The Eighth International Conference on Business Information Security (BISEC-2016), 15th October 2016, Belgrade, Serbia

CryptoSMS ANDROID APPLICATION

JOVANA ĐUROVIĆ
University of Defence, Military Academy, Belgrade, jovanadjurovicloki@gmail.com

BOBAN MIHAILOV
Serbian Armed Forces, CKISIP, Belgrade, b.mihailov@ymail.com

IVAN TOT
University of Defence, Military Academy, Belgrade, totivan@gmail.com

IVANA OGNJANOVIĆ
Univerzitet Donja Gorica, ivana.ognjanovic@udg.edu.me

Abstract: *There are serious security risks that follow smartphones which includes Android cellular telephones. According to the claims of experts in the fields of mobile phones, it is known, for years already, that, from the aspect of security, the whole infrastructure of mobile phones is almost useless, and this is a fact that no one would refute. This is certainly exploited by, among others, those who wish to acquire certain data. Concerning the issue of which of the three following comunication channels is the most secure: a call, an SMS message, or communication via the Internet, the latter prevails. The potential of Android mobile platform protection achieved by CryptoSMS application, which was created as a response to the task of preventing security lapses, more promptly to prevent certain people, who can record mobile communication, from spying on SMS messages, is shown in this paper.*

Keywords: *Android, SMS, security, cryptographic algorithms.*

1. INTRODUCTION

Today, it is unimaginable to work, provide services, information of high quality and exchange information without modern information technologies. Up until recently, paper has been used to communicate and great deal of time has been spent filling forms and reports. Information technology plays a key role in everyday life of today’s society. It is used in every aspect of life.

In the last few years a huge increase of number of mobile phones in the population has been noticed. Because of that there is a need for cellular telephone application development so it would become more user friendly. *Android* operating system takes precedence in operating systems for mobile phones.

Android Inc which was sponsored by *Google*, and later in 2005. bought by it, created *Android* operating system. *Android* was presented for the first time in 2007. *Android* operating system is used for cellular telephones more than any other operating system. It is based on *Linux kernel* and it is usable for most mobile phones, tablet computers, *notebook* computers, *netbook* computers, *smartbook* computers, e-book readers, even for watches. *SQLite* database is used as data storage. *Android* supports connection technologies, such as *GSM/EDGE*, *CDMA*, *Bluetooth*, *Wi-Fi*, *LTE* and *NFC*. It also supports *SMS* and *MMS* and large number of languages. Additionally, it has modern *Web* browser, as well as different multimedia formats.

As well as every other mobile platform, *Android* mobile platform has some security flaws. Most of types of attacks on *Android OS* has already been seen in some form on different mobile operating systems. As the number of users grows so does the number of threats to *Android OS* and being the most used the number of threats is also the largest.

To develop its security, test application *CryptoSMS* that enables coded *SMS* exchange has been presented. This way secure flow of information has been provided.

2. SECURITY OF ANDROID OPERATING SYSTEM

Looking at scientific research program of mobile phones experts it has been known for years that the whole infrastructure of mobile phones is under great security risks and threats. This fact allows access to information to those who are not meant to see it [2] [6] [7].

If someone decides to encrypt or lock their data, they have to overcome certain logical obstacles at the start. For installation of encryption to be successful users at both ends need to install it. It is not enough for one user to encrypt their data. That way the data sent to the other user will be just a pile of unwanted data that can’t be decoded. [1].

Other experts suggest that users should pay attention to certain characteristics of programs that are offered to them. *Android* applications should be made in accordance with the principle of open code. In other words, source code should be visible to anyone who wants to see it. The best way for protection is to show how security is applied.

There are many free programs on the Internet with open source that make it difficult to listen in on or follow communications. To protect their smart phones, different programs are suggested for coding text and multimedia messages, for voice encryption in real time and for e-mail protection.

Users should ask themselves how much of their privacy are they ready to let go of. *G-mail* address is much more than regular e-mail address. At the same time user accepts that his mail goes through checking for marketing purposes.

Many programs, like *G-mail* and *Facebook*, look for user's approval to check their messages when they are sent via those programs. Applications for mobile phones with *Android OS* look for the same approval but it is made as a warning for users before they install a program. That is the moment when user can stop and think – why does the application look for access to user's phone book?

