Комп'ютерний практикум №7

Тема: Електронно-цифровий підпис на основі алгоритму RSA

Мета: Ознайомитись з порядком утворення підпис на основі алгоритму RSA

Базові відомості

Електронно-цифровий підпис (ЕЦП) – вид електронного підпису, отриманого за результатом криптографічного перетворення набору електронних даних, який додається до цього набору або логічно з ним поєднується і дає змогу підтвердити його цілісність та ідентифікувати підписувача.

ЕЦП накладається за допомогою особистого ключа та перевіряється за допомогою відкритого ключа.

Одним із видів електронно-цифрового підпису є ЕЦП на основі криптографічної системи RSA. Розглянемо принципи його функціонування.

Схема створення цифрового підпису RSA полягає у такому:

1. З вхідного повідомлення створюється 160-бітовий дайджест повідомлення (геш) за допомогою алгоритмуSHA-l.
2. Дайджест повідомлення шифрується за допомогою секретного ключа RSA, відомого тільки відправнику повідомлення. Результат цього шифрування і називають цифровимпідписом.
3. Підписане повідомлення формується об'єднанням вихідного повідомлення, його цифрового підпису та відкритогоключа.

Схема перевірки цифрового підпису RSA полягає у такому:

1. Отримане повідомлення розбивається на три компоненти: на вихідне повідомлення, відкритий ключ і цифровийпідпис.

2. Знову обчислюється гешповідомлення.
3. Якщо геш збігається з розшифрованим гешем, то підпису вважається перевіреним.

На платформі .NET цифровий підпис RSA реалізується за допомогою об'єктів класу **RSACryptoServiceProvider** з простору імен **System.Security.Cryptography**.

Порядок створення цифрового підпису полягає у такому:

1. Створюється контейнер з ключами зазамовчуванням:

```
CspParameters signParam = new CspParameters(); signParam.KeyContainerName

= "Bob";

RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(signParam);

//Для контролю можна виввести згенерований секретний ключ

//Console.WriteLine("---Secret key:\n{0}\n", rsa.ToXmlString(false));
```

2. Публічний ключ експортується для передачі іншій стороні в XML- форматі:

```
string pubKey = rsa.ToXmlString(false);
Console.WriteLine("---Public key:\n{0}\n", pubKey);
```

3. Вхідне повідомлення представляється у вигляді байтовоїпослідовності:

```
byte[] message = Encoding.UTF8.GetBytes("Hello world!");

//Console.WriteLine("---HexMessage: \n{0}\n", BitConverter.ToString(message));
```

4. Створюється дайджест повідомлення за алгоритмомSHA-1:

```
SHA1 sha1 = new SHA1CryptoServiceProvider(); byte[] hmessage

= sha1.ComputeHash(message);

//Console.WriteLine("---Hesh: \n{0}\n", BitConverter.ToString(hmessage));
```

5. Створюється підпис для дайджесту, наприклад, в 16-мупредставленні:

```
byte[] signature = rsa.SignHash(hmessage, "sha1");

Console.WriteLine("---Signature: \n{0}\n", BitConverter.ToString(signature));
```

6. Підпис додається доповідомлення.

Порядок перевірки цифрового підпису полягає у такому:

1. Вхіднеповідомленняпредставляєтьсяувиглядібайтовоїпослідовності:

```
byte[] message = Encoding.UTF8.GetBytes("Hello world!");

Console.WriteLine("---HexMessage: \n{0}\n ", BitConverter.ToString(message));
```

2. Створюється дайджест повідомлення за алгоритмомSHA-1:

```
SHA1 sha1 = new SHA1CryptoServiceProvider(); byte[]
hmessage = sha1.ComputeHash(message); Console.WriteLine("---Hesh: \n{0}\n",
BitConverter.ToString(hmessage));
```

