Data Security in M-Learning Messaging Services

Cătălin Boja, Paul Pocatilu, Alin Zamfiroiu

***Abstract*—** The m-learning field has been defined new educational services that deliver knowledge and content on mobile devices, highlighting a dreamed educational concept, anywhere, anytime. Education processes have evolved in the last decades by integrating new technology. Many m-learning services use SMS to deliver simple text content because it has a low cost, it is available on all mobile devices and has wide coverage and availability. As more and more sensitive data is processed by m-learning services, we propose a solution that will secure the SMS content and that will have minimal impact on the device performance. The solution implements a symmetric encryption scheme based on AES and is used by a distributed m-learning architecture. The paper presents two examples developed for Java ME and .NET CF platforms.

***Keywords*—**Cryptography, Mobile Applications, Security, SMS.

I. Introduction

ONE of the most used facilities offered by the mobile devices to communicate, with over 6.1 trillion messages in 2010 [1], is short written messages. For that, phone companies provide communication through Short Message Service (SMS). These forms of communication are widely used in adverse conditions such as those in which there is a lot of noise or verbal communication is not allowed. Communication via messages is preferred for privacy reasons. Conversation voice could be heard by others and for practical reasons and noise could hinder the conversation. Many people choose to use text messages that are cheaper than voice call.

Since the definition of digital natives concept by Marc Prensky, [13], [17], [21] and the analysis of the impact of technology on the youth daily activities, [5], research has been conducted on using this technology to increase the quality and the output of the educational process. A distinct approach is to use mobile devices to deliver educational content in anytime, anywhere scenario. Today, this scenario is something that can be achieved with small effort because:

- the degree of mobile devices penetration, mostly smartphones, describes a world in which each person uses at least one mobile device; the number of mobile phone subscriptions worldwide has reached 5.3 billion in 2010 and the mobile cellular penetration rates exceeds 100% in Europe and North America, [1]
- the majority of students use in daily activities mobile devices to socialize, search data on Internet, play, download multimedia content;
- the costs needed to acquire mobile technology decreased and also the lowest device specifications include data connections capabilities through Wi-Fi, Bluetooth and IR; it is estimated, [1] that until 2015 the Internet will be accessed by more mobile devices than desktop computers.

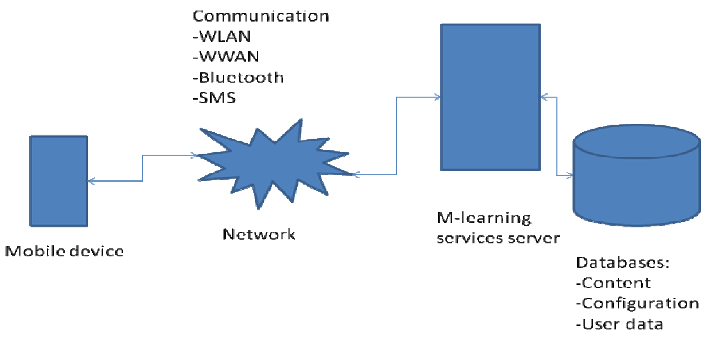


Fig. 1. The architecture of communicating through short encrypted messages

M-learning services are provided through standalone Java or Windows Mobile applications or in distributed environments, using different communication technologies like, Bluetooth in PicoNets, data connections for accessing Web applications or SMS [9] and [10]. Despite a large pool of communications technologies that provides mobile connections, m-learning solutions are restricted by the cost of using such technologies. The cost can be represented by the fees of using data connections, SMS services, Wi-Fi private networks, or by the availability of the service in a location, like a Wi-Fi or Bluetooth access point. Based on these pros and cons, the technology that is the cheapest and assures an anywhere coverage, hence assuring mobility, is the SMS service. These are the reasons for which, many mobile learning, social, banking and health services are based on SMS. Furthermore

Manuscript May 30, 2011: Revised version May 30, 2011.

This work was supported by CNCSIS –UEFISCSU, project number PNII – IDEI 2637/2008, project title: Project management methodologies for the development of mobile applications in the educational system.

C. Boja is with the Academy of Economic Studies, Economic Informatics Department, Bucharest, Romania (e-mail: catalin.boja@ie.ase.ro)

P. Pocatilu is with the Academy of Economic Studies, Economic Informatics Department, Bucharest, Romania (e-mail: ppaul@ase.ro).

