SECURE MOBILE-CLOUD FRAMEWORK – IMPLEMENTATION ON THE MOBILE DEVICE

D. POPA[1] K. BOUDAOUD[2] M. BORDA[1]
[1]*Communications Department, Technical University of Cluj-Napoca, Romania*
Str. Dorobantilor, 71-73, Tel/Fax: +40(0)264401575, {Daniela.Popa, Monica.Borda}@com.utcluj.ro
[2]*I3S-CNRS Laboratory, University of Nice Sophia Antipolis, France*
930 Route des Colles - BP 145- 06903, Tel(Fax): +33(0)492965172(55), karima.boudaoud@unice.fr

Abstract: Secure Mobile-Cloud is a framework proposed to secure the data transmitted between the components of a mobile cloud application. In addition, the framework, takes into account the following aspects: 1) the users options regarding the security level required for private data and 2) the device energy consumption. The framework includes several distributed components. Some of these components are deployed on the mobile device and some of them in Cloud. This paper is focused on the implementation of the Secure Mobile-Cloud framework components on the mobile device. A proof of concept Android prototype is proposed.

Keywords: *Mobile Cloud Computing, Applications, Security*

I. INTRODUCTION

One of the greatest opportunities that every person wants to enjoy is mobility. Furthermore, each person has a small amount of curiosity, supplemented by a strong need for communication and knowledge. The mobile devices seem to be the devices that are able to link the mobility property with human emotional needs and information technology. All this is done using the Internet.

In order to capture people's attention towards mobile devices, powerful applications were developed for these devices. The applications allow mobile users to perform tasks like: managing personal health, games, editing, making reservations and paying tickets. As it is generally known, mobile devices are characterized by lack of resources. Thus, in order to run this new kind of applications, mobile hardware and network have known several improvements; but it wasn't enough. A solution to the mobile device challenges is Mobile Cloud Computing.

Mobile Cloud Computing (MCC) [1] is a new concept that can be described as the availability of Cloud Computing resources and services on the mobile device. This fact brings several advantages for the mobile devices (saving device energy, new storage place, additional computing power, etc.) and enables new powerful applications developed for them (e.g. a wide ranges of features) [2].

However, Mobile Cloud Computing increases the security risks and privacy invasion due to data outsourcing and synchronization via Internet. The security issues are various and fall into one of these three categories: mobile threats [3], Cloud threats [4] and threats at the communication channels level. Personal data (e.g. credit card numbers, passwords, contact database, calendar, location) is one of the main target of the hackers.

We are particularly interested in the security of data transmission, more specifically, the security of private data transmitted between the components of the same mobile cloud application. In our work, we focus on the security protocol adaptation according to end-user needs and mobile devices constraints. Furthermore, we assume that there is no need to apply the same security level (i.e. same security properties) for all data transmitted between the mobile cloud application's components.

We proposed a framework in [5] called Secure Mobile-Cloud (SMC). This framework has to secure the communication between the same mobile cloud application components. Also, it has to be able to adapt the security services according to the user needs and device (particularly the energy constraints). The framework includes two kinds of components: components deployed on the mobile device and components deployed in the Cloud.

In this paper we discuss in detail the implementation of the Secure Mobile-Cloud framework components on the mobile device side. In addition, we describe the design and implementation of the databases used for storing the user options. The user interface it is also presented.

This paper is organized as follows: section II describes the security framework. This section is divided in three parts: the first part is a short overview of the Secure Mobile-Cloud Framework described in more details in [5] and [6]; the second part presents the framework implementation on the mobile device; and in the last part are shown some unit tests. Section III presents the conclusions.

II. SECURITY FRAMEWORK

This section presents the Secure Mobile-Cloud framework design, short overview, and implementation.

