A Windows Phone 7 Oriented Secure Architecture for Business Intelligence Mobile Applications

Silvia TRIF[1], Adrian VIŞOIU[2]
[1]Academy of Economic Studies, Bucharest, Romania,
[2]Economic Informatics Department, collaborator
silviatrif@gmail.com, adrian.visoiu@csie.ase.ro

This paper present and implement a Windows Phone 7 Oriented Secure Architecture for Business Intelligence Mobile Application. In the developing process is used a Windows Phone 7 application that interact with a WCF Web Service and a database. The types of Business Intelligence Mobile Applications are presented. The Windows mobile devices security and restrictions are presented. The namespaces and security algorithms used in .NET Compact Framework for assuring the application security are presented. The proposed architecture is showed underlying the flows between the application and the web service.
Keywords: *Security, Secure Architecture, Mobile Applications, Business Intelligence, Web Service*

1 Business Intelligence Mobile applications

The new achievements in mobile device technologies opened the way for new applications designed to run on mobile devices. In the beginning, mobile devices offered very limited functionality due to small memory, computing power and difficult interaction. Nowadays, mobile devices become more and more popular, available memory grew considerably, being comparable with some desktop computers, mobile processors have improved performance, interaction is becoming more user friendly. These characteristics allow the development of complex applications that make use of available hardware capabilities.

Business Intelligence applications help managers to make decisions based on quantitative methods applied to available business data. Mobile business intelligence applications extend such functionality on devices used by the decision makers [1].

Such applications take several forms:

– standalone applications; these run entirely on the mobile device and do not depend on an external entity to realize their functionality; such applications offer independence to the user as they do not need network access; a disadvantage is represented by the difficulty to feed data to the application and the lack of processing power compared to a PC or an application server; standalone applications are recommended for solving small size problems or obtain a skeleton for solving bigger problems;
– network applications; these applications have distributed components; some components reside on the mobile device and other components reside on external systems; a network application may obtain the data to be processed from a server or a web service and may store the results on the external system; also, data and results may be obtained and stored from the device itself as a standalone application; the problems solved are of medium complexity;
– web applications; these applications only render the interface on the mobile device; all the application logic takes place on external application servers; these applications are totally dependent on the network access but the complexity of the solved problem is high as all the processing is done on powerful machines.

When using business intelligence mobile applications the user accesses sensitive data from the inside of its organization. Also, the obtained results under the form of reports are sensitive information that must be protected. In this scenario, security becomes an important aspect that has to be

considered.Each type of mobile application has particular characteristics and security is implemented in specific ways.

2 Windows Mobile devices security and restrictions

There are numerous mobile platforms, each of them having specific characteristics, specific functionality and specific API that may be used to develop secured applications. Of these, the windows phone 7 platform is chosen to develop secured Business Intelligence Mobile Applications. An inventory is performed to discover the elements that contribute to secured application development.

In the mobile applications development process are taken into account the following concepts: sandbox, isolated storage, the cryptographic elements and the permissions.

The sandboxing model for applications on the phone means that third party applications are not allowed to run in the background, applications can only access their own isolated storage and they cannot directly interact with user data and phone functionality.

Isolated Storage provides safe client-side storage for partial trust applications; it enables managed applications to create and maintain local storage. Isolated storage is a space assigned to every application where this can read or write files. Other applications do not have access to another isolated storage than its own[2].

For developers there is special API in the .NET Compact Framework to address security elements, found in the namespaces [3]:

- System.Security.Cryptography provides cryptographic services related to encrypting and decrypting of data and related operations;
- System.Security.Permisions defines classes that control access to operations and resources based on policy;
- System.Security.Principal defines a principal object that represents the security context under which code is running, related to role-based security.

A number of encryption algorithms are implemented [4]:

- AES - represents the abstract base class from which all implementations of the Advanced Encryption Standard (AES) must inherit;
- AesManaged - provides a managed implementation of the Advanced Encryption Standard (AES) symmetric algorithm;
- CryptoStream - defines a stream that links data streams to cryptographic transformations;
- Hash Algorithm - represents the base class from which all implementations of cryptographic hash algorithms must derive;
- HMAC - represents the abstract class from which all implementations of Hash-based Message Authentication Code (HMAC) must derive;
- HMACSHA1,SHA256 - computes a HMAC using the SHA1and SHA256 hash functions;
- Rfc2898DeriveBytes - implements password-based key derivation functionality, PBKDF2, by using a pseudo-random number generator based on HMACSHA1;
- SHA1, 256 - computes the SHA1 and SHA256hash for the input data;
- SHA1, 256 Managed - computes the SHA1 and SHA256hash for the input data using the managed library;
- Symmetric Algorithm - represents the abstract base class from which all implementations of symmetric algorithms must inherit.