3. Створюється об’єкт RSA, до якого експортується публічний ключ з контейнера:

```
RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(); string pubKey =

  "<RSAKeyValue><Modulus>mJ32CjZjzD2Qw4oRc/3O4xyVBLLPHeXELhIa9kwPhPCERkhuvROalKgfod
SgOrVr2K9TgTAMw3Ua4Q9O/vWhJRzsQPcSEYc5wJuHU9QY0a9jPn7WJQY91uCwfA54GsVYTLo2VI60D
Et

uES7Dhlj4VK5KemvudEkmHiY8/BfO0XM=</Modulus><Exponent>AQAB</Exponent><P>z9JDG1Avu9
q5LlBNG6NQcxdhZc+HnJsQahjmza/fZb6T/K0tThAjAM18dH2gCCac05oJn2QEyCbtW5RgQ3lgvQ==</P

   ><Q>u/9yCtIK1GjfS16K5pNkSqdVZr2QDHbjy0cHiO7LfKBtWugUyN3FCSSUBooF2DmNFvRKDm1
mZ4iPP
7nMsqYV7w==</Q><DP>burHyjIX5+kq4J8XKGMXsvWNlCsZM+pG7nlv5eOyFbmLflZnUbynEeyaOgo
6eV
8yYHVcIWfebXzpPfGJFzoW+Q==</DP><DQ>hZeph6zoyzZW7u0ZEW7NxwsP8flk4qadizdHUHQ4n7A0
5X
OkSXTmbm/SzK7KJnQHIbeo5IWzToFJIkS7BHxnew==</DQ><InverseQ>jbhzN1BGEp4rA8njVFoHv8nY
oywX0fG0tVoeMkuu+V5St81pD5oY5rz+OMksTxD+scpSF8DnhEewtBPMDOTJDg==</InverseQ><D>P2
u
U7NWBT0RePgPIEO1t5c7g1h0AGKp8hbCcZ7Ff2ZyhOxuqeQQGfrQGwRc8pmjxsg/ZoZu4Ehk93DyiVSz
5 k3AfGe7vW+Mwk6jM71deoSyBdKpP1S/cpssOJHY781Tcx8jqqmP/gN19jEUkU7mJcmv4ahkis79THvo/F
vAmkJE=</D></RSAKeyValue>"; rsa.FromXmlString(pubKey);
```

4. Вхіднеповідомленняпредставляєтьсяувиглядібайтовоїпослідовності:

```
byte[] message = Encoding.UTF8.GetBytes("Hello world!"); Console.WriteLine("--- HexMessage: \n{0}\n ",
BitConverter.ToString(message));
```

5. Перевіряється підпис для дайджесту:

```
string hexSign =
"4725D1135F715C2FCFA2C6E93C839B106F1F8B2481CD599A5062652C39D2B63675D5254377B7400A
```

F85E3338F8023AA53D59AEC0144815C76FD02E22C0F3D8B976CDCDCF1DE42BAF7D4314C3124B54E4B
17B114A6E226001913AE15939584001AA1E98FA81C1F435F0F272DFEED1C27476B6F2C3E5A7AF968C

AC149892B7A198"; byte[] signature = StringToByteArray(hexSign); bool match = rsa.VerifyHash(hmessage, "sha1", signature); Console.WriteLine("Verdict:\n{0}\n", match);

Тут перетворення 16-вого цифрового підпису в байтовий масив використовуєтьсяфункція:

```
public static byte[] StringToByteArray(string hex)
{ return Enumerable.Range(0, hex.Length)
    .Where(x => x % 2 == 0)
    .Select(x => Convert.ToByte(hex.Substring(x, 2), 16)) .ToArray();
}
```

Можна обійтись без перетворення 16-вого цифрового підпису в байтовий масив. Для цього цифровий підпис треба зберегти,наприклад, у форматі Base64: byte[] signature = rsa.SignHash(hmessage,"sha1");

Console.WriteLine("---Signature: \n{0}\n", Convert.ToBase64String(signature));