Vulnerabilities of *Android* devices are always assessed. Fact that *Android* devices are relatively new on the market and there are always new models and operating systems created makes it hard for experts to define all the vulnerabilities and risks [3].

Main concern of corporations is vulnerability of *Android*, because there is no proof of security and the risks are not fully known. That is the reason why corporations' smart phones are widely targeted, while their use is what hackers are mainly focused on [4].

3. IMPLEMENTATION OF THE CRYPTOSMS SECURITY SOFTWARE

SMS messages being sent from one mobile device to another are not protected, which means that there is a possibility that cyber criminals could intercept and access potentially sensitive data. The task is to develop an application which will guarantee a safe and secure data flow via *SMS*. The idea is to prevent illegal users from intercepting data traffic by implementing application layer security. In the event of data interception, hackers will not be able to utilize the actual data because it would be, as mentioned, previously protected.

Android mobile platform supports a multitude of cryptographic algorithms for use in data protection, both symetrical and asymetrical. Depending on users' needs, the most reliable algorithm will be used. Concerning symetrical algorithms, because the same key is being used on both the sending and the receiving side, the problem is ensuring a secure key interchange. On the other hand, asymetrical algorithms, which are usually by one order of magnitude slower than symetrical algorithms, are suitable for solving this problem (secure key interchange), because they utilise a public key which is available to everybody. If we ignore the process of key/keys interchange, the question of choosing betweeen these two types of criptograhic algorithms arises.

From the execution speed standpoint, symmetrical algorithms impose themselves as a logical choice (image 1). Considering the fact that the algorithm itself is executed on a mobile device, execution speed is an important factor in choosing the cryptographic algorithm to use.

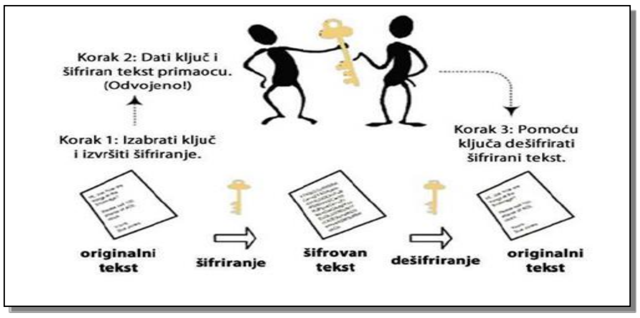

Korak 2: Dati ključ i šifriran tekst primaocu. (Odvojeno!)
Korak 1: Izabrati ključ i izvršiti šifriranje.
Korak 3: Pomoću ključa dešifrirati šifrirani tekst.
originalni tekst
šifriranje
šifrovan tekst
dešifriranje
originalni tekst

Image 1: Representation of encryption and decryption processes using a symetrical algorithm

Thanks to extensive research, conducted in the course of previous years, a symmetrical cryptographic algorithm known as *AES* proved to be the most secure. *AES* algorithm implements encryption and decryption operations on a block of data in a variable number of cycles. The number of cycles depends on key length and equals 10/12/14 for the 128/192/256 bit key, respectively. Prior to encryption or decryption, key expansion is performed. Block encryption systems are utilized in one of the so-called block cypher modes of operation. In terms of cryptography, a mode of operation implies a method of usage of a base algorithm and usually involves a combination of some sort of a loopback and specific basic operations. Operations performed on the algorithm are usually fundamental in nature because the aspect of security is determined by the base algorithm itself and not the mode of operation. In the *CryptoSMS* software, *CBC* (*Cipher Block Chaining*) mode is implemented [5].

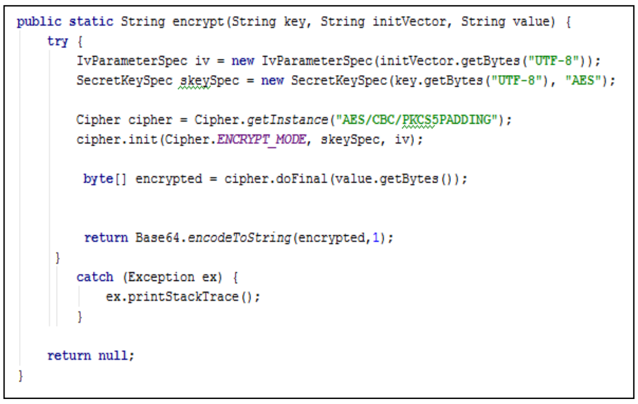
The application in question was developed for *Android* 5.0.1 operating system, using *Android Studio*, because the earlier versions of the *Android* operating system do not possess the necessary features for some newer technologies and the fact that the popularity of the newer versions of *Android* operating system greatly surpasses that of the previous versions.