Secure communication with external servers uses SSL but this approach requires that a certificate issued by a recognized authority is installed on the device. This implies additional costs for achieving security.

It is observed that Windows Phone 7 platform offers enough elements to implement complex secured mobile applications.

3 Secure architecture for Business Intelligence Mobile Applications

A Business Intelligence Mobile Application is considered. The following hypotheses about the application are made:

- only authenticated users may access the application;
- the application offers the possibility to download external data from a server inside the organization via web services, also data can be uploaded to the server;
- the application saves final and intermediary results as well as reports on the device and also can upload them to the external server;
- the application performs various processing on primary data.

Taking into account the hypotheses made, a security architecture is built. The proposed architecture must comply withthe functionality of the application and the restrictions given by the platform where it is implemented.

The deployment of the proposed application architecture is given in Figure 1.

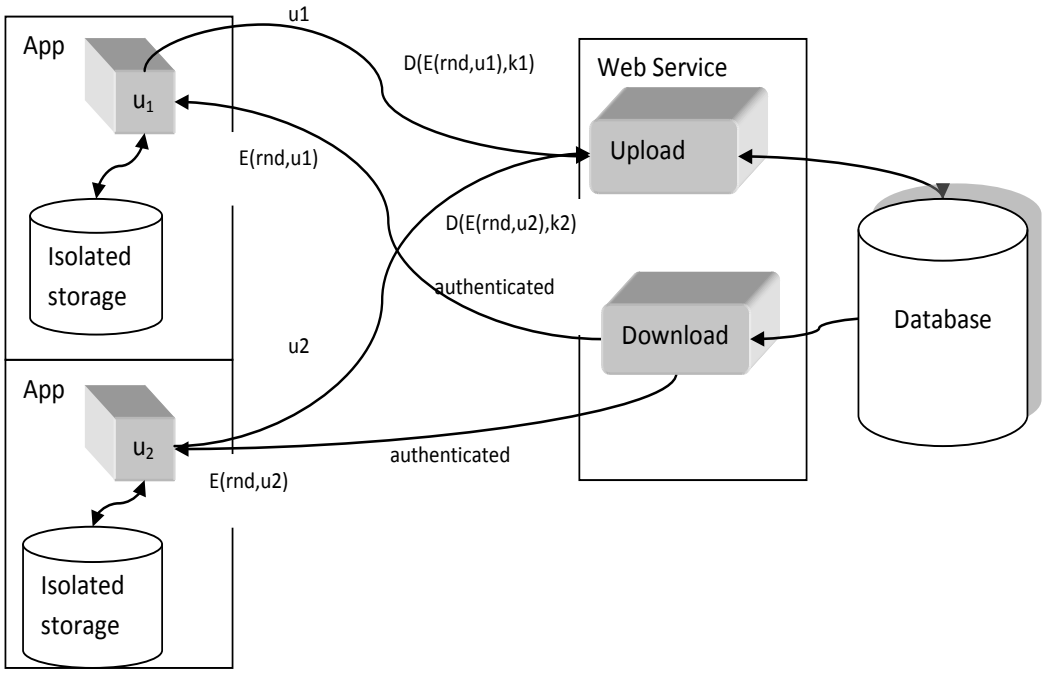

Fig. 1. The deployment of the proposed application architecture

It is observed that from the application development point of view, the key points where security is involved are:

- access to application; an authentication method must be implemented to grant access only for a specific users;
- storage of data; data must be stored encrypted in order to avoid access from other than authorized users;
- data transfer via the network; network traffic can be captured but an eventual intruder must not be able to understand the exchanged content.