Тоді перевірити підпис можна так:

```
string base64Sign =
"RyXRE19xXC/PosbpPIObEG8fiySBzVmaUGJlLDnStjZ11SVDd7dACvheMzj4AjqlPVmuwBRIFcdv0C4iwPPYuXbNzc8d5CuvfUMUwxJLVOSxexFKbiJgAZE64Vk5WEABqh6Y+oHB9DXw8nLf7tHCdHa28sPlp6+Wj KwUmJK3oZg="; byte[] signature = Convert.FromBase64String(base64Sign);

Console.WriteLine("---Signature: \n{0}\n", Convert.ToBase64String(signature));
```

Хід виконання роботи

1. Розробіть інтерфейс системи формування і перевірки ЕЦП RSA, передбачивши окремий діалог для формування відкритогоключа.
2. Розробіть методи, які бзабезпечували:
    a. Генерацію пари «відкритий –закритий»ключі.
    b. Підписування довільного повідомлення з використанням закритого ключа.
    c. Перевірки ЕЦП з використанням відкритогоключа.

3. Перевірте правильність роботи системи на основі використання даних з чисельного прикладу.

Додаткові завдання:

1. Ознайомтесь з алгоритмом цифрового підписуDSA(Digital Signature Algorithm) та описом криптопровайдера .NET з реалізацією цього алгоритму **DSACryptoServiceProvider** з простору імен **System.Security.Cryptography**. Побудуйте систему формування і перевірки ЕЦП за алгоритмомDSA.

Додаток 1.

