http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 6 (7): 987-995, 2007
ISSN 1812-5638

A Solutions for Authentication of Web Service Users

Seifedine Kadry and Khaled Smaili
Lebanese University, Beirut, Lebanon

Abstract: Business systems, which develop software products, are constantly being challenged with the need for protection of their products from piracy. One solution is to sell those software products with limited functionality and after licensing they are to be activated i.g., by buying a license, a key which unlocks the software product is obtained and it gets its full functionality. In this study the new idea is to activate the software products by using web services, on the part of the manufacturer, which offers a better control of software products licensing. On the basis of web service is SOAP (Simple Object Access Protocol). SOAP is a technology based on XML. Web services are deployed and executed on some web server. This means that the security of the web service should be taken care of, the same as with any other web application. When we talk about the web service security we usually refer to realization of authentication and authorization of access to those web services. In addition, a custom solutions for authentication of web service users based on the RIJNDAEL algorithm has been proposed.

Key words: Web service, SOAP, authentication method, RIJNDAEL algorithm

INTRODUCTION

Today, companies rely on thousands of different software applications each with their own role to play in running a business. To name just a few, database applications store information about customers and inventories, web applications allow customers to browse and purchase products online and sales tracking applications help business identify trends and make decisions for the future. These different software applications run on a wide range of different platforms and operating systems and they are implemented in different programming languages. As a result, it is very difficult for different applications to communicate with one another and share their resources in a coordinated way. Take, for example, a company that has its customer data stored in one application, its inventory data stored in another application and its purchasing orders from customers in a third. Until now, if this company wanted to integrate these different systems, it had to employ developers to create custom bridging software to allow the different applications to communicate with one another. However, these sorts of solutions are often piecemeal and time consuming. As soon as a change is made to one application, corresponding changes have to be made to the other applications linked to it and to the bridges that link the applications together.

To solve the problem of application-to-application communication, businesses need a standardized way for applications to communicate with one another over networks, no matter how those applications were originally implemented. Web services (Iverson, 2004; Fensel and Musen, 2001; Hendler, 2001) provide exactly this solution by providing a standardized method of communication between software applications. With a standardized method of communication in place, different applications can be integrated together in ways not possible before. Different applications can be made to call on each other's resources easily and reliably and the different resources that applications already provide can be linked together to provide new sorts of resources and functionality. Moreover, application integration becomes much more flexible because web services provide a form of communication that is not tied to any particular platform or programming language. The interior implementation of one application can change without changing the communication channels between it and the other applications with which it is coordinated. As a summary, web services provide a standard way to expose an applications resources to the outside world so that any user can draw on the resources of the application.

Kent (1980) reported which addresses the security requirements of software vendors.

Protection from software copying and modification (e.g., the latter by physical attacks by users, or program-based attacks). Tools proposed to address these requirements include physical Tamper-Resistant Modules (TRMs) and cryptographic techniques; one approach involves using encrypted programs, with instructions decrypted immediately prior to execution. Kent also noted

Corresponding Author: Seifedine Kadry, Lebanese University, Beirut, Lebanon

the dual of this problem: user requirements that externally-supplied software be confined in its access to local resources. Gosler (1985) software protection survey examines circa-1985 protection technologies including: hardware security devices (e.g., dongles), floppy disc signatures (magnetic and physical), analysis denial methods (e.g., anti-debug techniques, checksums, encrypted code) and slowing down interactive dynamic analysis. The focus is on software copy prevention, but Gosler notes that the strength of resisting copying should be balanced by the strength of resisting software analysis (e.g., reverse engineering to learn where to modify software and for protecting proprietary algorithms) and that of software modification (to bypass security checks). Useful tampering is usually preceded by reverse engineering. Gosler also notes that one should expect that an adversary can carry out dynamic analysis on the target software without detection (e.g., using in-circuit emulators and simulators) and that in such a case, as a result of repeated experiments, one should expect the adversary to win. The practical defense objective is thus to make such experiments extremely arduous. Another suggestion (Gosler, 1985) is cycling software (e.g., via some forced obsolescence) at a rate faster than an adversary can break it; this anticipates the paradigm of forced software renewal. Jakobsson and Reiter (2002) who propose discouraging pirates through forced updates and software aging. This approach is appropriate where protection from attacks for a limited time period suffices. Herzberg and Pinter (1987) consider the problem of software copy protection and propose a solution requiring CPU encryption support (which was far less feasible when proposed almost 20 years ago, circa 1984-1985). Cohen (1993) reported on software diversity and obfuscation is directly related to software protection and provides a wealth of techniques and insights. Goldreich and Ostrovsky (1996) provide one of the earliest theoretical foundation pieces. They reduce the problem of software protection-which they take to mean unauthorized software duplication-to that of efficient (in the theoretical sense) simulation on oblivious RAMs. A new issue they address is the extraction of useful information gained by an adversary examining the memory access patterns of executing programs. To address this, oblivious RAMs replace instruction fetches in the original program by sequences of fetches, effectively randomizing memory access patterns to eliminate information leakage.