A. Framework design – short overview

The Secure Mobile Cloud (SMC) framework is composed of two types of components, as presented in [6]: 1) security components and 2) management components. The security components have been designed in [7] for the LECCSAM architecture. The security components implement the eponym security properties: integrity, authenticity, confidentiality and non-repudiation. These security components are deployed in both, mobile device and in the Cloud. The management components have been designed to identify and apply the appropriate security properties to user's data. Some of these management

Manuscript received October 28, 2013; revised November 20, 2013

components are deployed on the mobile device and some of them are deployed in the Cloud.

The users are able to express their choices regarding the security level they want to apply to their data. In order to provide a solution that allows achieving this characteristic an analysis system was designed. This system is integrated into the security framework. Its function is to provide to the framework the security combination needed to be applied to data (security combination = security properties + security algorithms); and also the location where this combination can be performed (e.g. on the mobile or in Cloud).

The components of the SMC framework that are designed for the mobile device are presented in Figure 1. A short description for each component is given in Table I.

TABLE I. COMPONENTS DESCRIPTION

Component Name	Description
Mobile Security Manager	Manager, whose role is to ensure the composition of the security components on the mobile side.
Integrity	Security component, which applies the integrity property to data.
Authenticity	Security component, which applies the authenticity property to data.
Confidentiality	Security component, which applies the authenticity property to data.
Non-Repudiation	Security component, which applies the non-repudiation property to data.
Policy Manager	Manager, whose role is to determine which security components are required for a specific security level.
State Manager	Manager, whose role is to send the information regarding mobile device energy state to the Mobile Manager.
Mobile Manager	Manager, whose role is to collect data and events on the mobile device; it also includes the functionality of the analysis system.

B. Framework implementation

This section is divided in five subsections: 1) The Security Part, 2) The Auxiliary Part, 3) The Analysis System, 4) The Databases and 5) The User Interface.

The Security Part

This section presents the implementation of the Mobile Security Manager and the Security Components.

The class diagram is presented in Figure 2. The diagram depicts the connections between the various classes. The diagram consists of seven classes:

- *MobileSecurityManager class*: implements the functionality of the Mobile Security. It includes several methods, between which the most significant are the following two methods: apply*_combination_toEncript* and *apply_combination_toDencript*.

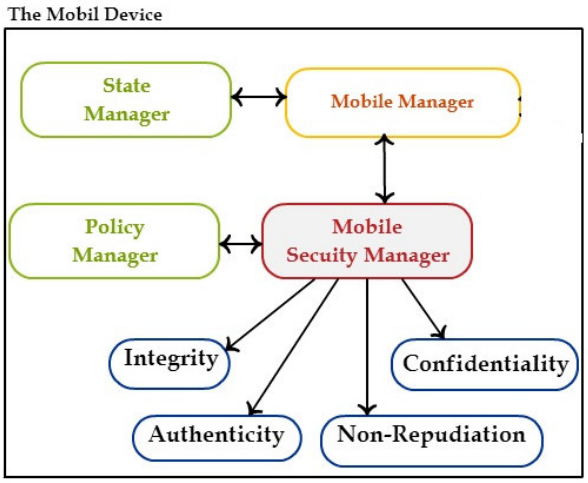

Figure 1. SMC framework – mobile device side

The *apply_combination_toEncript* method applies the appropriate security level to data in order to encrypt them (as the *apply_combination_toDencript* method is used when needed to decrypt the received data). The method has the following steps:

1) It finds that security properties combinations and corresponding algorithms for data security level provided. This is done by calling a method implemented by the Policy Manager; this method result returns a string with information.

2) It reads the string returned at the previous step; it sets the internal parameters (e.g Integrity, Authenticity) with the information read from the string.

3) It applies the corresponding security properties, by calling the appropriated methods.

- *Integrity, Authenticity, Confidentiality, NonRepudiation classes*: implement the security components functionality. Each of them comprises four methods, two private and two public methods. In the following there are described only the Integrity methods:

- *applyIntegrity* and *aIntegrity*: This two methods, the first one public and the second one private, are designed to provide the functionality of the Integrity component for plain text data. The public method calls the private method which uses a hash method implemented into the Operations class to perform the operation.