The proposed architecture has the following elements:

- a solution with symmetric key is taken into account;
- a central authority within the organization issues a password *Pu* for each user *u*; the authority securely stores the passwordand also the user has the responsibility of keeping it secret;
- the issued password is used by the user to access the application;
- inside the application, the password is used to derive the key *Ku* used to encrypt and decrypt data on the Isolated storage;
- when communicating with the web services, the key *Ku* is used first to authenticate the user to the web service in

an authentication dialog and later, after the user is authenticated, to exchange encrypted data with the web service.

The authentication dialog consists of the following steps:

- the application connects to the web service sending the username;
- the web service responds with a random string encrypted with the key corresponding to the received username;
- the application uses the key to decrypt the random string and sends it back to the web service;
- the web service compares the received string with the generated random string; if they are equal the user is authenticated;
- after authentication the application can access the interface of the web service.

In Figure 2 is presented the authentication dialog between the application and the web service.

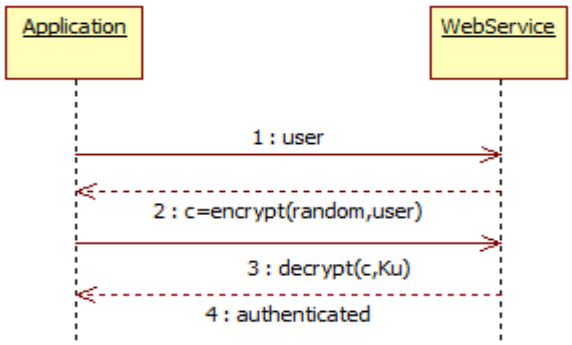

Fig. 2. The authentication dialog between the application and the web service

The transferred data between the application and the web service is saved on device, in the Isolated Storage that corresponds to the application. After the authentication process ends with success, the data is saved on the Isolated Storage. The process of saving the data into the Isolated Storage consists on the following steps:

- the application send a request to the web service to get the data that is needed;
- the web service verify if the user is authenticated and if it is authenticated, the service responds with the encrypted data, else it responds with a message;
- in the case of successful authentication, the application access the Isolated Storage and sends the encrypted data to be saved;
- the Isolated Storage send back a message that the saving process has ended successfully.

In Figure 3 is presented the process of saving data to the Isolated Storage.

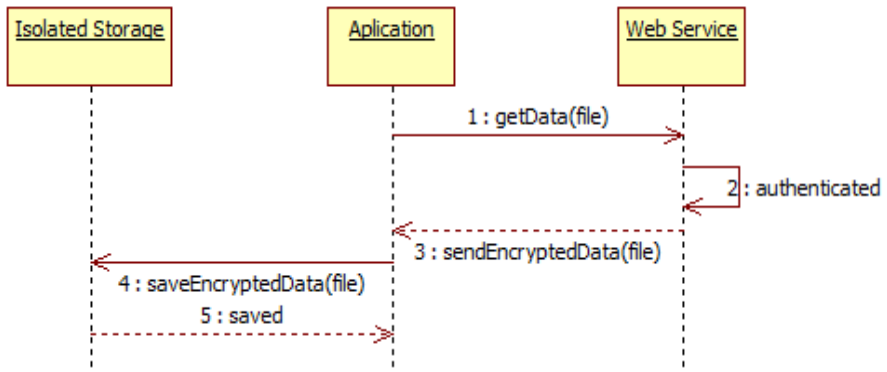

Fig. 3. The process of saving data to the Isolated Storage

The data can be saved also on the server, using the web service, not only on the device. In the case that the user wants to upload a file from the Isolated Storage to the web

service, the application get the file from the corresponding Isolated Storage, the file is encrypted with the user's correspondent key and send it to the web service, where the file is saved. All the files saved into the Isolated Storage are encrypted with the user's key. This process is shown in Figure 4.