A. Zamfiroiu is with the Academy of Economic Studies, Economic Informatics Department, Bucharest, Romania (e-mail: zamfiroiu@ici.ro)

the service cost supported by the user can be more reduced by collaborations between the mobile carrier and the mobile services authority.

II. Secure M-Learning Services based on Short Message Service

M-Learning services based on SMS communication deliver in real time different announcements, information, alerts, tasks. Other use of SMS is to define a mobile assessment architecture based on short quiz tests. Students receive questions and send the answer also in a SMS. SMS services can be used to provide a content delivery architecture that responds to requests send by text messages.

Security of sending SMS is becoming increasingly important when there is a risk of the interception of information or the data is sensitive like in a mobile banking service. The proposed SMSEncrypt solution solves the problem of message security through advanced and standardized symmetric encryption algorithms [22]. It uses symmetric encryption based on a secret key known by both parties. The disadvantage of this approach is in transmission of passwords between the two communication parties. The channel used to transmit the encryption/decryption key must be a secure one [20]. The Advanced Encryption Standard (AES) is also known under the name Rijndael. This is a standard algorithm for symmetric encryption, adopted as a standard, by the US National Institute of Standards and Technology (NIST) which defines data security procedures in governmental institutions, [22]. The algorithm, if it used correctly, has not been broken. It uses different size keys 128, 192, or 256 bits and the latter is still very difficult to break even with today technology. The AES (Advanced Encryption Standard) or Rijndael algorithm has these important characteristics:

- finalist and winner of the AES (Advanced Encryption Standard) contest launched by NIST in 1997
- created by two Belgians mathematicians: Joan Daemen and Vincent Rijman (Rijndael comes from their name)
- became cryptographic standard in 2000
- uses keys with 128, 192 or 256 bits
- is a symmetric cryptographic algorithm (the same key is used both for encryption and decryption)
- processes blocks of 128, 192 or 256 bits;
- effective both on Intel platforms and other software or hardware platforms
- it can be implemented on 32 bit processors and smart cards (8-bitprocessors);
- faster than DES;
- it is more secure than 3DES;

The proposed solution, SMSEncrypt, described in Fig. 2, is composed of two parts:

- the sender has a user key – UK used to encrypt the plaintext text message – M; in order to provide message integrity, the encrypted message contains the message hash;
- the receiver, which is another user or a service in the distributed m-learning architecture, uses the same key, UK, to decrypt the SMS; once the content is obtained, the receiver checks the message hash and responds if it is a service; because a symmetric key encryption solution is vulnerable in distributing the key to multiple users, the proposed m-learning architecture manages users and their associated keys;

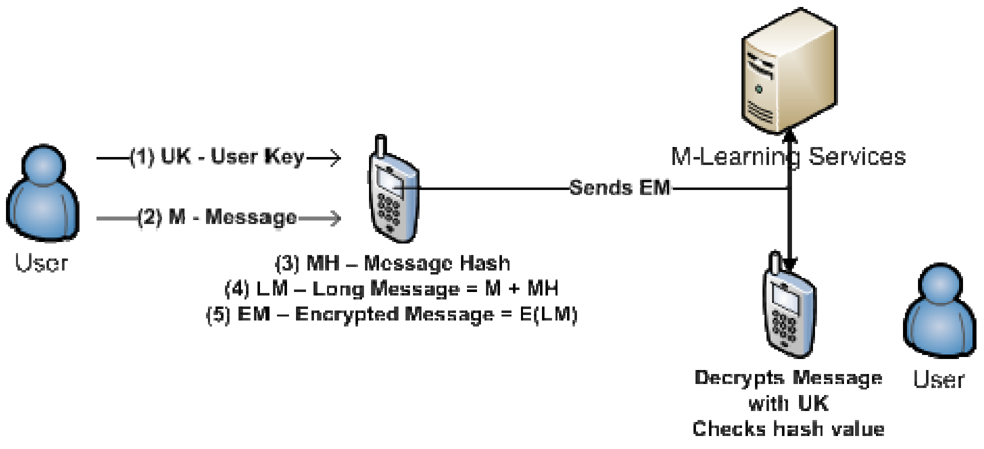


Fig. 2. The architecture of communicating through short encrypted messages

Encryption key and decryption key must be the same and should be known only by his message sender and receiver.