In this study, the novel idea is to activate the software products by using web services, which is platform independent, on the part of the manufacturer, which offers a better control of software products licensing. Additionally, a custom solutions for authentication of web service users based on the RIJNDAEL algorithm has been proposed.

WEB SERVICE AUTHENTICATION

On web services, authentication of service user can be achieved in three different ways (Adams and Boeyen, 2002):

- Platform level authentication
- Message level authentication
- Application level authentication.

PLATFORM LEVEL AUTHENTICATION

Windows authentication is used in cases when you control endpoints which are in the same or trusting domain.

Basic authentication: On MS Internet Information Server, it is possible to configure authentication on web service's virtual directory as basic authentication. In that case web service user must configure the proxy and provide credentials in the form of a user name and password. The proxy transmits the user name and password with each request towards the web service. The user name and password are transmitted in plaintext so Basic authentication is secure only if SSL is used. Code 1 shows how a web application can reach the user name and password when Basic authentication on MS IIS is in question.

```
//Retrieve client's credentials (available with Basic authentication)
string pwd =
Request.ServerVariables[AUTH_PASSWORD];
string uid = Request.ServerVariables[AUTH_USER];
//Set the credentials
CredentialCache cache = new CredentialCache();
cache.Add(new Uri(proxy.Url), //Web service URL
    Basic,
    new NetworkCredential(uid, pwd, domain));
proxy.Credentials = cache;
```

Code 1: Retrive client's credentials

Integrated windows authentication: If web service's virtual directory on MS IIS is configured for Integrated Windows authentication (which results in Kerberos or NTLM authentication which further depends on the client and server environment). The advantage of this approach compared with basic authentication is in the fact that the user name and password are not transmitted over the

network. In this case the user must explicitly configure the Credentials property on the proxy. In case the solution in which the user name and password are placed in web.config file is chosen, then they have to be encrypted by using DPAPI and not stored in plaintext.

Message level authentication: Message level authentication uses SOAP (Brose, 2003) header for passing data about the user name end password, assigned tickets and certificates. This type of authentication can be realized in several ways:

- Transmitting user name and password
- Transmitting Kerberos ticket
- Transmitting X.509 certificate
- Transmitting custom tokens.

User name and password: User name and passwords can be transmitted in the SOAP header as part of his <Security> element as it follows in the Code 2.

```
<wsse:Security
    xmlns:wsse:=http://shemas.xmlsoap.org/ws/2002/12
/secext">
    <wsse:UsernameToken>
    <wsse:Username>Goran</wsse:Username>
    <wsse:Password>P@ssw0rd</wsse:Password>
     </wsse:UsernameToken>
</wsse:Security>
```

Code 2. Sending the User name and password in the SOAP header

Kerberos ticket: In code 3, an example of sending a Kerberos ticket as part of the SOAP header is given.

```
<wsse:Security
    xmlns:wsse:="http://shemas.xmlsoap.org/ws/2002/1
2/secext">
    <wsse:BinarySecurityToken>
        ValueType="wsse:Base64Binary">
        U87GGH91TT...
     </wsse:BinarySecurityToken>
</wsse:Security>
```

Code 3. Sending a Kerberos ticket in the SOAP header

X.509: In order to achieve authentication of Web service, data about X.509 can be sent in the SOAP header as it follows in Code 4.

```
<wsse:Security
xmlns:wsse:="http://shemas.xmlsoap.org/ws/2002/12/se
cext">
    <wsse:BinarySecurityToken>
        ValueType="wsse:X509v3">
        Hg6GHjis1...
     </wsse:BinarySecurityToken>
</wsse:Security>
```

Code 4. Sending a X.509 certificate in the SOAP header

Application level authentication: Application Level Authentication, when Microsoft technologies are in question, can be realized in two ways:

- Authentication by using custom SOAP heder
- Developing a custom mechanism for authentication by using standard algorithms for encrypting from the System.Security.Cryptography.

