# Data Security Solution for Mobile Applications

Cătălin BOJA
Economic Informatics Department,
Academy of Economic Studies, Bucharest, Romania
catalin.boja@ie.ase.ro

*The paper describes security issues regarding accessing data from a mobile application. There are presented two scenarios regarding device stored information and data access over the network. The paper describes a security architecture based on Rijndael symmetric encryption algorithm that offers a high level of protection for device stored data. The security issues regarding data transfer over network are worked out using a solution based on RSA Asymmetric Algorithm. The architecture is implemented on a Windows Mobile platform using the Mobile Client Software Factory classes.*
***Keywords**: security, data, mobile applications, software*

## 1 Introduction

The rapid evolution of IT technology has allowed mobile devices to become more and more powerful, both as processing and data storing capacity. As a direct result, the software developers are offering solutions that resemble as functionality and characteristics to the desktop counterparts. This has allowed mobile users to access and use applications that are storing important personal and business data. The mobility characteristic of the devices makes it more vulnerable than a desktop computer which is locked in an office. As a fact, [7], each week in the US airports are lost over 12000 laptops and the number for small mobile devices worldwide is far greater. This requires more attention regarding the device sensitive data security.

## 2 Security Analysis

The analysis of the security level for a software application is an important phase in the development process. This characteristic is influenced by activities that take place in the analysis, development and maintenance stages. Hence, developers must define objectives in the early stages that take into consideration security measures to be implemented in the resulting software.

Because of this subject importance, past studies and researches, [8-9], have defined a process, which is integrated in the product development lifecycle, to be used to build and maintain a high level of security. This process, Security Development Lifecycle (SDL) it is used by Microsoft to develop software that needs to withstand malicious attack, [8].

The SDL phases are:

- Requirement; in this phase it is made a security analysis of the future product and there are identified and defined key security objectives; one of the phase objectives is to make a plan that will maximize the software security level, minimizing flaws or possible situations to attacks the application;
- Design; from a security point of view, the design phase will concentrate on a security architecture for the application; each component of this overall perspective on security design is analyzed in detail in order to determine its security risks and strengths; the components that have a high degree of vulnerability are included in the software attack zone; because components in this zone are more susceptible to attacks, there must be conducted threat modeling processes on them, in order to further analysis security issues;
- Implementation; during this phase the developer codes design specifications, tests and integrates them into software application; the phase of objective is to prevent insertion of security flaws or to remove them; in order to achieve it,

developers must apply coding and testing standards, test application with fuzzing tools, apply static-analysis code scanning tools and conduct code reviews.

- Verification; in this phase there are conducted more focused code reviews and security test on the functionally complete version of the application; further analysis of the results may affect the entire process;
- Release; developers must be assured that the application final version is ready, from a security point of view, to be delivered to end users; this is done by conducting a final security review that will provide an overall picture of the application security level and its protection against attacks; the revision must be conducted by development independent entities, internal or external, that will provide objective results;
- Response; because there is nearly impossible to develop software without vulnerabilities that is immune to attacks, developers must be prepared to take appropriate actions to reduce or repair the effects of new discovered security issues; based on known security reports, developers must update their best security practices and try to discover further vulnerabilities.

The Security Development Lifecycle process is defined around Microsoft SD$^3$+C, [8][10], principles:

- Secure by Design: the software analysis and design takes into consideration elements that will protect the application data and will provide a well defined level of security; the analysis must identify application sensitive zones and must provide solutions for them;
- Secure by Default: this principle take into consideration the fact that the resulting software will not reach a complete level of security; time and financial costs, low level of security knowledge at the software engineers level, deviations from initial plans and other factors have a great impact on the application security level and in the end these generate flows in the security architecture; in order to minimize their negative effects, software developers must provide multiple levels of protection, limit the application privileges and give access to necessary resources only when required;
- Secure in Deployment: developers must provide users with tools and knowledge that will help them to make own security reviews or to make use of the applications entire security capabilities; a common behavior according to this principle is to offer software updates that will repair known security issues;
- Communications: security issues must be openly discussed between developer, users and software community; this will help identify new threats or solutions.

As seen above, an important activity of the security development process is the security review of the code, especially the one in possible attack zones:

- unvalidated user input that could allow SQL Injection, buffer overflows;
- superficial or none user authentication ;
- high level of privileges for the application routines;
- unsecure data connections over the network;
- hard-coded sensitive and secret data;
- storing unencrypted sensitive data on the device, either in memory or external;
- use of deprecated API functions;
- low level of error handling routines.