To send a message sender goes through the following steps:

- writes the message;
- specify the message encryption key; the key is uniquely associated with this user;
- specifies the recipient;
- the application generated a hash value based on the message text; the has is used to allow integrity check; the hash is added to the initial message and the content is encrypted;
- the encrypted message is sent over the network;

To receive a message and read it, the receiver goes through the following steps:

- the application or the M-Learning service intercepts the message;
- the sender is identified;
- the sender key is retrieved from a repository and is used to decrypt the cipher text;
- the integrity of the message is verified and the plaintext message is delivered.

These steps are shown in Fig. 3.

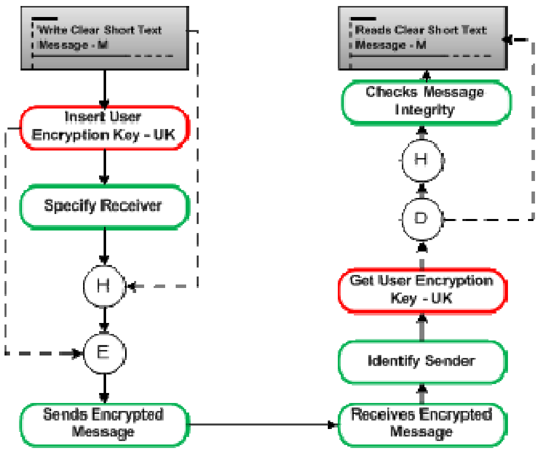

Write Clear Short Text Message - M
Insert User Encryption Key - UK
Specify Receiver
H
E
Sends Encrypted Message
Receives Encrypted Message
Identify Sender
Get User Encryption Key - UK
D
H
Checks Message Integrity
Reads Clear Short Text Message - M

Fig. 3. Stages of communication through encrypted messages

The quality of the m-learning service is directly influenced by the content quality and structure, and by the application quality. Regarding application quality there are defined multiple sets of quality characteristics [18]. The impact of a SMS based service on the mobile device performance and its power consumption is minimal, as described in figure 4 based on previous results, [23]. Also, the encryption stage has been analyzed and the results have shown that:

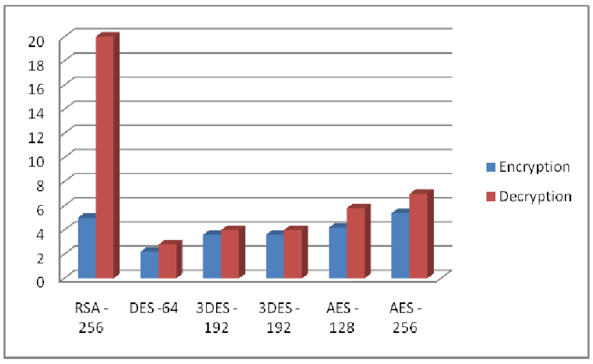


Fig. 4. Comparative results on encryption algorithms tested on mobile platforms, [23]

- generating a MD5 hash value is faster than a SHA-1;
- using a symmetric algorithm requires a less overhead than a asymmetric one, like RSA, that requires a public key – private key pair;

III. Technical Solution

In order to receive the encrypted message in a manner that allows the user to decrypt the message, the solution is based on message interception techniques:

- for .NET CF platform, the solution is implemented using the Microsoft.Practices.Mobile.PasswordAuthentication and System.Security.Cryptography classes.
- for Java ME platform, the same solution is using the Security and Trust Services API (SATSA) [8].

A. Java ME Implementation

In order to support cryptographic services, Java ME platform includes a package, the Security and Trust Services API (SATSA), [8], that is flexible enough to run with many types of cryptographic algorithms and protocols. The SATSA framework has been de-signed to run on any Java ME-based virtual machine, including the CDC and CLDC virtual machines. This Java standard specification has been defined by the Java Community Process (JCP) in JSR 177 [11].

The API provides interfaces that allow developers to implement secure solutions based on a smart card, the mobile device or a combination of the two. This survey concentrates only on the second solution, using only the mobile device processing unit, because there are other restrictions, legal and technical, that will not allow a smart card solution intended for a wide range of devices.

From all the SATSA packages, the one that does not require a smart card is the SATSA-CRYPTO package. It provides classes for implementing data security architectures based on message digests, digital signatures and symmetric and asymmetric encryption / decryption algorithms.

Classes fromSATSA packages like javax.crypto and javax.crypto.spec.SecretKeySpec provides the tools for symmetric encryption.