- *verifyIntegrity* and vIntegrity: This two methods, the first one public and the second one private, are designed to provide the functionality of the Integrity component for ciphered data.

- *Operations class*: implements methods that perform symmetric and asymmetric encryption and decryption and also the hash operation. These methods are as it follows:

- *encrypt*: which is symmetric or asymmetric; it takes as input an array of byte and a cipher; it performs a symmetric or an asymmetric encryption and returning an array of bytes.

- *decrypt*: which is symmetric or asymmetric; it takes as input an array of byte and a cipher; it performs a symmetric or an asymmetric decryption and returning an array of bytes.

- *hash:* it takes as input an array of byte and an hash algorithm performing an hash and returning an array of bytes.

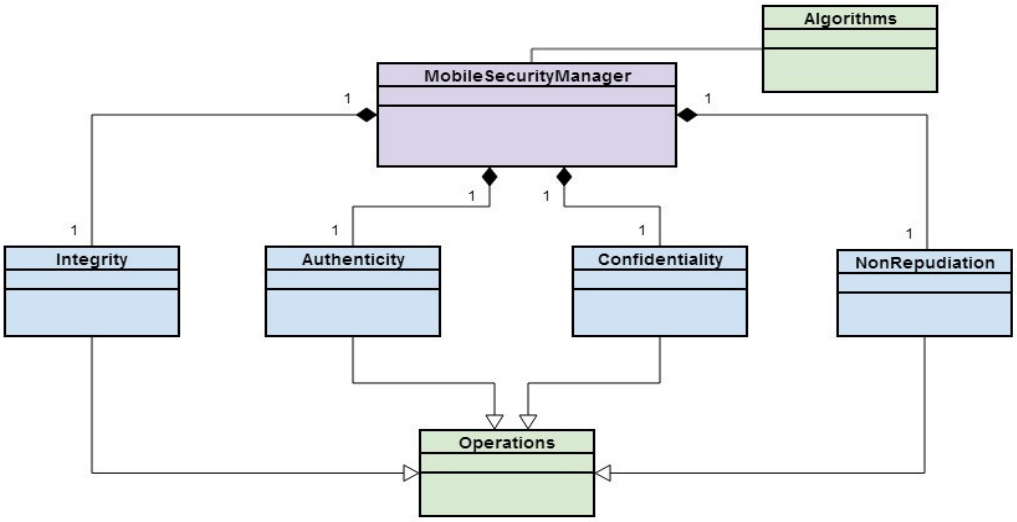

Figure 2. Security Part – Class diagram

The Auxiliary Part

In this section is presented the implementation of the Policy Manager and the State Manager. The implementation consists of two classes: PolicyManager and StatusManager as it can be seen in Figure 3.

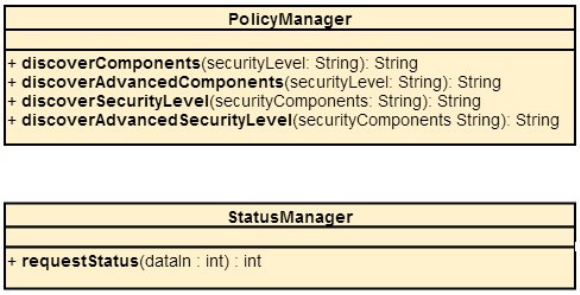

Figure 3. Auxiliary Part – Class diagram

The Policy Manager manages the security composition rules. These rules define the security properties (components) combination specific to a certain level of security.

Into the implementation there two types of encoding for a security level :

- *Basic code*: it defines the combinations of the security properties. Its form is as follow: C[n]; where C stands for combination and n it is a number (e.g. C7). This code is used when the user is of type standard or intermediary.
- *Advanced code*: it also defines the combinations of the security properties; but also includes the security algorithm chose by the user. Its form is as follow: AC[n]; where AC stands for advanced combination and n it is a number (e.g. AC33). This code is used when the user is of type advanced.