In order for the application to be able to utilize the features and services of the mobile device it must be provided with the necessary permissions. By analyzing possible application usage and resources needed for proper function, *SEND_SMS*, *READ_SMS* and *RECEIVE_SMS* services were determined as necessary. These permissions were implemented in *AndroidManifest.xml* file, and the users themselves permit the usage of those services in the installation process.
While designing the application, the necessary classes and methods were imported into the project so connection to the service could be established

For the means of display optimization, components which take up relatively little space on the display and enable dynamic display expansion depending on the data influx were used. One of such components which was used extensively in the application is *ListView*. When using the aforementioned component, it is possible to set up the appearance of a single link, a *ListItem* and the data source, and the *ListView* control simulates the contacts list in the *CryptoSMS* application.

The database, implemented in the test application, enables the creation of a phonebook that is useful but not necessary for the application operation. Considering that the user, recipient and sender information needs to be saved because it is used in application operation, the process of basic data storage is enabled using a local database. Although *SQLite database* is relatively lightweight, in this instant it is an excellent option because such simple information does not require much space. In order for the *SQLite database* to be used, it is necessary to implement *Database Helper*. *Database Helper* is an auxiliary class used to manage the created database. It is an extension of *SQLite Open Helper* class which is used to establish a connection to the database if it exists, creates it if it doesn't, and upgrades it if necessary via *onCreate*, *onUpgrade* and *onOpen* methods.

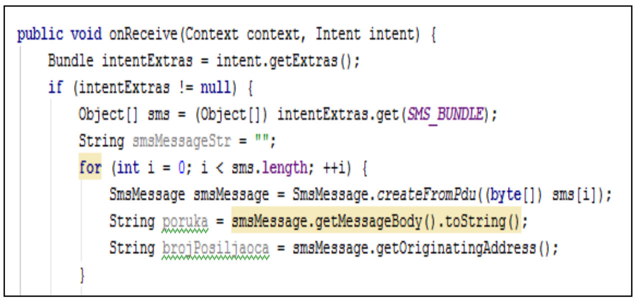
SMS Manager which controls operations such as data, text and *PDU SMS* message transmissions is used to send *SMS* messages. By tapping the Send button in the *CryptoSMS* application, the text is encrypted and the recipient receives an encrypted message which can be decrypted only if they possess this application. Encryption, as well as decryption, is conducted using *AES* symmetrical cryptographic algorithm in *CBC* mode (image 2). Using the *Toast* component, which provides simple feedback about the operation performed, any errors that can arise are processed.



```
public static String encrypt(String key, String initVector, String value) {
    try {
        IvParameterSpec iv = new IvParameterSpec(initVector.getBytes("UTF-8"));
        SecretKeySpec skeySpec = new SecretKeySpec(key.getBytes("UTF-8"), "AES");

        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec, iv);

         byte[] encrypted = cipher.doFinal(value.getBytes());


         return Base64.encodeToString(encrypted,1);
    }
        catch (Exception ex) {
            ex.printStackTrace();
        }

    return null;
}
```

Image 2: Representation of *SMS* encryption via *AES* cryptographic algorithm

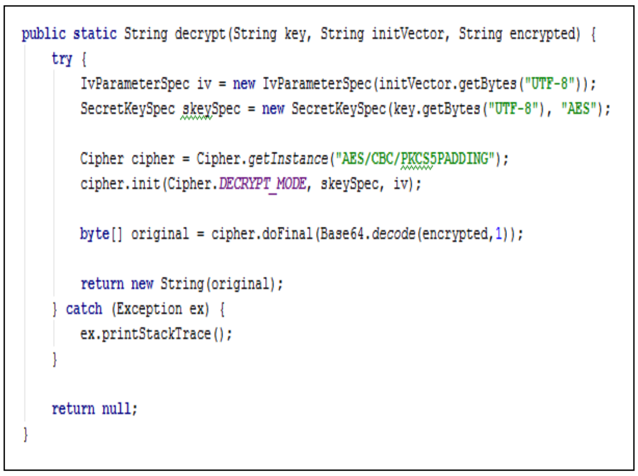
SmsBroadcastReciever class, as an extension of *BroadcastReciever class*, imports messages from the mobile device. When *SMS* is received by the mobile device, it is forwarded to *SmsBroadcastReciever* which receives the SMS in the form it was sent. This class does not perform decryption as that is conducted by *SimpleCrypto class*. *SmsBroadcastReciever* only receives the message and activates *SmsReader*. An example of *SMS* message extraction is given in image 3. *SmsReader* displays the *SMS* message after reception.



```
public void onReceive(Context context, Intent intent) {
    Bundle intentExtras = intent.getExtras();
    if (intentExtras != null) {
        Object[] sms = (Object[]) intentExtras.get(SMS_BUNDLE);
        String smsMessageStr = "";
        for (int i = 0; i < sms.length; ++i) {
            SmsMessage smsMessage = SmsMessage.createFromPdu((byte[]) sms[i]);
            String poruka = smsMessage.getMessageBody().toString();
            String brojPosiljaoca = smsMessage.getOriginatingAddress();
        }
```