Лістинг програми зі створення і перевірки ЕЦП

RSA.

```
using System; usingSystem.Text;

usingSystem.Security.Cryptography; using System.Linq;


namespace RSA_sign

{

class Program

{


//Функціястворенняпідпису static void GetSign()

{

Console.WriteLine("\n<<<GetSign>>>:\n");


//Створюється контейнер з ключамизазамовчуванням
CspParameterssignParam

= new CspParameters();

signParam.KeyContainerName = "Bob";

RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(signParam);

//Для контролю виводиться згенерований секретний ключ Console.WriteLine("---Secret
key:\n{0}\n", rsa.ToXmlString(false));

//Публічний ключ експортується для передачіншій стороні
stringpubKey

= rsa.ToXmlString(false);

Console.WriteLine("---Public key:\n{0}\n", pubKey);


//Вхідне повідомлення представляється у вигляді байтовоїпослідовності byte[] message =
Encoding.UTF8.GetBytes("Hello world!");

Console.WriteLine("---HexMessage: \n{0}\n", BitConverter.ToString(message));


//Створюється дайджест повідомлення за алгоритмом SHA-1 SHA1 sha1
=newSHA1CryptoServiceProvider(); byte[] hmessage =
sha1.ComputeHash(message);

Console.WriteLine("---Hesh: \n{0}\n", BitConverter.ToString(hmessage));

//Створюється підпис для дайджесту

byte[] signature = rsa.SignHash(hmessage, "sha1");

Console.WriteLine("---Signature: \n{0}\n", Convert.ToBase64String(signature));

}
```

```
//Функціяперевіркипідпису                static void VerifySign()

{

Console.WriteLine("\n<<<VerifySign>>>:\n");


//Вхідне повідомлення представляється у вигляді байтовоїпослідовності byte[] message =
Encoding.UTF8.GetBytes("Hello world!");

Console.WriteLine("---HexMessage: \n{0}\n ", BitConverter.ToString(message));


//Створюється дайджест повідомлення за алгоритмом SHA-1 SHA1 sha1 =
newSHA1CryptoServiceProvider();

byte[] hmessage =sha1.ComputeHash(message);

Console.WriteLine("---Hesh: \n{0}\n", BitConverter.ToString(hmessage));

//Створюється          об’єкт          RSA          з          розміщенням          ключів          у


контейнері RSACryptoServiceProvider rsa =newRSACryptoServiceProvider();

                        string pubKey =

"<RSAKeyValue><Modulus>mJ32CjZjzD2Qw4oRc/3O4xyVBLLPHeXELhIa9kwPhPCERkhuvRO
alKgfodSgOr

V

r2K9TgTAMw3Ua4Q9O/vWhJRzsQPcSEYc5wJuHU9QY0a9jPn7WJQY91uCwfA54GsVYTLo2V
I60DEtuES7Dhlj4V
K5KemvudEkmHiY8/BfO0XM=</Modulus><Exponent>AQAB</Exponent><P>z9JDG1Avu9q5LlBNG6N
QcxdhZ
c+HnJsQahjmza/fZb6T/K0tThAjAM18dH2gCCac05oJn2QEyCbtW5RgQ3lgvQ==</P><Q>u/9yCtIK1GjfS1
6K
5pNkSqdVZr2QDHbjy0cHiO7LfKBtWugUyN3FCSSUBooF2DmNFvRKDm1mZ4iPP7nMsqYV7w==</Q>
<DP>burHyj
IX5+kq4J8XKGMXsvWNlCsZM+pG7nlv5eOyFbmLflZnUbynEeyaOgo6eV8yYHVcIWfebXzpPfGJFzoW+
Q==</DP

><DQ>hZeph6zoyzZW7u0ZEW7NxwsP8flk4qadizdHUHQ4n7A05XOkSXTmbm/SzK7KJnQHIbe
o5IWzToFJIkS7B
Hxnew==</DQ><InverseQ>jbhzN1BGEp4rA8njVFoHv8nYoywX0fG0tVoeMkuu+V5St81pD5oY5rz+OMk
sTxD+
scpSF8DnhEewtBPMDOTJDg==</InverseQ><D>P2uU7NWBT0RePgPIEO1t5c7g1h0AGKp8hbCcZ7Ff2Z
yhOxuq
eQQGfrQGwRc8pmjxsg/ZoZu4Ehk93DyiVSz5k3AfGe7vW+Mwk6jM71deoSyBdKpP1S/cpssOJHY781Tcx
8jqqm P/gN19jEUkU7mJcmv4ahkis79THvo/FvAmkJE=</D></RSAKeyValue>";

rsa.FromXmlString(pubKey);


//Перевіряється  підписдлядайджесту                        string base64Sign =

"RyXRE19xXC/PosbpPIObEG8fiySBzVmaUGJlLDnStjZ11SVDd7dACvheMzj4AjqlPVmuwBRIFc
dv0C4iwPPY

u
XbNzc8d5CuvfUMUwxJLVOSxexFKbiJgAZE64Vk5WEABqh6Y+oHB9DXw8nLf7tHCdHa28sPlp6+WjK
wUmJK3oZg

=";
```

```
        byte[] signature = Convert.FromBase64String(base64Sign);

        Console.WriteLine("---Signature: \n{0}\n", Convert.ToBase64String(signature));


        bool match = rsa.VerifyHash(hmessage,"sha1",signature);
Console.WriteLine("-

        --Verdict:\n{0}\n", match);


        }


        //*************************************************************************//
static void Main()

        {

        GetSign(); VerifySign();

        }

        }

        }
```

Додаток 2.