The **Bouncy Castle Crypto API** for Java provides a lightweight cryptographic API that works with everything from the J2ME to the JDK 1.6 platform.

The API is different for J2ME platform than the one for JDK platform. For example, the lightweight API has different implementations for the two platforms:

- *lcrypto-jdk16-145* for JDK platform;
- *lcrypto-j2me-145* for J2ME platform;

and also, at functions level, the situation in also different:

- for the JDK platform the main class for cryptographic algorithms is *Cipher;*
- for the Java ME platform there are classes for each cryptographic algorithm; e.g. for AES there are three: AESEngine, AESFastEngine and AESLightEngine.

For AES, the **Bouncy Castle Crypto API** for Java ME platform provides three implementations (their description is taken from the Bouncy Castle API documentation):

- AESEngine – "*The middle performance version uses only one 256 word table for each, for a total of 2Kbytes, adding 12 rotate operations per round to compute the values contained in the other tables from the contents of the first*" [19]
- AESFastEngine – "*The fastest uses 8Kbytes of static tables to precompute round calculations, 4 256 word tables for encryption and 4 for decryption*" [19]
- AESLightEngine – "*The slowest version uses no static tables at all and computes the values in each round*" [19]

In this example, we will use the first implementation – AESEngine that will process 128 bits (16 bytes) blocks and will use a 128 bit key.

Because, the files that are going to be encrypted may have or NOT (most of the times) a dimension that is multiple of block size (128 bits), we must use padding for the last block. In this case we will use the **PaddedBufferedBlockCipher** class, which is "*a wrapper class that allows block ciphers to be used to process data in a piecemeal fashion with padding*" [19].

The encryption solution (complete encryption/decryption solution file) is defined by these steps:

1. define the **PaddedBufferedBlockCipher** instance used for encryption – *encryptCipher* in this solution; the **PaddedBufferedBlockCipher** class provides two constructors; both require a **BlockCipher** instance; one use by default **PKCS7** padding (used by this solution), the other one requires a **BlockCipherPadding** instance; for the **BlockCipher** instance we will create a AESEngine object;

2. init the cipher for encryption with a key; the key could be predefined or received; these 2 first steps are implemented by the constructors:

```
public class AES_BC {

    PaddedBufferedBlockCipher encryptCipher;
    PaddedBufferedBlockCipher decryptCipher;

    // Buffer used to transport the bytes from
one stream to another
    byte[] buf = new byte[16];
//input buffer
    byte[] obuf = new byte[512];
//output buffer

    byte[] key = null;

    public AES_BC(){
  //predefined key value
        key =
"SECRET_1SECRET_2SECRET_3".getBytes();
        InitCiphers();
    }
    public AES_BC(byte[] keyBytes){
        key = new byte[keyBytes.length];
        System.arraycopy(keyBytes, 0 , key, 0,
keyBytes.length);
        InitCiphers();
    }

    private void InitCiphers(){
        encryptCipher = new
PaddedBufferedBlockCipher(new AESEngine());
        encryptCipher.init(true, new
KeyParameter(key));
        decryptCipher =  new
PaddedBufferedBlockCipher(new AESEngine());
        decryptCipher.init(false, new
KeyParameter(key));
}
```

3. read bytes from the file; in the solution, we read 16 bytes blocks from the file; each block is processed by the **int processBytes(byte[] in,int inOff, int len, byte[] out, int outOff)** function; the output of the processed block is put in the out buffer which is written in the encrypted file.

4. VERY IMPORTANT STEP call the **doFinal** function which will process the last block in the buffer; the internal mechanism of the algorithm implementation, based in its encryption mode (ECB, CBC, or other) keeps an internal buffer which must be also discarded into the output file (this is NOT the last block of the input file); the **doFinal** it is a MUST DO step.