The PolicyManager includes four methods as it can be seen in Figure 3:

- *discoverComponents()*: receives a basic code of security level as input and returns a string with the corresponding security properties. This string contains the first letter of each security component name, if that component corresponds to the security level; the letters are separated by a colon. An example is presented in Figure 4.

Figure 4. Operating example for discoverComponent() method

- *discoverAdvancedComponents()*: receives an advanced code of security level as input and returns the string of corresponding security properties together with the corresponding algorithms. This string contains the first letter of each security component name, if that component corresponds to the security level, and a number which represent the security algorithm the user has choosen. All the information is separated by a colon. An example is presented in Figure 5.

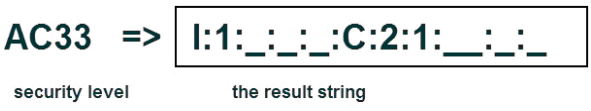

Figure 5. Operating example for discoverAdvancedComponent() method

- *discoverSecurityLevel()* and *discoverAdvancedSecurityLevel():* are the reverse operations of the methods presented above.

The StateManager class implements the functionality of the State Manager. On the current implemented version the StateManager class does not collect the energy level of the device. It contains a method that returns a certain number according to the value received as input.

The Analysis System

This section will present the implementation of the Analysis System. The Analysis System is integrated in the Mobile Manager. The class diagram of the Analysis System is shown in Figure 6; it can be seen here the connection between the various classes. The implementation for the Analysis System consists of two classes and one interface: MobileManager (the interface), MobileManagerParA and MobileManagerPartB (the classes).

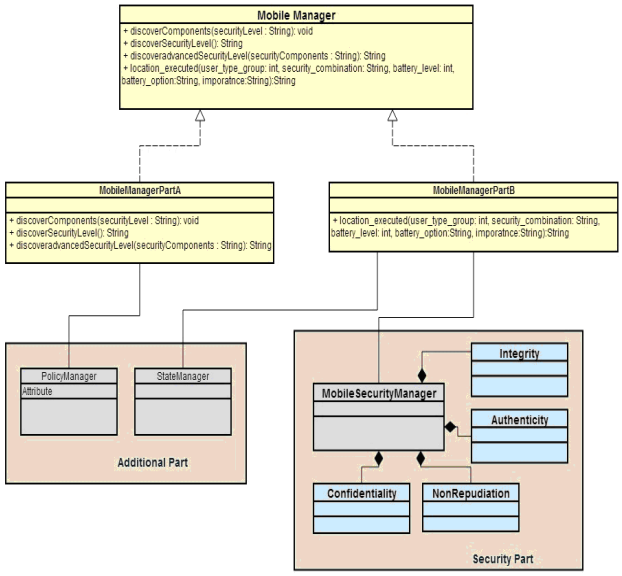

Figure 6. Analysis System - Class diagram

- *MobileManagerPartA class*: was implemented as part of the process that deals with the capture of users choices. It is the link between the user interface and the PolicyManager class. The methods implemented here are designed to use the PolicyManager methods in order to discover the adequate security level for a certain security properties combinations (or for the reverse operation).

- *MobileManagerPartB class*: was designed in order to implements the Analysis System The method that handles the Analysis System functionality is called location_executed(). This method receives as input the users constrains chosen through the user interface. It returns the security combination, the security algorithms and the execution location.

The Databases

The databases used by the framework are the following: Admin, Applications, and User.

The Admin database was designed to keep the default information needed by the security framework. The default information is data already predetermined; and refers to the general type of applications and the security properties combination encoding. This scheme contains three tables, described in Table II.

TABLE II. ADMIN DATABASE – TABLES DESCRIPTION

Table Name	Role Description
Types Table	It contains the types of applications.
Combination Encodes Table	It contains the security properties combination encoding.
Security Combinations Table	It contains the links between the two tables previously defined.

The Applications database was designed in order to keep the user options regarding the data security level of a certain mobile cloud application. This scheme contains four tables, described in Table III.