Image 3: *SMS* message extraction

When the user receives the *SMS* message, after accessing the inbox of the application, layoutactivity_sms.xml enables the display of sender information and of the message itself in decrypted form. *ListView* component which, depending on the message length, makes changes dynamically, is also used to display this activity. Image 4 represents the *SMS* message decryption process via *AES*.



```
public static String decrypt(String key, String initVector, String encrypted) {
    try {
        IvParameterSpec iv = new IvParameterSpec(initVector.getBytes("UTF-8"));
        SecretKeySpec skeySpec = new SecretKeySpec(key.getBytes("UTF-8"), "AES");

        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
        cipher.init(Cipher.DECRYPT_MODE, skeySpec, iv);

        byte[] original = cipher.doFinal(Base64.decode(encrypted,1));

        return new String(original);
    } catch (Exception ex) {
        ex.printStackTrace();
    }

    return null;
}
```

Image 4: Representation of *SMS* decryption via *AES* cryptographic algorithm

4. USER INTERFACE

On application startup, a splash screen appears which serves as a feedback that the application has started, while in the background the environment is being initialized. After the splash screen, the *Main menu* page is displayed.

Main menu page prompts the user with three options:

- *Create an encrypted message*,
- *Phonebook* and
- *Received messages*.

Tapping on the *Phonebook* button, the contact list activity is displayed which enables the user to add a new contact by tapping the button *Add new*. If fields *Name* and *Phone number* are left blank, after the *Save* button is tapped, a notification informing the user that all fields need to be filled out is shown (image 5).

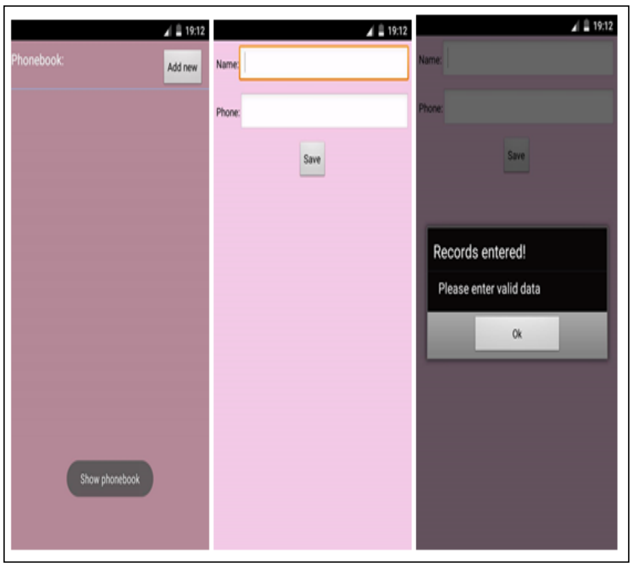


Image 5: Add new contact page

Deleting a contact from the phonebook is performed holding down the contact that is to be deleted (*ListItem* component) after which the user is prompted with the delete option (image 6).