The last two steps are implemented by the **encrypt** function**:**

```
public void encrypt(InputStream in,
OutputStream out)
    throws ShortBufferException,
IllegalBlockSizeException,
    BadPaddingException,DataLengthException,
    IllegalStateException,
InvalidCipherTextException
{
   try {
   // Bytes written to out will be encrypted
   // Read in the cleartext bytes from in
InputStream and
   //      write them encrypted to out
OutputStream

   int noBytesRead = 0;        //number of
bytes read from input
   int noBytesProcessed = 0;   //number of
bytes processed

   while ((noBytesRead = in.read(buf)) >= 0)
{

//System.out.println(noBytesRead +" bytes
read");

  noBytesProcessed =
encryptCipher.processBytes(buf, 0,
noBytesRead, obuf, 0);

//System.out.println(noBytesProcessed +" bytes
processed");
out.write(obuf, 0, noBytesProcessed);
}

//System.out.println(noBytesRead +" bytes
read");
              noBytesProcessed =
encryptCipher.doFinal(obuf, 0);


//System.out.println(noBytesProcessed +" bytes
processed");
 out.write(obuf, 0, noBytesProcessed);

 out.flush();
}
catch (java.io.IOException e) {

System.out.println(e.getMessage());
        }
}
```

The decryption solution is similar to the encryption one and is implemented by the **decrypt** function.

The complete solution is implemented by the **AES_BC** class in this complete encryption/decryption solution file.

In order to use these tow functions you must open the clear text file and the encrypted one. This is a sample for use of encryption from a running MIDlet:

```
void encryptFile(String fileName)
{
   try {
    FileConnection fci =

(FileConnection)Connector.open("file://localho
st/" + currDirName + fileName);
    if (!fci.exists()) {
       throw new IOException("File does not
exists");
           }
    //createFile("encrypt.txt", false);
    FileConnection fco =

(FileConnection)Connector.open("file://localho
st/" + currDirName + "encrypt.txt");

    if (!fco.exists())
       fco.create();

    if (!fco.exists()) {
       throw new IOException("Cannot create
encrypted file");
           }

    InputStream fis = fci.openInputStream();
    OutputStream fos = fco.openOutputStream();

    AES_BC encrypter = new AES_BC();

    // Encrypt
    encrypter.encrypt(fis, fci.fileSize(),
fos);
    fis.close();
    fos.close();

    Alert alert =
                new Alert("Confirmation","File
encryption terminated", null, AlertType.INFO);
    alert.setTimeout(Alert.FOREVER);

Display.getDisplay(this).setCurrent(alert);
         }
 catch (Exception e) {
  Alert alert =
                new Alert("Encryption error!",
Cannot access file " + fileName + " in
directory " + currDirName +
        "Exception: " + e.getMessage(), null,
AlertType.ERROR);
        alert.setTimeout(Alert.FOREVER);

Display.getDisplay(this).setCurrent(alert);
        }
}
```

B. .NET CF Implementation

.NET CF includes the System.Security.Cryptography namespace. .NET CF supports SHA1, TripleDES, DSA, RSA and other algorithms. Also, platform invocation of native CryptoAPI functions and third party libraries can be used.

For example, the class MD5CryptoServiceProvider is used for MD5 hash and the class and the class RSACryptoServiceProvider for RSA encryption.

.NET CF provides a namespace that include the MessageInterceptor class, used to intercept SMS messages based on user condition: phone number, sender name, text content etc. The message interception is managed through Outlook Mobile API. The OutlookSession class is associated to the OutlookMobile application.