TABLE III. APPLICATIONS DATABASE – TABLE DESCRIPTION

Table Name	Role Description
Applications Table	It contains the list of the mobile cloud applications installed on the mobile device.
Applications Security Table	It contains the user options regarding the data security level for a certain mobile cloud application.
Applications Battery Table	It contains the user options regarding of to preserve or not the battery while a certain mobile cloud application is running.
Applications Priority Table	It contains data that specifies which of the two constraints: security or battery is more important for the user.

The User database was designed in order to keep the user options regarding to his level of knowledge in the security field. This database has only one table: “User Level Table”.

To store information into the database, it has been used the SQLite database in Android applications. SQLite is an Open Source database. SQLite supports standard relational database features like SQL syntax, transactions and prepared statements. The advantages for using SQLite are: 1) the database requires limited memory at runtime; 2) SQLite is embedded into every Android device; 3) it is not required a setup procedure or the database administration; it is only necessary to define the SQL statements for creating and updating the database.

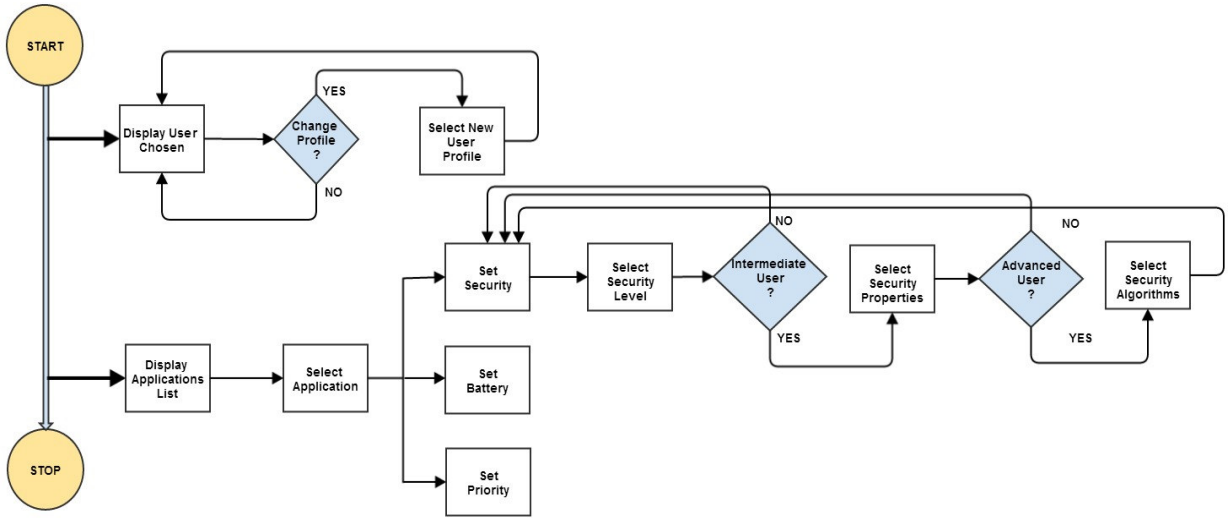

Figure 7. User Interface - Functionality diagram

The User Interface

The user interface was designed in order to capture the user option regarding the security level of his data and also regarding the device energy consumption. The user interface functionality is presented in Figure 7. Its functionality is divided in two phases: 1) setting the user profile and 2) setting the security.

The first phase allows the user to select the group to which it belongs according to his level of knowledge in the security field. As it can be seen in the Figure 8, there were defined three types of users: 1) Standard User Type, 2) Intermediary User Type and 3) Advanced User Type.