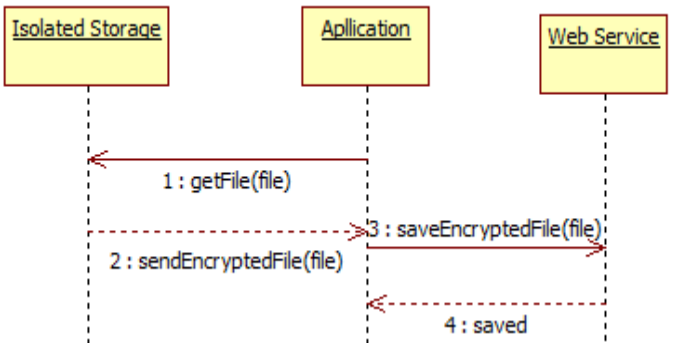

Fig. 4. The process of saving data to the web service

The storage of data on the device implies encrypting the contents with the *Ku* key and writing data. There is the case when the password is lost or needs to be changed. In this case, both the old *Ku* key and the new *Ku'* key are needed. All the encrypted data in the storage must be decrypted with the old *Ku* key and the clear data must be encrypted with the new *Ku'* key. The user cannot change the password on its own without notifying the authority. In the case when the password is changed on the application side, then it will not match the password stored by the security authority and the user will never get authenticated when connecting to the web service.

The process of changing the password consists on the following steps:

– the user notify the central authority that he lost the password or wants to change it;
– the central authority generate a new password *Pu'* for the user and send it to the user;
– using the new password *Pu'* the application connect to the web service , which send the old password *Pu* encrypted, these steps are not visible to the user;
– using the old encrypted password *Pu*, the application access and decrypt the Isolated Storage and encrypt it back using the new *Pu'* password; the old password is invisible to the user.

In Figure 5 is presented the process of changing the password. Only the central authority can change the password, and the changing process is not an everyday task. The user must keep in a secure place the password.

A secure architecture for M-learning mobile applications is presented in [5].

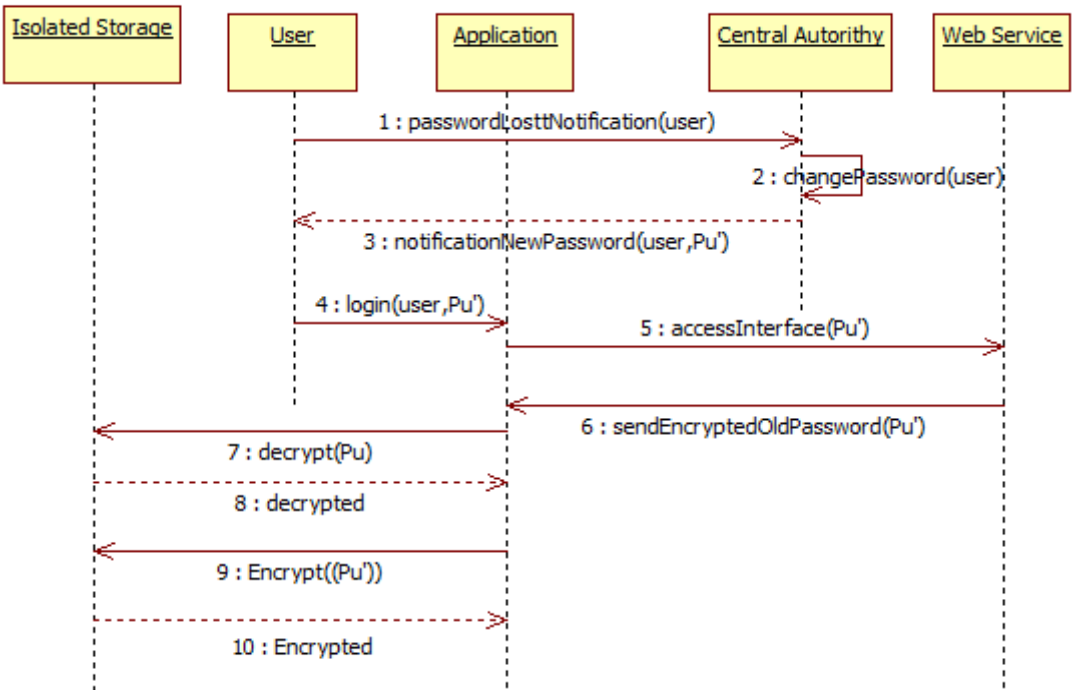

Fig. 5. The process of changing the password

4 Practical implementation of the propos architecture

The practical implementation developed to implement the proposed security architecture consists of a Windows Phone 7 application and a WCF web service.

On the device side the inputs and outputs are encrypted and decrypted using the AES symmetric algorithm. The data used by the application are received and saved using Xml format. For the security component found inside the application this type of data is handled via String data type. There are implemented methods for encrypting and decrypting a string and for saving and loading a string to a file and from a file.