The Security reviews process consists of activities that are conducted during the application development lifecycle. The results are influenced by many factors, as the developers' experience, applications characteristics, policies regarding software security implementation, used technologies. As a result, security reviews must be repeated at each stage and the results must have a high objectivity degree.

## 3 Data Threats

In [2] there are describes a series of known threats for desktop applications that are available also in the mobile area. The concepts apply to situations in which the

mobile device is using the network to communicate data or the attacker has physical access to the device.

**Brute force attack or DoS** on a mobile device that has been lost or stolen. If the attacker wants to guess the password or it tries to make a DoS attack on the application, the developer may include some delays or it might even lock out the application if too many attempts are made.

**Password guessing** because it is common or simple to deduce. Users tend to simplify the authentication process by providing less secure passwords like the common 1234 or a very short one like admin, birth date, a name. To counter this approach, the developer may include routines that are analyzing the strength of the user password forcing him to give a minimum number of alphanumerical characters.

**Access secret data sent over the network in clear text.** In order to communicate with the remote database or to access centralized information the mobile application must use network access. If the access requires authentication data like passwords and usernames, these must be sent over network in the message body. To access the Internet network mobile users may use wireless data communication, GPRS, 3G, provided by the mobile service provider or wireless hot spots. The first approach does not offers extra protection because it must be taken into consideration that the traffic is routed through a TCP/IP network between the mobile service provider WAP Gateway ant the remote server. In both cases, someone could filter the incoming transmissions and could read sensitive data if is sent as clear text inside the TCP packet. One solution to this problem is to use certificate based data transfer like SSL/TSL that will encrypt data sent over the network with the server certificate. The developers may use proprietary solutions that are also based on encryption.

**Access private or secret data stored on the device.** If the attacker has physical access to the device, he might try to read data that is stored locally in system registers, in memory or in local configuration files. The risk of losing the mobile device by theft or by mistake is a great concern and it has a greater probability of happing than the risk associated to a personal computer. The solution is to best hide the data and to encrypt it. This has a great impact on the application performance but reduces the risk of losing important data. The architecture described later in this paper offer an encryption solution based on the Rijndael algorithm. Another way of protecting data is to correlate the security levels of the mobile application with the ones of the operating system.

**Reverse engineering the application code** and retrieve passwords or other sensitive data. If the passwords are stored as unencrypted scripts in the source code it becomes available to anyone who reverse engineers the executable and looks in the source code. For .NET applications the situation is risky because the application source is a portable executable written in the MSIL language. The code obfuscator has no impact on securing this type of data because their role is to protect intellectual property by modifying MSIL so it becomes hard for someone using a decompiler to understand the code. Obfuscation does not change the value of constant values that are used to initialize variables as password or connection strings.

**The use of "proprietary" encryption algorithms** that aren't tested and are based on an idea, considered by the developers new and liable. A good encryption algorithm is made public because its strength comes from the intensive examination by the cryptography community, including hackers. This approach validates the algorithm and highlights its efficiency and unbreakability.

## 4 Techniques for storing sensitive data

The methods described in the previous chapter provide solutions for storing data but, as seen generates a lot of problems regarding the privacy and the security of the information.

If the user case or the application typology

requires that data must be stored inside the device or at a central point in the network, the developer must design and implement solution that will protect best possible the information that is sensitive for the user or developer:

- personal information with security risks like PINs, passwords, account numbers;
- application configuration data used to connect to remote central points like connection strings, client authentication accounts, encryption keys; this type of information is transparent to the user and is used only by the application automated processes, like the update routine , or support user activities.

The best technique for storing sensitive data that has the highest security level is not to store this information and to request it to users every time it is needed. This approach is not entirely efficient because a user will not want to memorize a 160 bit Rijndael encryption key. In practice, the solution is more user friendly and it represents only the entry point in the secure architecture. The user uses an account identified by a user name and a password which are more ease to use, even if it is required a 12 characters strong password.