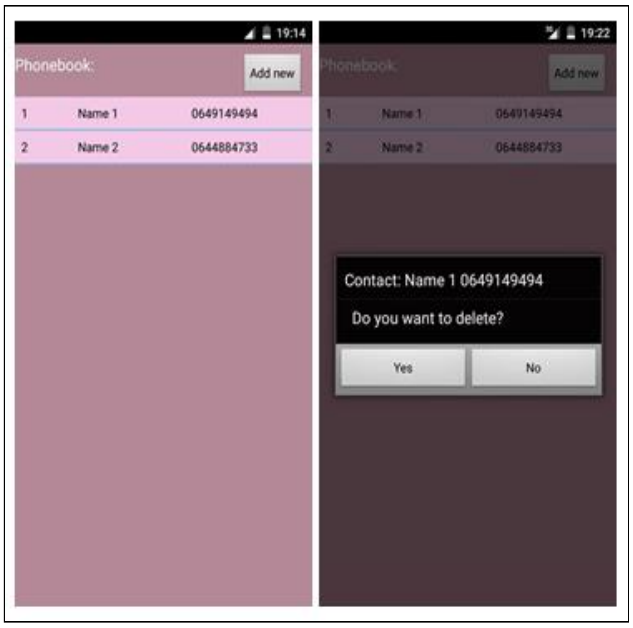


Image 6: Deleting a contact

Contact update (image 7) is done by tapping the contact in question. The updated contact is saved in the *SQLite database* and in further operation the contact will be stored in the phonebook.

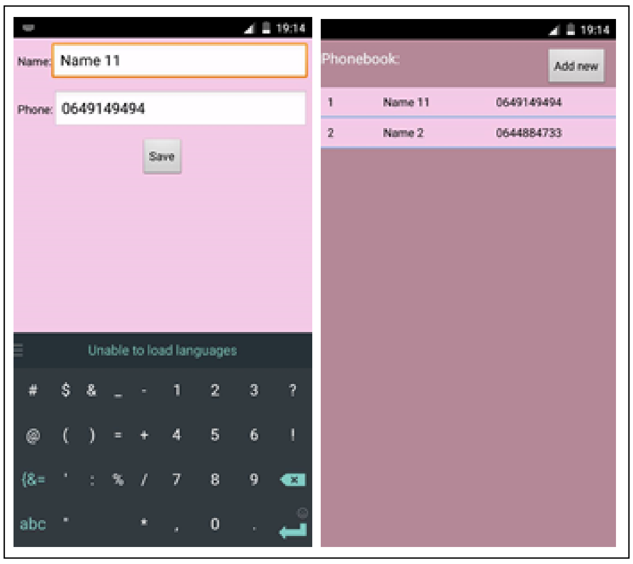


Image 7: Updating a contact

The *Create an encrypted message* option is used for creating a message and selecting a contact to which the message is to be sent (image 8). Test application provides an option not to select a contact from the phonebook, but to enter the recipient number manually if that recipient was not previously added to the phonebook. By tapping the "+" button, a list of contacts stored in the phonebook will be displayed using a *Spinner*. After contact selection the message transmission is enabled because the message recipient is defined. After defining the recipient, the message intended for that recipient can be entered. After tapping the text input control, the keyboard is displayed which greatly increases input speed and reduces the probability of error.

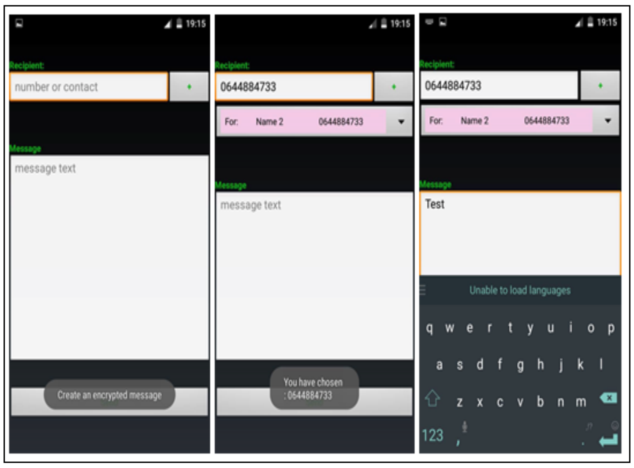


Image 8: Creating an encrypted message

SMS message is ready to be sent. Tapping the *Send* button multiple actions are performed:

- Encryption process is activated,
- Message is sent,
- Notifications about successful/unsuccessful operations during message transmission are displayed,
- If the *SMS* message is not delivered error processing is performed.

Assuming the message was sent successfully, the receiver would be able to open and read the message on their device. The message will be stored in the core messaging application on the device in encrypted form (image 9). This forces the illegal user who wishes to view received messages to decrypt the encrypted content. There is only one way of decrypting the content of such a message and that is via *CryptoSMS* application. Using the test application shown in this paper it is possible to view the actual content of the message.

