271-278, 2010.

[7] Tips for Designing Mobile Applications, Clickbrand, January 12, 2009. In: http://www.clickbrand.com/blog/web-design/tips-for-designing-mobile-applications/ [Accessed March 16, 2012].

[8] C. Wysopal: The Challenges of Developing Secure Mobile Applications, Veracode, July 22, 2009. In: http://www.veracode.com/blog/2009/07/the-challenges-of-developing-secure-mobile-applications/ [Accessed March 16, 2012].

[9] G. Nudelman: Designing Mobile Search: Turning Limitations into Opportunities, UXmatters, March 8, 2010. In: http://www.uxmatters.com/mt/archives/2010/03/designing-mobile-search-turning-limitations-into-opportunities.php [Accessed March 16, 2012].

[10] "Bluetooth", Wikipedia [website], Available: http://en.wikipedia.org/wiki/Bluetooth [Accessed March 16, 2012].

[11] "Wi-Fi", Wikipedia [website], Available: http://en.wikipedia.org/wiki/Wi-fi [Accessed March 16, 2012].

[12] "3G", Wikipedia [website], Available: http://en.wikipedia.org/wiki/3G [Accessed March 16, 2012].

[13] "GPRS", Wikipedia [website], Available: http://en.wikipedia.org/wiki/Gprs [Accessed March 16, 2012].

[14] The Federal Information Processing Standards Publication Series of National Institute of Standards and Technology (NIST): Personal Identity Verification of Federated Employees and Contractors (FIPS 201), March 2006

[15] I. Kounelis, H. Zhao, S. Muftic, Secure Middleware for Mobile Phones and UICC Applications, Mobile Wireless Middleware, Operating Systems, and Applications, Springer, 2012, pp. 143-152

[16] M. Bishop, Computer Security: Art and Science. Boston, MA: Addison-Wesley, 2003, pp. 348-349

[17] F. Zhang, "Secure Applications for Financial Environments (SAFE) System", Licentiate Thesis, Royal Institute of Technology, Stockholm, Sweden, June 2010

[18] S. Muftic, F. Zhang, and K. DeZoysa, "SAFE System: Secure Applications for Financial Environments Using Mobile Phones", Proceedings of IADIS International Conference IADIS e-Society, February 2009.

[19] "SATSA Developer's Guide", Oracle [website], Available: http://download.oracle.com/javame/config/cldc/opt-pkgs/api/security/satsa-dg/index.html [Accessed March 16, 2012].

[20] CardContact, "OpenCard Framework", December 28, 2010, Available: http://www.openscdp.org/ocf/ [Accessed March 16, 2012].

 ISBN: 978-1-61208-209-7