Other solutions to this problem are based on:

- storing the secret data in active memory for short periods of time; once the data has been decrypted it is stored in clear format until is used; the developer must reduce to minimum the duration of time the data is stored in clear; despite the value is now in the device active memory someone may attach a process that scans it; best practice, suggests to manage the value by reference and not by value, moving it away from the program stack to the heap; also, after the operation has ended the memory can be rewritten with a bogus value to remove the secret;
- using registers to manage sensitive data; data is stored encrypted in the device OS registers; this approach adds a additional layer of protection if the device has supplementary access protection based on accounts or a general PIN (Personal Identification Number);
- using auxiliary devices in order to remove sensitive data from the device itself and to put it on a dedicated secure component like a security access card or a token;
- using configuration files to store encrypted sensitive data; the advantage of this solution is given by the possibilities to update the configuration file without affecting the application; also, this file can be send to the device from a central point of management; the drawback is given by the fact someone can get access with ease to the configuration file;
- getting data from the network; this removes the data out of the device but generates additional problems regarding storing data needed to communicate in the first phase with the central point;

## 5 Secure Architecture for Internal Data

The proposed architecture, described in figure 1, offer solutions to the problem of storing sensitive data as connection strings or passwords inside the mobile application. This solution, [1] has been implemented using Microsoft Mobile Application Blocks in the Mobile Client Software Factory (MCSF), [5]-[6].

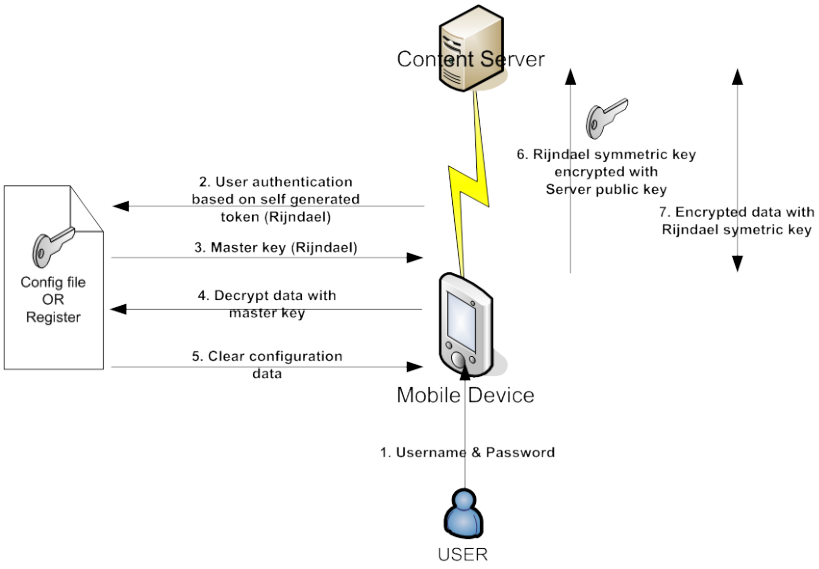


**Fig. 1.** Secure architecture for storing local sensitive data.

The scenario that is considered to explain the solution features has been the starting point for deploying distributed mobile application applications in which:

- there is a central data store that provides access to Web Services; to used them, mobile devices need to use data networks, and communication must be secured;
- there are multiple users with mobile devices; access to the device is secured and each one stores sensitive data in a configuration file; to prevent unwanted users to access the device, a user account is required and secret data is stored encrypted on the device;
- each user has its own authentication credentials and this can be managed from the central point; problems generated by forgotten passwords are managed only by the central point because it is the only one that can generate new encrypted configuration files;
- access to the central content server and to possible device data stores is made base on a secure login;

The secure components of the architecture are completing the security objective as they hides clear data from anyone who tries to get it without proper rights:

- the configuration file is a clear XML or ASCII file containing sensitive data as connection strings and user credentials; part of this file is encrypted using a Base64 encryption key and Rijndael AES symmetric algorithm; this key, called master key (MK), is generated by the central management point and it is delivered to all the mobile devices in the system; figure 5 describes the encrypted form of figure 4 configuration file;
- to protect the MK data, it is encrypted with another key, named user key (UK), also with the AES algorithm; the encrypted MK string is stored in the configuration file;
- the user key (UK) is generated from the username and password; these are data associated to each one of the users; this approach does not allow a user to change its account data because it must be done only by the central point; in the case of recreating or changing the password, the central point rewrites the encrypted configuration file and sends it to the device; figure 2 describes both the clear version of the XML configuration file and the encrypted one; each XML node contains the user name, the user key (UK) as a hash token and the encrypted MK string;

```
<?xml version="1.0" encoding="utf-8" ?>
<!--
//clear users configuration file
-->
<Users>
  <User Name="John"
        Pass="123456"
        MK="lCfUzlyL+0FEuZqT1zyC/w5OJUc0EQFWeFcQiShaRYw="/>
  <User Name="Andy"
        Pass="111111"
        MK="lCfUzlyL+0FEuZqT1zyC/w5OJUc0EQFWeFcQiShaRYw="/>
  <User Name="Joe"
        Pass="123123"
        MK="lCfUzlyL+0FEuZqT1zyC/w5OJUc0EQFWeFcQiShaRYw="/>
</Users>
```