Лістинг програми зі створення і перевірки ЕЦП DSA.

```
        using System; using System.Text;
```

```
usingSystem.Security.Cryptography;                using System.Linq;

namespace DSA_sign

{

class Program

{

//Функціястворенняпідпису                static void GetSign()

{

Console.WriteLine("\n<<<GetSign>>>:\n");

//Створюється об’єкт DSA з ключами за замовчуванням DSACryptoServiceProvider dsa = new
DSACryptoServiceProvider();

//Для контролю виводиться згенерований секретний ключ string prKey = dsa.ToXmlString(true);

Console.WriteLine("---Secret key:\n{0}\n", dsa.ToXmlString(true));

//Публічний ключ експортується для передачііншій стороні
stringpubKey

= dsa.ToXmlString(false);

Console.WriteLine("---Public key:\n{0}\n", pubKey);

//Вхідне повідомлення представляється у вигляді байтової послідовності byte[] message =
Encoding.UTF8.GetBytes("Hello world!");

Console.WriteLine("---HexMessage: \n{0}\n", BitConverter.ToString(message));

//Створюється дайджест повідомлення за алгоритмом SHA-1 SHA1 sha1
=newSHA1CryptoServiceProvider(); byte[] hmessage =
sha1.ComputeHash(message);

Console.WriteLine("---Hesh: \n{0}\n", BitConverter.ToString(hmessage));

//Створюється підпис для дайджесту

byte[] signature = dsa.SignHash(hmessage, "sha1");

Console.WriteLine("---Signature: \n{0}\n", Convert.ToBase64String(signature));

}

//Функціяперевіркипідпису                static void VerifySign()

{

Console.WriteLine("\n<<<VerifySign>>>:\n");

//Вхідне повідомлення представляється у вигляді байтовоїпослідовності byte[] message =
Encoding.UTF8.GetBytes("Hello world!");

Console.WriteLine("---HexMessage: \n{0}\n ", BitConverter.ToString(message));
```

```
//Створюється дайджест повідомлення за алгоритмом SHA-1 SHA1 sha1 =
newSHA1CryptoServiceProvider();

byte[] hmessage =sha1.ComputeHash(message);

Console.WriteLine("---Hesh: \n{0}\n", BitConverter.ToString(hmessage));

//Створюється об’єкт DSA, до якого експортується публічний ключ з контейнера
DSACryptoServiceProvider dsa =newDSACryptoServiceProvider();
stringpubKey

=

"<DSAKeyValue><P>jmQh1u62F7h3RMkRs+uhoGn4fQMhih8GyZPWmJJHRwcOVxmJC5d5cR0
k7PFf0x0+PvkHr

+wFSTvB2ZregiasXJ8WnHUvEcFCHbJ+iRiKQD1Rd75HfN8o8ViqL36Ia6PM1ZwI2Fqyc0zJshoZ
qg2CQX8cDwO

7T/Ogea+S+5bnrrE=</P><Q>68r+49IOAxNZNJkG43A5I+Ma9hE=</Q><G>g9RWkaYfOs1tOew
wxFHZFgN9NFT
Dz6FW5UWN2bakbfyOOZ2xeveYQ87nJEg5nXv4ZSTo7mxP2AhuL8lFKKmGcPLDgPFwjcWqTAeMTd3x
6zKDb1Ev0
N4VTzzHD0GNLfTfdtQpTYuzzifp+gh2rtkmqF29g8DqOgG1uI9pVYfvYF0=</G><Y>Q6SrbyDo6/T0n8aZr
Wow
e3YJKgEzEZhT7qAY8nZ2O7CvCIM7Y3M8/9DkddrAosk1X4pSye5bmBTERpDB1+I7TlAqS+sOeUx69UL
6OJlRGT
fdai51g9T/qy7nPqwgY/jBQx4UvPjLIsFaovCjjRDLTje9X8Q2i+5c/OSdHN8/mfQ=</Y><J>mpgEUdSn4XqlW
ugO+TKaF4r7WFQkux4wRPRIJHoL/wBgjIY8oA9Ji8RgVOGgb9TI61ll5BTM4mfR/ucCgMHtoKRbi8dOz
Ogoj6h
dT1BWDg4vW23t6v8Up3/APn/tO+ZAOaL26rZuscyaJ1Ow</J><Seed>U1+1KTe5EGYeuLYQojzYeoTfGnk
=</S eed><PgenCounter>Ag==</PgenCounter></DSAKeyValue>";dsa.FromXmlString(pubKey);


//Перевіряється  підписдлядайджесту        string base64Sign =
"mxdOkPaZrDdDnl15sEL8GQim6B4/c2fbd2ksc4ByqxOVS7TA1ouXqg==";

byte[] signature = Convert.FromBase64String(base64Sign);

//Console.WriteLine("---Signature: \n{0}\n", Convert.ToBase64String(signature));

bool match = dsa.VerifyHash(hmessage,"sha1",signature);
Console.WriteLine("-

--Verdict:\n{0}\n", match);


}


//**************************************************************************//
static void Main()

{

//GetSign(); VerifySign();

}

}

}
```