WEB SERVICE AUTHORIZATION

After authentication of the Web service caller, restrictions on the level of functionality can be defined based on the identity of the caller or role memebership that he belongs to Short (2002), Kao (2001) and Raina (2004). Restricted access can be defined at the endpoint of the service (on the level of. asmx files), web methods or at the specific functionality inside the web method.

Web service endpoint authorization: If the web service is configured for integrated windows authentication, NTFS permissions can be configured (based on the security context of the original service caller) for access to it on the level of asmx files.

Web method authorization: For access to web methods that are part of web service, when individual access or access based on role membership is in question, the principal object associated with the current Web request can be used (access through HttpContext.User). The example of the code follows in Code 5.

```
[PrincipalPermission(SecurityAction.Demand,
Role=@"Manager")]
[WebMethod]
public string QueryEmployeeDetails(string empID)
{
}
```

Code 5: Using principal object.

Programmatic authorization: For more subtle authorization inside the web service method IPrincipal.IsInRole can be used (imperative permission checks or explicit role checks) as it can be seen in Code 6.

```
//This assumes non-Windows authentication. With Windows authentication cast the User
//object to a WindowsPrincipal and use Windows groups as role name
GenericPrincipal user = User as GenericPrincipal;
if (null != user)
{
    if (user.IsInRole(@"Manager")
        {
        // User is authorized to perform manager funkciomality
    }
}
```

Code 6: Using IPrincipal.IsInRole

RIJNDAEL ALGORITHM

Rijndael (Murphy and Robshaw, 2000) is a symetric algorythm for encryption (Simmetric key block cipher) and it is the winner of the US Advanced Encryption Standard competition. Rijndael combination of safety, performance, efficiency, ease of implementation and flexibility make it a fine choice when custom security solutions are considered.

When web services are considered, SESO system developing team has decided to develop a custom mechanism for authentication of service users. For that purpose, a Rijndael class, that is part of Namespace System.Security.Cryptography Microsoft. NET Framework, was used. The examples of the code (C#. NET) for encryption and decryption are given in the Code 9 and Code 10. The methods given in the examples are part of the class of SESO project.

```
public byte[ ] EncryptData(string Inputstring, string Key, string Addition)
{
    byte[ ] errorencrypted = new byte[0];
    if (Inputstring != "" and and Key != "" and and Addition != "")
    {
        try
        {
        ASCIIEncoding textConverter = new ASCIIEncoding();
        RijndaelManaged myRijndael = new RijndaelManaged();
        byte[] encrypted;
        byte[] toEncrypt;
        byte[] key = textConverter.GetBytes(Key + Addition);
        byte[] IV = textConverter.GetBytes(Key + Addition);
        //Return encryptor.
        ICryptoTransform encryptor = myRijndael.CreateEncryptor(key, IV);
        //Encrypting data.
        MemoryStream msEncrypt = new MemoryStream();
        CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write);
        //Convert data to byte array.
        toEncrypt =textConverter.GetBytes (Inputstring);
        //Write all data to crypto stream and transfer them
        csEncrypt.Write(toEncrypt, 0, to Encrypt.Length);
        csEncrypt.FlushFinalBlock();
        //Return encrypted array of bytes.
        encrypted = msEncrypt.ToArray();
        return encrypted;
        }
        catch
        {
        return errorencrypted;
        }
    }
    else
    {
        return errorencrypted;
    }
}
```