**Fig. 2.** Unencrypted users credentials

- to authenticate users another protection layer is implemented by generating a hash token from the user name and password; this information is also included in the configuration file; at login, the application first authenticates the user by generating its token and comparing it with the one in the configuration file; after this point it will generate the UK and it will decrypt the MK and other sensitive data;

```
<?xml version="1.0" encoding="utf-8" ?>
<!--
//password for John is 123456
//password for Andy is 111111
//password for Joe is 123123
//initial MK is lCfUzlyL+0FEuZqT1zyC/w5OJUc0EQFWeFcQiShaRYw=
-->
<Users>
  <User Name="John"
        UK="AQASsvIMVMPNT3YWAjzSsv6P14dypn0jA/D5mQtH"
        EncMK="AQAQRkbPKXdCjFI8QqCxOn4d5qJs7tq3QM0dRaUR3oGyNYa8LX+/PiZwtZGm0E1oGT9hoGmYoHi7nguM4ZYchrcOjg=="/>
  <User Name="Andy"
        UK="AQBfr8ajiK7WqImQ0umFuhap3tZl+BUTR+faq8nQ"
        EncMK="AQAQH0Ec9XIt5Pt3ABt6NfHA6xcwsFlPcJ9l/gB/z2meKywPfMfBLiQAhfQyJ6TCARDDdDBmhZwrBEcX3QVYQI4V/Q=="/>
  <User Name="Joe"
        UK="AQAHfdDfNMrIriPiFCYhIZJkHg2lJ5ZA1zDXEbcf"
        EncMK="AQAQTOxczDiSAZ1vKxLLJ/4vdvbXuu29Pd0bGuYHS6uOa8yelgRhLpa14ftl6ssqdE8S+q2NPXQvjLv/fvWDArj9SA=="/>
</Users>
```

**Fig. 3.** Encrypted user key and master key

- to communicate with the content server, data transmitted over the network is encrypted by AES symmetric algorithm using a network key (NK); this key is not stored in the device because it can be received from the server or it can be generated and send it to the server; in order to communicate the NK key to both communication parties, the information is transmitted between user application and the server using RSA Asymmetric Algorithm, [3]-[4]; the common scenario is to generate at the device the NK key and send it to the server encrypted with the central point public key; after this synchronization, the communication data is encrypted using the NK key; this approach is more efficient than using RSA algorithm because the encryption with a symmetric is more rapid and less time consuming; if this approach has a negative impact on the application execution, it can be used another approach, based on the Secure Sockets Layer (SSL) protocol; the device can get the server certificate over the network before the communication starts.

The configuration file is managed using mobile Configuration Application Block from the MCSF. This framework provides classes and methods used to read and encrypt XML based configuration files.

```
<Endpoints>
<EndpointsItems>
<add Name="ClientsDatabase"
      Address="https://MyClientsServer/SecureServices/Clients.asmx"
      UserName="SalesAgent" Password="P@$$w0rd" />
</EndpointsItems>
</Endpoints>
```

**Fig. 4.** Mobile application configuration file

Considering the sensitive information from the figure 4 configuration file, it is used either *ConfigSectionEncrypt* tool from the Mobile Client Software Factory (MCSF) or a custom in-house developed application, to encrypt the data in order to store it on the device.

