Лабораторная работа № 8

ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ

**Вариант 9**

Воробей Иван Александрович 10701118



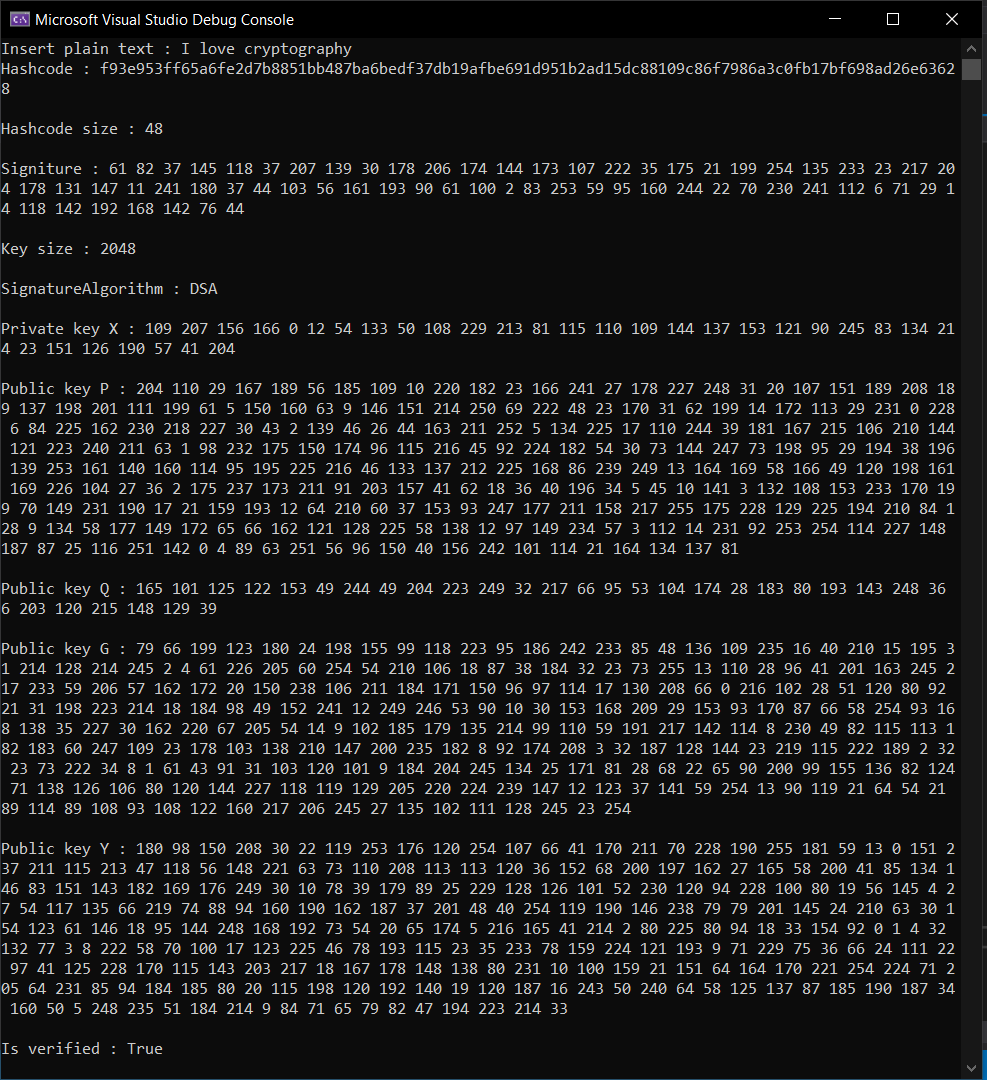
**Цель работы.** Изучить применение электронной цифровой подписи, основанной на криптографическом хэше и асимметричном алгоритме для решения задач аутентификации, контроля целостности и подтверждения обязательств с использованием средств .NETSecurityFramework.

*Я создал class Message который хранит*

* *Текст сообщения*
* *Подпись*
* *«Сертификат»*

*Из метода мейн сначала вызываю метод создающий объект сообщения(он хеширует ввденный в консоли текст, создает подпись). Потом отправляю объект сообщения в метод который верифицирует подпись.*

*Вывод. Научился использывать DSA, протестировал его работу на C#. В моей программе, подпись прошла верификацию(true).*



**Листинг**

using System.Security.Cryptography;

using System.Text;

using static System.Console;

namespace DigitalSignature

{

class Message

{

public string plainText;

public byte[] signature;

public DSA x\_dsa;

public Message(string plainText, byte[] signature, DSA dsa)

{

this.plainText = plainText;

this.signature = signature;

x\_dsa = dsa;

}

}

class EDS

{

static SHA384 x\_sha = SHA384.Create();

static void Main(string[] args)

{

Message message = CreateDS();

bool isVerified = VerifyDS(message);

WriteLine("Is verified : " + isVerified);

}

static Message CreateDS()

{

Write("Insert plain text : ");

string palinText = ReadLine();

DSA x\_dsa = DSA.Create();

byte[] x\_plaintext = Encoding.Default.GetBytes(palinText);

byte[] x\_hashcode = x\_sha.ComputeHash(x\_plaintext);

byte[] x\_signature = x\_dsa.CreateSignature(x\_hashcode);

WriteLine("Hashcode : {0}\n", ByteArrayToString(x\_hashcode));

WriteLine("Hashcode size : {0}\n", x\_hashcode.Length);

WriteLine("Signiture : {0}\n", ArrayToString(x\_signature));

WriteLine("Key size : {0}\n", x\_dsa.KeySize);

WriteLine("SignatureAlgorithm : {0}\n", x\_dsa.SignatureAlgorithm);

WriteLine("Private key X : {0}\n", ArrayToString(x\_dsa.ExportParameters(true).X));

WriteLine("Public key P : {0}\n", ArrayToString(x\_dsa.ExportParameters(false).P));

WriteLine("Public key Q : {0}\n", ArrayToString(x\_dsa.ExportParameters(false).Q));

WriteLine("Public key G : {0}\n", ArrayToString(x\_dsa.ExportParameters(false).G));

WriteLine("Public key Y : {0}\n", ArrayToString(x\_dsa.ExportParameters(false).Y));

return new Message(palinText, x\_signature, x\_dsa);

}

static bool VerifyDS(Message message)

{

byte[] x\_plaintext = Encoding.Default.GetBytes(message.plainText);

byte[] x\_signature = message.signature;

byte[] x\_hashcode = x\_sha.ComputeHash(x\_plaintext);

DSA x\_dsa = message.x\_dsa;

return x\_dsa.VerifySignature(x\_hashcode, x\_signature);

}

static string ArrayToString(byte[] array)

{

StringBuilder sb = new StringBuilder();

foreach(byte b in array)

{

sb.Append(b).Append(" ");

}

return sb.ToString();

}

public static string ByteArrayToString(byte[] ba)

{

StringBuilder hex = new StringBuilder(ba.Length \* 2);

foreach (byte b in ba)

hex.AppendFormat("{0:x2}", b);

return hex.ToString();

}

}

}