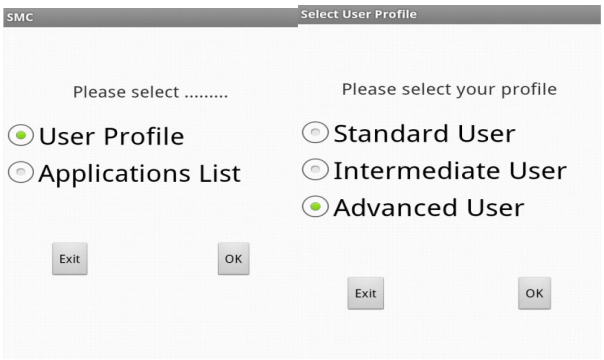

Figure 8. User Interface – The user type

The second phase, as its name suggest, allows the user to select data security level (Set Security step in Figure 7). In the case of the application users it comes to following question: How much flexibility a user shall have? Through flexibility it is understand the number of constrains (e.g. security level, security properties combination, security algorithms) the user can define. We decided for the flexibility to vary according to the user type. The Standard User Type is provided with the lowest flexibility and the Expert User Type has the greater flexibility. The lowest flexibility includes only the security level (e.g. strong, average). The grater flexibility includes, besides the security level, the security properties combinations and the security algorithms (see Figure 9). Also in this phase, all the users are allowed to chose if they want to save or not the mobile device energy (Set Battery step in Figure 7). In addition, all the users have to specify which of these two constraints: 1)security and 2)battery is more important for them (Set Priority in Figure 7).

Figure 9. User Interface – The security properties and algorithms

C. Unit tests

In this section there are presented a coupe of test. These tests target the security framework functionality.

The first scenario:

The user is of type advanced. He chooses the following options: 1) all data are secured equally regardless of the sensitivity level; 2) security level of type average; 3) as security properties he chooses only confidentiality; 4) as security algorithms he chooses: SHA(Secure Hash Algorithm), AES(Advanced Encryption Standard) and RSA; 5) he chooses to save battery; and 6) the priority is also the battery.

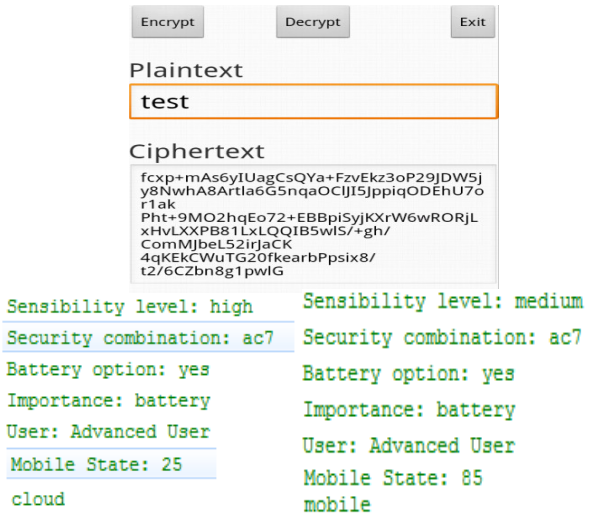

Figure 10. Results – Unit test first scenario

It can be seen in Figure 10 that, for data with different sensibility level (e.g. high and medium), there is the same security combination (e.g. ac7). Also, according to the mobile energy status (e.g. 25 or 85), one operation is executed on the mobile device (the result is also shown in Figure 10) and the other in Cloud.

The second scenario:
The user is of type standard. He chooses the following options: 1) all data are secured according to the sensitivity level; 2) security level of type average is chosen for low sensitivity data; 3) security level of type strong is chosen for medium and high sensitivity data; 4) he chooses not to save battery; and 5) the priority is the security.