Regarding network traffic,a proxy class references the web service. The web service exposes methods both for authentication and for data exchange. Every data exchange method checks if the user is authenticated. All the content exchanged between the application and the web service is encrypted with *Ku* key.

In Table 1 is presented an excerpt of network capture of the data exchange between the application and the web service.

Table 1. A network capture of the data exchange between the application and the web service

No.	App	Service
1	<StartAuthenticationS ..> <uservalue>user1</uservalue> </StartAuthenticationS>	
2		<StartAuthenticationSResponse ..><StartAuthenticationSResult> DCRv5VoXrJ59gBX2TQ+lqw== </StartAuthenticationSResult> </StartAuthenticationSResponse>
3	<FinishAuthenticationS><clarvalue>7234152</ clarvalue></FinishAuthenticationS>	
4		<FinishAuthenticationSResponse><FinishAuthenticatio nSResult>true </FinishAuthenticationSResult></FinishAuthenticationS

		Response>
5	<GetDataS><value>dataset1</value> </GetDataS>	
6		<GetDataSResponse><GetDataSResult>you get the data</GetDataSResult></GetDataSResponse>

The steps followed in the process of using the application that are seen in table 1 are:

- the application start the authentication process by sending the username "user1"to the web service;
- the web service generate a random "7234152" which is encrypted using the corresponding password for user1(DCRv5VoXrJ59gBX2TQ+lqw==) and sent to the application;
- the application decrypt the received value and finish the authentication process by sending to the web service the decrypt value obtained: 7234152;
- the received value is compared to the random value and the IsAuthenticated value is set to true, the user1 is authenticated;
- the application request the dataset1 data to the server, by using the GetDataS method;
- the user1 gets the data that he was asking for from the web service.

The authentication sequence source code is represented by the following functions:

- StartAuthentication: is called when the process of authentication starts. A new user is add to the current session, using the password of the user, a generated random , in this case, "7234152" is encrypted:

```
....
HttpContext.Current.Session.Add(uservalue, 1);
stringrezEncrypt=null;
switch (uservalue)
            {
case"user1":
rezEncrypt = EncryptString("7234152", parole[0]);
                        _random = "7234152";
username = "user1";
break;
   ….
            }
```

- StartAuthenticationSCompleted take place when the process of start authentication finish. The result obtained in the StartAuthenticationS method is decrypted, using the user password and theFinishAuthenticationSCompletedMethod is called, where the user authenticity is validated. The call to the web service is a asynchronous call, there are processes that happens when the process start and when the process finish:

```
…
voidclserv_StartAuthenticationSCompleted(object sender,
ServiceReference1.StartAuthenticationSCompletedEventArgs e)
        {
rnd = e.Result;
clar = DecryptString(rnd, parole[0]);
clserv.FinishAuthenticationSCompleted += newEventHandler<ServiceReference1.FinishAuthenticationSCompletedEventArgs>(clserv_FinishAuthenticationSCompleted);
clserv.FinishAuthenticationSAsync(clar);
        }
```

- FinishAuthenticationS and FinishAuthenticationSCompleted: these two functions determine if the user is authenticated or not and allow the user to access the following elements of the web service, to access the data via GetDataS method.

```
..
if (username == "user1"&&clarvalue == "7234152")
IsAuthenticated = true;
else
IsAuthenticated = false;
returnIsAuthenticated;
..

voidclserv_FinishAuthenticationSCompleted(object sender,
ServiceReference1.FinishAuthenticationSCompletedEventArgs e)
        {
IsAuthenticatedLocal = e.Result;
if (IsAuthenticatedLocal == true)
            {
MessageBox.Show("autenticat");
clserv.GetDataSCompleted += newEventHandler<ServiceReference1.GetDat
```

```
aSCompletedEventArgs>(clserv_GetDataSCom
pleted);
clserv.GetDataSAsync("dataset1");
            }
else
MessageBox.Show("non-autenticat");
        }
```

For accessing a method of the web servicethe following source code is used:

```
ServiceReference1.Service1Clientmyclient
= newServiceReference1.Service1Client();
string s = "sss";
myclient.OpenCompleted +=
newEventHandler<System.ComponentModel.As
yncCompletedEventArgs>(myclient_OpenComp
leted);
myclient.OpenAsync(s);
```

A reference to the web service is declared and the asynchronous process is started: calling the OpenCompleted method and then the OpenAsync method.
The capacity of the web service to respond to an asynchronous call allows the multithreading process. The thread that initiates the asynchronous call can respond to any operation while the method finished[7].
Regarding the implementation of the web service, there are two available options related to the state.
When using a web service with associated object lifetime *per session* for each new client application, a new web service object is created. This object deals with all the requests from that client, and is destroyed at the end of the dialogue session. If a large number of clients connect to the web service then a large number of objects would be created.
In Figure 6 is presented the instantiation of the per session web service.