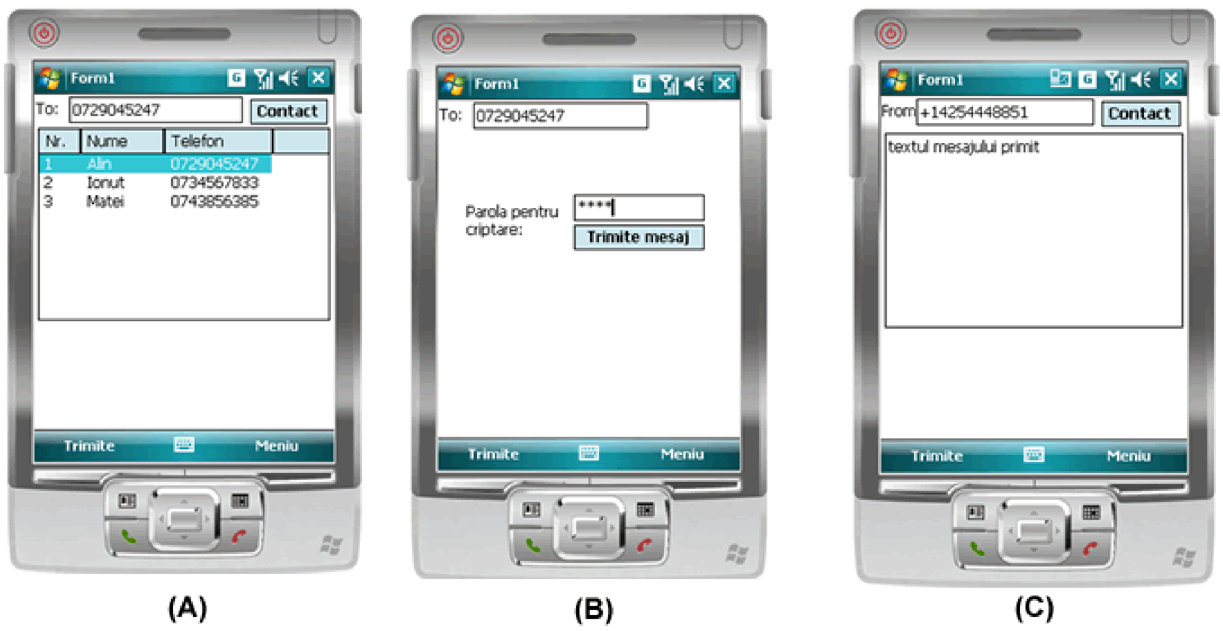
For example, the following code is used to intercept a SMS message and to define the function that will receive the message.

```
  using Microsoft.WindowsMobile.PocketOutlook;
  using
Microsoft.WindowsMobile.PocketOutlook.MessageI
nterception;

  OutlookSession outlookSession =
  new OutlookSession();
  MessageInterceptor mi =
  new MessageInterceptor();
  mi.InterceptionAction =
    InterceptionAction.Notify;
  mi.MessageCondition =
  new MessageCondition(
  MessageProperty.Sender,
  "+0711123456");

  mi.MessageReceived+=new
MessageInterceptorEventHandler(mi_MessageRecei
ved);
```

The function mi_MessageReceived will receive the message bases on criteria and the implementation is used to decrypt message:

```
  void     mi_MessageReceived(object     sender,
MessageInterceptorEventArgs e)
  {
    SmsMessage message =
      (SmsMessage)e.Message;

    //decryption of message goes here
  }
```

The same pattern is used for .NET CF applications, similar to Java ME applications.



To: 0729045247 Contact

Nr.	Nume	Telefon
1	Alin	0729045247
2	Ionut	0734567833
3	Matei	0743856385

To: 0729045247

Parola pentru criptare: **** Trimite mesaj

From +14254448851 Contact

textul mesajului primit

Fig. 4. Example of application interfaces in .NET CF

Application user interface is very simple and facilitate ease of use. The windows (Fig. 4) are built like applications to send messages to mobile phone.

After writing the message, press the "Send" menu and did not specify a phone number will appear select list of contacts for a phone number (Fig. 4A). This can be achieved and by pressing the "Contact". After is selected a contact which will be sent the message and the send button is pressed, it will prompt for the user (transmitter) key that will be used to encrypt the message (Fig. 4B).

At message reception it will prompt the user (receiver) key that will be used to decrypt the message. At the top appears the message sender's phone number so that the receiver knows the password will have to introduce. After entering the password to decrypt the message it will be decrypted and shown on the screen (Fig. 4C).

IV. DISCUSSIONS

For the analysis of mobile computing applications, these aspects are taken into account, to help determining the correct level of quality of the application source code:

- the number of lines of written code
- the number of commented lines
- the number of blank lines of code
- the number of private methods of a class
- the number of public methods of a class
- the number of IF instructions
- the number of ELSE instructions
- the number of Try-Catch instructions

If these parameters are named k1, k2, ..., kn, each one has a measured level for the tested application. These parameters will be used to achieve the indicators measuring the quality of source code through the following metrics:

GD (Level of documentation of source code) - represents the degree of the source code which contains enough commented lines to understand the code. In this way, a person having a first contact with the application in order to modify its code can easily understand it.

The calculation formula for this indicator is:

$$GD = (NLC / (NTLC-NLG)) * 100 \qquad (1)$$

where

NLC = number of commented lines
NTLC = total number of lines
NLG = number of blank lines

GL (degree of readability) - legibility influences the quality of the program source code, since the code can be changed in a later version of the software application. When someone who needs to change the source code does not understand it due to lack of readability, they are forced to rewrite it and thus losing valuable time. To enhance the readability of programs is recommended to:

- avoid writing more instruction on the same line
- add white space

The calculation formula for GL indicator is:

$$GL = NLG / NTLC * 100 \quad (2)$$

GTC (Level of management of cases) - this indicator has a major importance in the quality of the application as it indicates the degree of handling the exception cases. If the application programmer lefts untreated cases, the final user will encounter several errors, which will lead to a decrease in the quality of the application.