Code 7: The method for data encryption

```
private stringDecryptData(byte[ ] toDecrypt, string Key, string Addition)
{
    string Outputstring = "";
    try
    {
    if (toDecrypt.Length > 0)
    {
        ASCIIEncoding textConverter = new ASCIIEncoding();
        RijndaelManaged myRijndael = new RijndaelManaged();
        byte[ ] fromEncrypt;
        byte[ ] key = textConverter.GetBytes(Key + Addition);
        byte[ ] IV = textConverter.GetBytes(Key + Addition);
```

```
        ICryptoTransform decryptor =
myRijndael.CreateDecryptor(key, IV);
        MemoryStream msDecrypt = new
MemoryStream(toDecrypt);
        CryptoStream csDecrypt = new
CryptoStream(msDecrypt, decryptor,
CryptoStreamMode.Read);
        fromEncrypt = new byte[toDecrypt.Length];
        csDecrypt.Read(fromEncrypt, 0,
        fromEncrypt.Length);
        Outputstring =
        textConverter.GetString(fromEncrypt);
        int Position = 0;
        Position = Outputstring.IndexOf("\0", prmlPozicija
        + 1);
        if (Position!= -1)
        {
            Outputstring = Outputstring.Substring(0,
            Position);
        }
        return Outputstring;
    }
    else
    {
        return Outputstring;
    }}
    catch
    {
        return Outputstring;
    }}
```

Code 8: The method for data decryption

DESCRIPTION OF THE APPLICATION

SESO is a software product of the company PC which possesses the functionality of a send mail and SMS organizer and is intended for mass sale. SESO, along with the windows forms application within itself, possesses a system for licensing and application activation through web service and a system for mass sending of SMS messages through PC messages system (on the side of the customer) or through the web service of some SMS provider (Fig. 1).

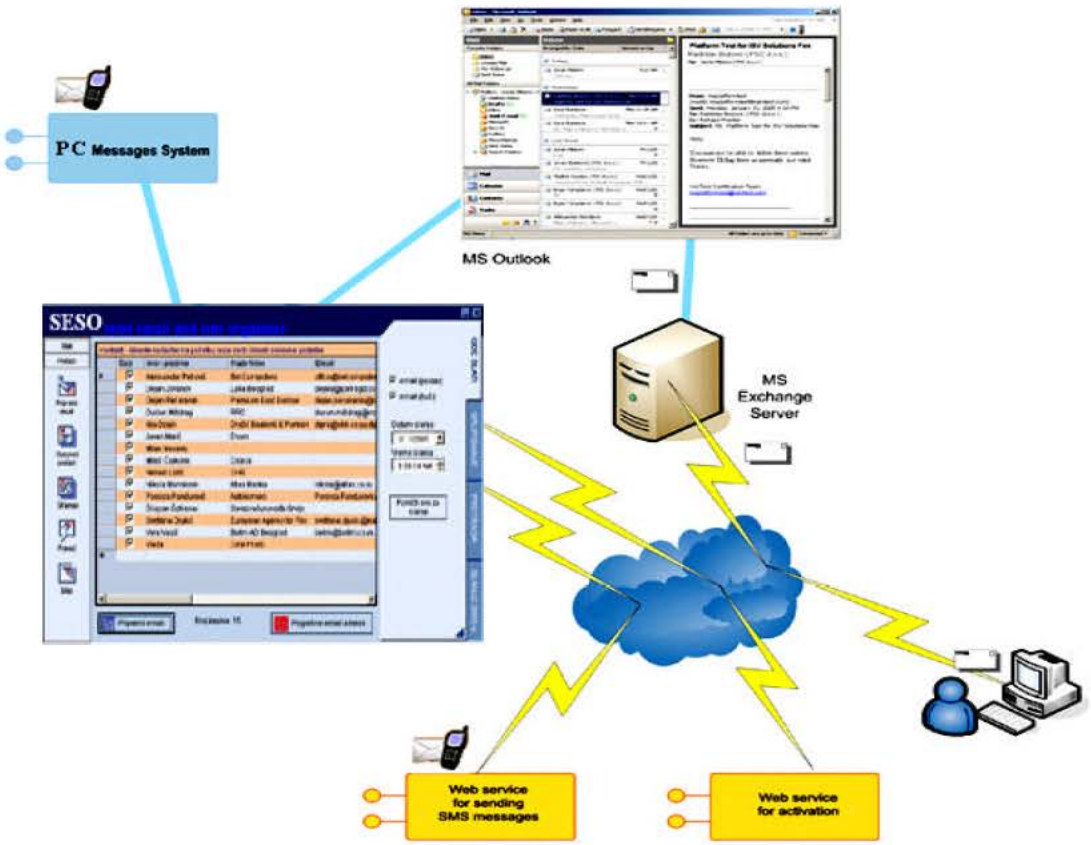

Fig. 1: The functionality of sending email and SMS messages of the application SESO