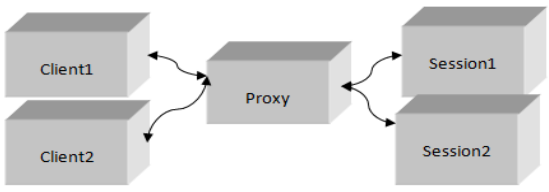

Fig. 6. The Instantiation of Per Session web service [6]

The following source code represents the way of setting the behavior of the web service to per session `[ServiceBehavior(InstanceContextMode = InstanceContextMode.PerSession)].`

The application server would need enough resources to cover all the requests. On the other hand individual clients are treated independently of other connected clients without synchronization problems.
When using a web service with associated object lifetime *single session*, then a single web service object is allocated. This is friendly with the application server's resources as only one object is created and maintained in memory.
The instantiation of the single session web service is presented in Figure 7.

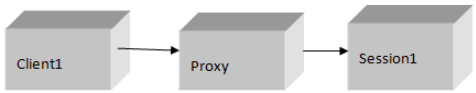

Fig. 7. The Instantiation of Single Session web service

The following source code represents the way of setting the behavior of the web service to single session `[ServiceBehavior(InstanceContextMode=InstanceContextMode.Single )].`

The problem arising with this approach is the synchronization of user requests. For each request made to the web service's methods the user identifies itself. At web service level, a user map has to be kept to store the each user's state, whether it is connected, authenticated or denied access. Regardless of the implementation the exposed functionality remains the same. There are some peculiarities that make the single session approach as it does not require reliable session binding which adds extra configuration in the service development.

5 Risks related to the proposed architecture

The proposed architecture contains several interconnected components exposed to security risks:

– standalone mobile application: at this level, the executable is present along with

configuration files and the most important, the data files
- communication channel: at this level, the network connecting the two endpoints carries the exchanged data
- web service level: at this level, there is the application server and the database server.

At each architecture level, several scenarios are considered to assess the strength of the proposed architecture.

A first scenario takes into account attacks at application level. There is protection to rewriting applications on WP7 platform but on a development enabled phone there is the possibility of updating the executable with a modified version that could expose access to the Isolated Storage. In this case the protection is given by the fact that the files stored by the application are encrypted with the key derived from the user password which is never stored. Without the key, found data is unintelligible for the attacker. More protection is achieved if file names or paths do not contain suggestive clues about the contained information.

A second scenario takes into account the communication channel between the mobile application and the web service residing on the application server. An attacker may intercept the data packets exchanged between the two entities. In this case the fact that the content is encrypted protects the data from being understood. There is some information that is sent unencrypted, as seen in Table 1, related to the web service communication protocol, the user identifier, the random number and also the XML tags of the message corresponding to the service internals. However this data is insufficient to derive encrypted content.

A third scenario is also taken into account at network level. A man in the middle case implies the presence of a third entity that exposes itself as the other party to each of the endpoints. This may cover the authentication process where the mobile application exchanges the random with the web service. The user may see itself as authenticated and also the web service may see the user as authenticated but the "you get the data" portion of the dialog is encrypted. The third party cannot send false data to the application on behalf of the web service as the data must be encrypted with the symmetric key; otherwise it cannot be decrypted by the user. The third party may request data from the web service on behalf of the application but the received data is encrypted with the symmetric key. In this case, only a denial of service can be achieved by a third party in the middle of the dialog.

A fourth scenario takes into account the application server and the associated database. At this endpoint of the communication there is concern about storing the password. As the application server is likely to be found inside the organization the security issues are related to the security of the organization itself. The password is not stored at mobile application level as the risks are high on this side. The password has to be stored by the issuer at organization level as this helps restoring saved data from the device as seen in the case of lost password. The storage of the password has to be done in a secure manner in the database with respect to database security rules and recommendations.