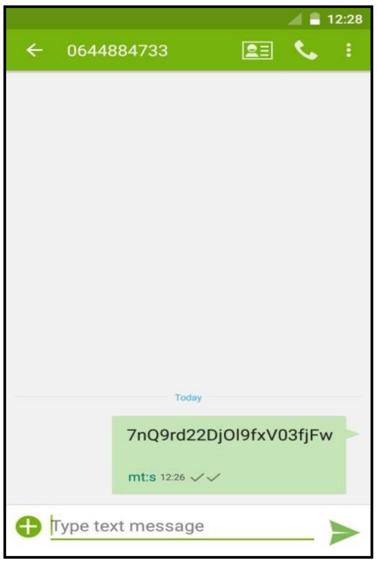
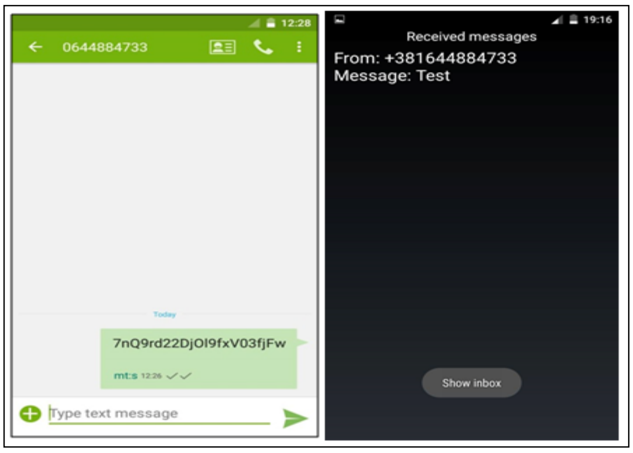


Image 9: Representation of the received encrypted message

When a message is received, thanks to the built-in *Toast* control, the sender and content of the message will be displayed. Accessing the application itself and its *Received messages* option reveals the sender and the decrypted message content (image 10).



From: +381644884733
Message: Test

Image 10: Received and decrypted message in *CryptoSMS* application

As an end result, the security of *Android* mobile platform is improved. Testing consisted of using this application on real devices in communication between two or more users. If the aforementioned application requirements are met, the user can be certain that the content sent/received via *SMS* is not exposed to risk because the protection is implemented on the application layer. It is worth mentioning that absolute security does not exist, but it is important to strive towards improving the existing solutions so every security risk can be resolved.

5. CONCLUSION

Security of the operating system is one of the biggest problems. With new discoveries come new problems that need to be solved. *Android* is a promising project. One of its main qualities is good organization that has a potencial to use all the power and knowledge of the open source community.

While there is technology development, maintaing the security of every component will be the main mission. Inovations are what makes people want to improve themselves. Flaws are made knowingly, because that way people are going to want to improve and to look for new undiscovered facts. Security and protection of mobile phones are two of the main problems of today's society.

CryptoSMS application has already exceeded expectations. Next development of the application will be in the field of security and in its possibilities of applying it in reality. Expanding the functions or even making new ones is the main idea.

REFERENCES

Books:

[1] Hoog A., "Android foresnics", (Vol. 1st Ed). Waltham, MA, USA, Syngress, 2011., ISBN: 978-1-59749-651-3

[2] Rogers D., "Mobile Security: A Guide for Users", Copper Horse Solutions Limited, 2013., ISBN 978-1-291-53309-5

Periodicals:

[3] Hoog A., "Introduction to Android foresnics", from DFI News, 2010.

[4] Folloder A., "Digital Forensics and File Carving on the Android Platform", Department of Computer Science, University of Texas at Dallas, 2012.

[5] Android Core Technologies, http://source.android.com/devices/tech/

Articles from Conference Proceedings (published):

[6] A. Chatzikonstantinou, C. Ntantogian, G. Karopoulos, C. Xenakis, "Evaluation of Cryptography Usage in Android Applications", 9th EAI International Conference on Bio-inspired Information and Communications Technologies, At, New York USA, 2015.

[7] A. Kandul, A. More, O. Davalbhakta, R. Artamwar, D. Kulkarni, "Steganography with Cryptography in Android", proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA), pp 57-64, 2014.