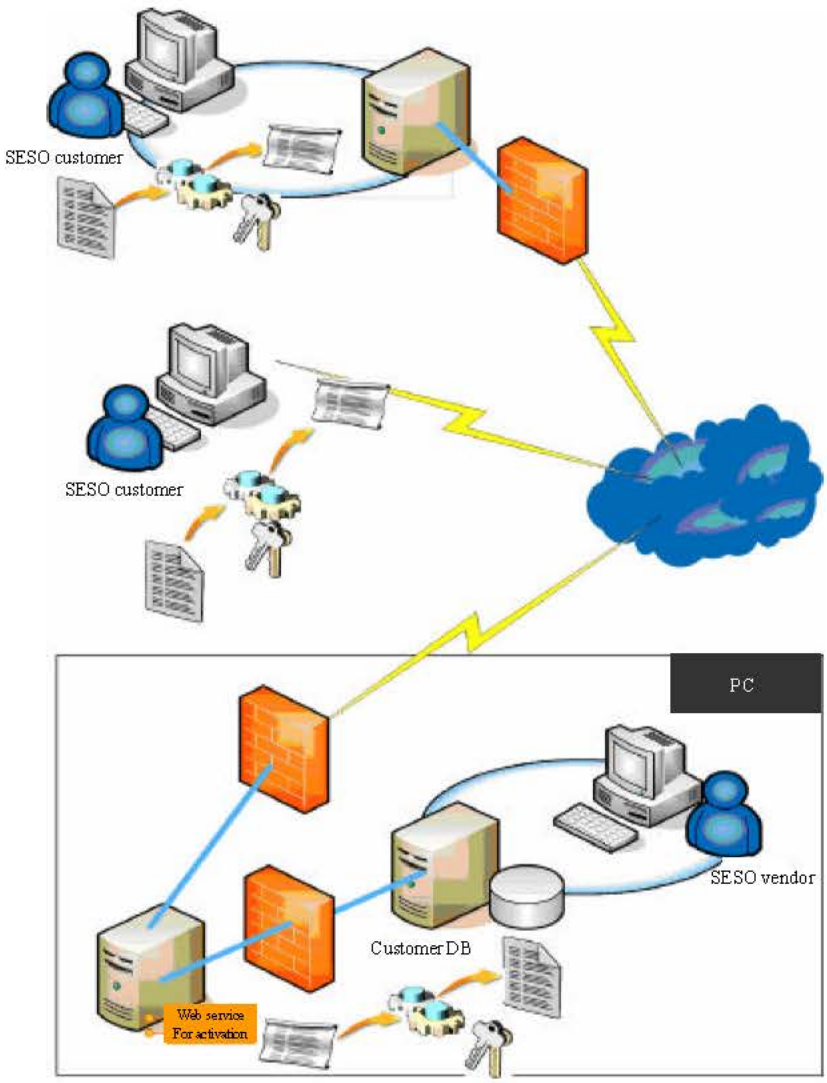

Fig. 2: System architecture of the system for licensing and activation of the SESO

An installation version of the application has got limited functionality (working with 15 records only, the possibility of sending only 5 SMS messages and activity period of 60 days).

LICENSING AND APPLICATION ACTIVATION

The system for licensing and activation of the application SESO is based on the web service for licensing and activation (Ben-Itzhak, 2005; Wassermann and Su, 2004; Benjamin, 2005; Huang *et al*., 2003, 2004). It goes without saying that for performing licensing and activation, an active connection to the Internet is needed (Fig. 2).

The system for licensing and activation of the application SESO is based on a pair of numbers (Id and Key of a customer) which a customer obtains from a PC employee in direct contact (Fig. 3).