The formula for calculating this indicator is:

$$GTC = (NREL / NRIF) * 100 \quad (3)$$

where

NREL = number of ELSE instructions

NRIF = number of IF instructions

GTE (degree of exception handling) – this indicator is similar to the one presented above; it is important in the determination of the cases in which the developer hasn't considered all the possible exceptions that can occur while running the application.

The formula is:

$$GTE = (NRTRY / (NRMPr + NRMPu)) * 100 \quad (4)$$

where

NRTRY = number of try-catch instructions

NRMPr = number of private methods

NRMPu = number of public methods

The calculation is based on the following indicators of the efficiency metric, measured for each class of the application. The formula is:

$$QC_i = (GD_i + GL_i + GTE_i + GTC_i) / 4 \text{ for } i = 1,2\ldots n \quad (4)$$

where

N = number of classes in the application

QC_i = level of quality measured for i class

GD_i = level of GD indicator measured for i class

GL_i = level of GL indicator measured for i class

GTC_i = level of GTC indicator measured for i class

GTE_i = level of GTE indicator measured for i class

The formula for calculating the aggregate quality of the application is:

$$QA = \frac{\sum_{i=1}^{n} QC_i}{n} \quad (5)$$

This indicator determines a quality level for the entire source code of the intended application.

The measurements for SMSEncrypt application are presented in Table 1.

Table 1. Measurements made for SMSEncrypt

Parameter	Value for „ClassCriptare" class	Value for „CryptoTasks" class	Value for „Form1" class
NTLC	46	135	140
NLC	0	20	3
NLG	16	48	57
NRELSE	0	0	2
NRIF	0	0	2
NRTRY	0	0	3
NRMPr	0	0	6
NRMPu	2	9	2

The quality indicators of the application are calculated based on these measurements, Table 2.

Table 2 Quality Indicators for SMSEncrypt

Indicator	Value for „ClassCriptare" class	Value for „CryptoTasks" class	Value for „Form1" class
GD	0	22	3
GL	34	35	40
GTC	100	0	66
GTE	100	100	62,5
Total	**48**	**33**	**35,7**

The quality level of SMSEncrypt application in terms of source code is QA = 38.9.

V. Conclusion

Text encryption for sensitive data (e.g. personal information, passwords, and marks) is very important for distributed m-learning applications. Data optimization and processing speed are crucial factors to distributed m-learning applications success and these have to be taken into account.

Both Java ME and .NET CF provides mechanisms for data encryption and message interception.

Nevertheless the limitations of mobile devices are not taking away any of the benefits of m-learning. Mobile devices industry is progressing at very fast pace, current limitations will be overcome and m-learning applications will be enhanced even further.

The next step is to test several implementations and to extend the solution to other platforms like Android, iPhone and Windows Phone.

REFERENCES

[1] ITU, The world in 2010, ICT Facts and Figures, United Nations International Telecommunications Unit, http://www.itu.int/ITU-D/ict/index.html

[2] A. Wigley, D. Moth and P. Foot, Microsoft Mobile Development Handbook, Microsoft Press, 2007.

[3] J. Craig, AES Encryption in C#, Last accessed on March 2011, http://www.gutgames.com/post/AES-Encryption-in-C.aspx

[4] Code Project, 2006, Cryptor - Encrypt Files With Rijndael 256, last accessed on March 2011, Available at: http://www.codeproject.com/KB/security/Cryptor.aspx

[5] F. Obisat, E. Hattab, A Proposed Model for Individualized Learning through Mobile Technologies, International Journal of Computers and Communications, Issue 1, Volume 3, 2009, ISSN: 1998-4308, pp. 125-132

[6] P. Pocatilu and C. Boja, Quality characteristics and metrics related to m-learning process, Amfiteatru Economic, pp. 346-354, Vol. 11, no. 26, 2009

[7] C. E. Ortiz, The Security and Trust Services API for J2ME, Sun Developer Network (SDN), 2005, Available: developers.sun.com/mobility/apis/ articles/satsa1/

[8] Sun, SATSA Developer's Guide, SATSA Reference Implementation 1.0, December 2004, Available: http://java.sun.com/j2me/docs/satsa-dg/