As seen, the proposed architecture is solid enough from security point of view, to assure the protection of data in scenarios where harmful intentions are present.

Advantages of the proposed architecture are:
- easy to implement;
- reduced costs as no certificate has to be bought by the organization;
- good protection in common cases of attacks.

Disadvantages of the proposed architecture are:
- the password has to be retained by the user and is prone to be lost, divulged or forgotten;
- it is customized for the Windows Phone 7 platform;
- the strength is given by the strength of the key derived from the password; for highly sensitive data it is recommended to use more secure architectures.

Overall the security level offered by the architecture related to the complexity its implementation makes it fit for applications were data security is important not to divulge sensitive information between users of the same service but not the primary requirement but not against heavy attacks from computer experts.

6 Conclusions

Assuring a secure architecture for Business Intelligence Mobile Applications has an important role in the security development process. Assuring the implementation of secured methods on the web service used and on the application developed for Windows Phone 7 a high degree of securityis obtained.
The proposed architecture is able to offer a reasonable protection for the data exchanged between the client application and the web service. Also, the data is secured both on the device and at application server level. The security level obtained is fit to protect data between application users. The proposed architecture is cheap to implement as it does not require a certificate issued by a known authority nor an unsigned certificate generated by the organization. The architecture is designed for Windows Phone 7, but many elements may be shared with other platforms: the web service is a common element that does not affect the mobile application, also the protocol how the methods are called is easily implemented on other platforms; only the Isolated Storage is highly specific for Windows Phone 7 platform. Implemented security needs to have little impact on usability of the application. Implemented security elements must not affect performance, or functionality of the application. The proposed architecture meets these requirements and uses authentication data that needs o be kept by the user, to assure the desired level of security.
Business Intelligence applications have to be easy to use and offer their support for decision making. Security is an element that improves these applications to new standards.

Acknowledgements

This work was co-financed from the European Social Fund through Sectoral Operational Programme Human Resources Development 2007-2013, project number POSDRU/107/1.5/S/77213 „Ph.D. for a career in interdisciplinary economic research at the European standards".

References

[1] B. Ghilic, M. Stoica and M. Mircea, "How to Succeed in Business Intelligence Initiative: A Case Study for Acquisitions in Romania Public Institutions,"*in Proc. WSEAS TRANSACTIONS on BUSINESS and ECONOMICS*, Issue 6, Vol. 5/2008, pp. 298-309.

[2] S. Trif, "Using Genetic Algorithms in Secured Business Intelligence Mobile Applications," *Informatica Economica Journal*, vol. 15, no.1/2011, Bucharest, pp.69-80.

[3] H. Dwivedi, C. Clark, D. Thiel, *Mobile Application Security*, McGraw Hill Professional Publisher, USA, 2010, pp.1-15, 79-121.

[4] System.Security.Cryptography namespace [Online] Available: http://msdn.microsoft.com/en-us/library/system.security.cryptography.aspx.

[5] C. Boja, et al.,"Secure architecture for M-learning Bluetooth services," *Informatica Economica Journal*, vol.14 no. 3, 2010, pp. 47-59.

[6] Discover Mighty Instance Management Techniques for Developing WCF Apps [Online] Available at: http://msdn.microsoft.com/en-us/magazine/cc163590.aspx.

[7] Professional ASP.NET Web Services: Asynchronous Programming [Online] Available: http://www.stardeveloper.com/articles/display.html?article=2001121901&page=1.

Silvia TRIF graduated the Faculty of Cybernetics, Statistics and Economic Informatics. She has a Master's Degree in Project Management. She is a PhD student of the Doctoral School of Bucharest Academy of Economic Studies in the field of Economic Informatics. Her interests are mobile applications, information security, web applications and project management.

Adrian VIŞOIU graduated the Bucharest Academy of Economic Studies, the Faculty of Cybernetics, Statistics and Economic Informatics. He has a master degree in Project Management. He has a PhD in the field of software quality. He is a Software Engineer in telecom field and a collaborator of Economic Informatics Department of the Bucharest Academy of Economic Studies. He published articles alone or in collaboration and he is coauthor of three books. His interests include: programming, genetic algorithms and neural networks.