Through the user application the employee inserts data about the customer into the database and then assigns him free Id from the table with already created pairs of numbers Id and Key. Id is generated by a generator of random numbers and consists of six numbers. The number of possible combinations is 6^6. The Key of the customer is a number of fifteen figures which is generated by an algorythm developed by PC for this purpose.

Fig. 3: Id and Key are assigned to the customer in direct contact only

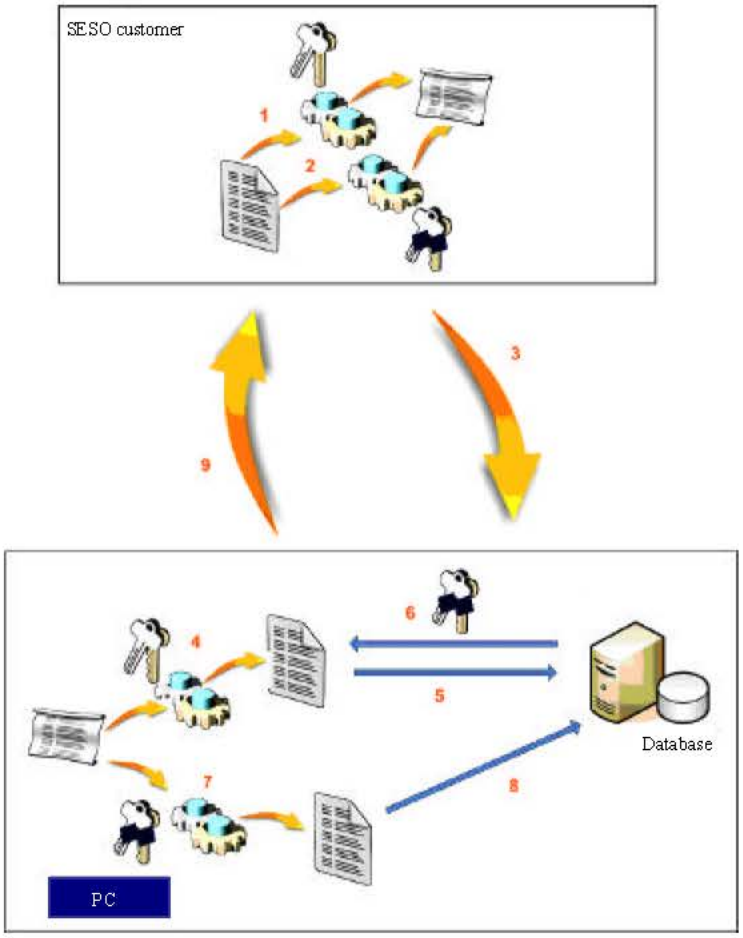

Fig. 4: Licensing and activation of the application SESO

Values for the Id and Key of the customer are not repeated in the base. This itself indicates that the pair of numbers for the Id and Key is unique and that it denotes the customer. Every customer can buy n number of licenses (the number of licenses corresponds to the number of computers on which SESO will be installed) and the employee at PC adds them to the data base of customers through the user application.

Activation of the application SESO begins with entering a pair of numbers Id and Key in the fields on the form for activation which is a part of the application SESO. After starting the activation the Id of a customers is

encrypted by the Private key (Step 1, Fig. 4). Then, all the other data are encrypted (Key of the customer, active user, computer name, domain name or work group) and are sent to the web service for activation (Step 2, Fig. 4). For encryption of these data the key of the customer is used. After establishing communication with the web service for activation, encrypted data are sent to its method (Step 3, Fig. 4). Web service method decrypts the encrypted Id of the customer with the pair of keys by which it is encrypted (Step 4, Fig. 4) and then from the data base of customers it takes its pair for the Key of the customer (Steps 5 and 6, Fig. 4).

Using the value that the Key of the customer possesses the encrypted data for the Key of the customer that the application SESO has sent is decrypted first (Step 7, Fig. 4). If the value for the Key of the customer agrees with that acquired from the data base then the status of the licenses is checked i.e., we check whether the customer has more free licenses in the data base. If the customer has some free licenses in the data base, then all the other data that the application SESO has sent are encrypted (active user, computer name, domain name, or work group) by using the data for the Key of the customer as a key, one free license of the customer is checked and the decrypted data are inserted into the data base (Step 8, Fig. 4). Then the web service method for activation returns the confirmation of activation to the application SESO (Step 9, Fig. 4) which activates and acquires its full functionality.