[9] A. Muntean and N. Tomai, A Simple Web Platform Solution for M-Learning, Informatica Economica, vol. 14, no 1, 2010, pp. 172 – 181, ISSN 1453-1305

[10] P. Pocatilu, M. Doinea and C. Ciurea, Development of Distributed Mobile Learning Systems, Proc. of the 9th WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing (CSECS '10), Vouliagmeni, Athens, Greece, December 29-31, 2010

[11] Java Community Process, JSR-177, Security and Trust Services API for J2ME, Available at: http://jcp.org/en/jsr/detail?id=177

[12] A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997, http://www.cacr.math.uwaterloo.ca/hac/

[13] A. Bâra, A. Velicanu, I. Lungu and I. Botha, Using Geographic Information System for Wind Parks' Software Solutions, International Journal of Computers, Issue 2, Volume 5, 2011, pp. 149-156

[14] J. P Albuja. and E. V. Carrera, Trusted SMS Communication on Mobile Devices, In Proceedings of the 11th Brazilian Workshop on Real-Time and Embedded Systems, Recife - Brazil, pp. 165-170, May 2009.

[15] M. De Jode M. et al, Programming Java 2 Micro Edition on Symbian OS, A developer's guide to MIDP 2.0, John Wiley & Sons Ltd, 2004

[16] How Do I: Send and Receive SMS Messages? Available at: http://msdn.microsoft.com/ro-ro/netframework/bb905518

[17] M. Prensky, "Digital Natives, Digital Immigrants", in On the Horizon, MCB University Press, Vol. 9 No. 5, October 2001

[18] P. Pocatilu. and Boja C., "Quality Characteristics and Metrics related to M-Learning Process", Amfiteatru Economic, Nr.26, 2009, pp. 346-354

[19] The Bouncy Castle Crypto Package, Available: http://www.bouncycastle.org/documentation.html

[20] B. Schneier, Applied Cryptography: Protocols, Algorithms and Source code in C, 2nd Edition, Wiley, 1996

[21] B. Ghilic-Micu, M. Stoica and M. Mircea, A framework for measuring the impact of BI solution, Proc. of. rhe 9th WSEAS Int. Conf. on Mathematics and Computers in Business and Economics (MCBE'08), Published by WSEAS Press, Bucharest, Romania, pp. 68-73, 2008

[22] I. Ivan and C. Toma, Informatics Security Handbook, 2nd Edition, ASE Printing House, Bucharest, 2009, ISBN 978-606-505-246-8.

[23] J. P. Albuja and E. V. Carrera, Trusted SMS Communication on Mobile Devices, Proceedings of the 11th Brazilian Workshop on Real-Time and Embedded Systems, Recife - Brazil, pp. 165-170, May 2009

C. Boja is Assistant Professor at the Economic Informatics Department at the Academy of Economic Studies in Bucharest, Romania. In June 2004 he has graduated the Faculty of Cybernetics, Statistics and Economic Informatics at the Academy of Economic Studies in Bucharest. In March 2006 he has graduated the Informatics Project Management Master program organized by the Academy of Economic Studies of Bucharest.

He is a team member in various undergoing university research projects where he applied most of his project management knowledge. Also he has received a type D IPMA certification in project management from Romanian Project Management Association which is partner of the IPMA organization. He is the author and coauthor of more than 40 journal articles and scientific presentations at conferences. His work focuses on the analysis of data structures, assembler and high level programming languages. He is currently holding a PhD degree on software optimization and on improvement of software applications performance.

P. Pocatilu graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 1998. He achieved the PhD in Economics in 2003 with thesis on Software Testing Cost Assessment Models.

He has published as author and co-author over 45 articles in journals and over 40 articles on national and international conferences. He is author and co-author of 10 books, (Software Testing Costs, and Object Oriented Software Testing are two of them). He is associate professor in the Department of Economic Informatics of the Academy of Economic Studies, Bucharest. He teaches courses, seminars and laboratories on Mobile Devices Programming, Economic Informatics, Computer Programming and Project Management to graduate and postgraduate students. His current research areas are software testing, software quality, project management, and mobile application development.

Dr. Paul Pocatilu is member of ACM and IEEE, USA, from 2009 and of INFOREC Association, Romania from 1999.

A. Zamfiroiu has graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 2009. Currently he is in final year at Economic Informatics Master.