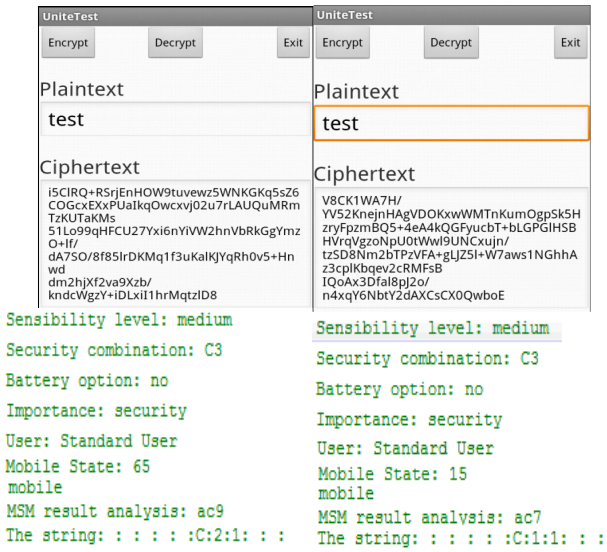

Figure 11. Results – Unit test second scenario

It can be seen in Figure 12 that, for data with different sensibility level there are different security combinations (e.g. C7[ac35] and C1[ac2]). The mobile energy status also can influence the security algorithm (Figure 11, security combination ac9 or ac7).

The security framework implementation on the mobile device was made using Java programming language and the Android [8] mobile platform. The programming environment used was Eclipse.

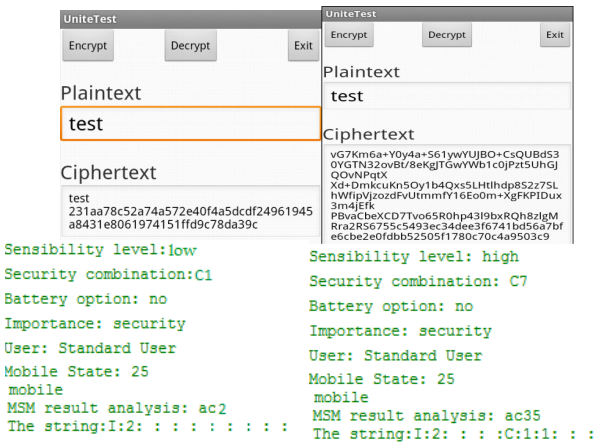

Figure 12. Results – Unit test second scenario

III. CONCLUSIONS

This paper describes the Secure Mobile-Cloud Framework implementation on the mobile device and the implementations details about the security components, the mobile security manager, the policy manager, the state manager and the analysis system. In order to allow the user to express his/her requirements regarding the security level to be applied to his/her data, a user interface was implemented. The information collected from users and the information from the analysis system are stored in a local SQLite database. Several unit tests were implemented in order to verify the security framework functionality. As future development we intend to integrate the proposed security framework into a mobile cloud application.

ACKNOWLEDGMENT

This paper was supported by the project: Improvement of the doctoral studies quality in engineering science for development of the knowledge based society-QDOC" contract no. POSDRU/107/1.5/S/78534, project co-funded by the European Social Fund through the Sectorial Operational Prog. HR 2007-2013.

REFERENCES

[1] S. Gautam Kumar, K. Dinesh, Mathew K. and Abhimanyu M.A. “Cloud Computing for Mobile World”.
[2] D. Kovachev, Y. Cao and R. Klamma, “Mobile Cloud Computing: A Comparison of application Models”, in eprint arXiv: 1107.4940, July 2011.
[3] Lookout Mobile Security, Lookout Mobile Threat Report, Aug. 2011.
[4] Cloud Security Alliance, Top Threats to Cloud Computing V 1.0, March 2010.
[5] D. Popa, K. Boudaoud, M. Cremene, M. Borda, “A Security Framework for Mobile Cloud Applications”, in Proceedings ROEduNet 11 th International Conference, Sinaia, 2013.
[6] D. Popa, K. Boudaoud, M. Cremene, M. Borda, “A System to Analyze the User’s Security Options for Mobile Cloud Applications”, The 6th International Conference on Security for Information Technology and Communications, June 25, 2013.
[7] M. Kamel, K. Boudaoud, S. Resondry and M. Riveill, Low-Energy Consuming and User-centric Security Management Architecture Adapted to Mobile Environments, in Proceedings of the 12th IFIP/IEEE, Dublin, Ireland, May, 2011.
[8] R Rodger, „Beginning Mobile Application Developement in the Cloud”, WROX Programer to Programer.