The Rijndael symetric algorithm for encrypting, which is part of the MS. NET Framework, is used for encryption.

CONCLUSIONS

The advantages of the presented solution, compared with the classic systems of licensing and activation of an application where the mechanism for licensing and activation is completely on the client, are based on the fact that the software vendor has a better control over the number of installations done. When the problem of software piracy is in question, even if a crack which generates pairs of numbers for Id and Key of the customer appears, there is a great possibility that the generated pairs of numbers are not in the base of the software vendor which unable licensing and activation of the application (not all combinations of pairs of Id and Key of the customer are present in the base of software vendor). Additional safety measure is the fact that the number of licenses is added separately for every pair of numbers (Id and Key of the customer) only after a direct contact of the customer and the software vendor. By using Rijndael algorithm for encryption, it is achieved that on the side of vendor's web services there does not have to be Server digital certificate that would enable using https as protocols for communication with the web services and therefore for encrypting data.

REFERENCES

Adams, C. and S. Boeyen, 2002. UDDI and WSDL extensions for web services: A security framework. Proceedings of the ACM Workshop on XML Security, pp: 30-35.

Ben-Itzhak, Y., 2005. Web application security-the next evolution. http://www.devx.com/security/Article/10236.

Benjamin, L., 2005. Defining a set of common benchmarks for web application security. Defining the state of the art in software security tools. (SoftSecTools'05 Proceedings) ISBN # 1-59593-179-1/05/08. SoftSecTools.

Brose, G., 2003. Securing web services with SOAP security proxies. Proceedings of the ICWS 03 International Conference on Web Services, pp: 231-234.

Cohen, F., 1993. Operating system protection through program evolution. Comput. Security, 12: 565-584.

Fensel, D. and M. Musen, 2001. Special issue on semantic web technology. IEEE Intell. Syst. (IEEE IS), 16: 60-71.

Goldreich, O. and R. Ostrovsky, 1996. Software protection and simulation on oblivious RAMs. J. ACM, 43: 431-473. Based on earlier ideas of Goldreich (STOC'87) and Ostrovsky (STOC'90).

Gosler, J., 1985. Software Protection: Myth or Reality?. Advances in Cryptology-CRYPTO'85, Springer-Verlag LNCS 218: 140-157.

Hendler, J., 2001. Agents and the semantic web. IEEE Intelligent Systems (IEEE IS), 16: 30-37.

Herzberg, A. and S.S. Pinter, 1987. Public protection of software. ACM Trans. Comput. Syst., 15: 371-393. Earlier version in Crypto'85.

Huang, Y.W., T.P. Lin and C.H. Tsai, 2003. Web application security assessment by fault injection and behavior monitoring. World Wide Web, pp: 148-159.

Huang, Y.W., F. Yu, C. Hang, C.H. Tsai, D.T. Lee and S.Y. Kuo, 2004. Securing web application code by static analysis and runtime protection. World Wide Web, pp: 40-52.

Iverson, W., 2004. Real World Web Services. O'Reilly.

Jakobsson, M. and M.K. Reiter, 2002. Discouraging Software Piracy Using Software Aging. Proc. 1st ACM Workshop on Digital Rights Management (DRM 2001), Springer LNCS 2320, pp: 1-12.

Kao, J., 2001. Developer's Guide to Building XMLbased Web Services with the Java 2 Enterprise Edn., The Middleware Company.

Kent, S., 1990. Protecting externally supplied software in small computers. Ph.D Thesis, MIT.

Murphy, S. and M. Robshaw, 2000. New observations on rijndael, information security group. Royal Holloway, University of London, Egham, Surrey, TW20 0EX, UK.

Raina, K., 2004. Trends in web application security. http://www.securityfocus.com/infocus/1809.

Short, S., 2002. Building XML Web Services for the Microsoft. Net Platform, Microsoft Press.

Wassermann, G. and Z. Su, 2004. An analysis framework for security in web applications. Specification and verification of component-based systems. Workshop at ACM SIGSOFT 2004. SAVCBS '04.