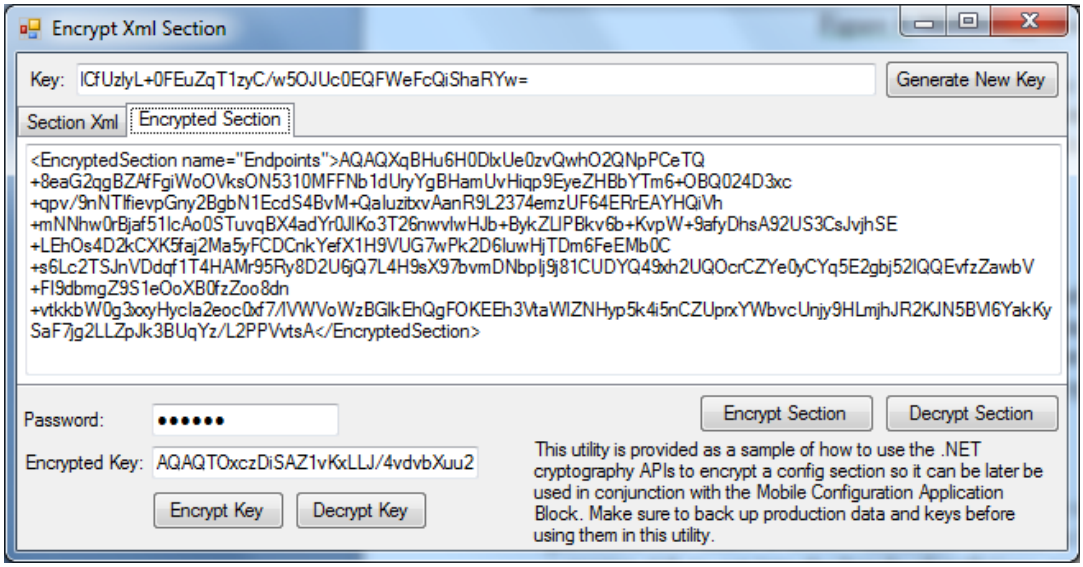


**Fig. 5.** Configuration file encrypted with the *ConfigSectionEncrypt* tool from the Mobile Client Software Factory

The presented solution has a high degree of security because in the end, only the user knows the key to the system. He starts the process by proving its own password. This information is not in the device. Also, the user does not know any other keys that provide access to the encrypted data. The information is self generated in a cascade type process. If is interrupted at some point, the access to clear data is denied.

## 6 Conclusions

Implementing good security is a must do for any software application. The mobile device and its particularities make it more vulnerable because it has the capacity to store sensitive data and has a greater risk to get into the hands of malicious users.

There are no 100% secure methods to protect data and the developer must ensure that has implemented procedures that will make data stealing a time consuming process that in the end will discourage anyone who tries it.

Good security starts in the early phases of the development process because it affects the solution and how it is implemented.

In the case of mobile applications, another characteristic that must be analyzed is the efficiency of implemented security methods as a time cost factor and its impact on the application behavior. The devices are still limited as hardware configuration and framework support.

## References

[1] A. Wigley, D. Moth, P. Foot *Microsoft*

*Mobile Development Handbook*, Microsoft Press, 2007, ISBN 978-0-7356-2358-3

[2] M. Howard, David LeBlanc – *Writing Secure Code*, Second Edition, Microsoft Press, 2003, ISBN 0-7356-1722-8

[3] C. Toma *Security in software distributed platforms*, ASE Publishing House, Bucharest, 2008, ISBN 978-606-505-125-6.

[4] V. V. Patriciu, I. Bica and M. Ene-Pietroseanu, O. Priescu *Semnatura electronica si securitatea informatica*, All Publishing House, Bucharest, 2005.

[5] Microsoft MSDN – Mobile Client Software Factory http://msdn.microsoft.com/en-gb/library/aa480471.aspx

[6] Microsoft patterns & practices – Smart Client Guidance, www.codeplex.com/smartclient

[7] *** Absolute Software – *Computer Theft & Recovery Statistics*, www.absolute.com/resources/computer-theft-statistics.asp

[8] S. Lipner "*The Trustworthy Computing Security Development Lifecycle,"* Annual Computer Security Applications Conference, ACSAC 2004, Tucson, Arizona, December 06-10, pp 2–13, ISSN: 1063-9527

[9] M. Howard and S. Lipner – *The Security Development Lifecycle*, Microsoft Press, Redmond, WA, 2006

[10] M. Howard *A Look Inside the Security Development Lifecycle at Microso*ft, MSDN Magazine November 2005, http://msdn.microsoft.com/en-us/magazine/cc163705.aspx

**Cătălin BOJA** is Lecturer 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 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.