МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное автономное бюджетное образовательное учреждение высшего образования

«**Севастопольский государственный университет**»

кафедра Информационных систем

Институт информационных технологий и управления в технических системах

Лисянский Александр Игоревич

курс 4 группа ИС/б-42-о

09.03.02 Информационные системы (уровень бакалавриата)

**ОТЧЁТ**

о лабораторном практикуме №5

СОЗДАНИЕ ЭЛЕКТРОННОЙ ПОДПИСИ В ДОКУМЕНТЕ.

по дисциплине «Основы защиты информации»

Отметка о зачёте \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность) (подпись) (инициалы, фамилия)

Севастополь

2016

**Цель работы**: разработка процедур выработки и проверки электронной цифровой подписи (ЭЦП) сообщений на базе асимметричного криптографического алгоритма с применением функции хеширования

**Ход работы**: В соответствии с вариантом задания был разработан алгоритм выработки электронной подписи сообщений пользователя.

В результате проверки было получено заключение, что программа работает правильно, подпись генерируется и проверяется.

Код программы:

namespace lab5

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

DigitalSignatureHelper ds = new DigitalSignatureHelper();

byte[] hash1;

byte[] hash2;

byte[] signedhash;

private void button3\_Click(object sender, EventArgs e)

{

try //создание подписи

{

MD5HashHelper md5 = new MD5HashHelper();

hash1 = md5.GetHash(textBox3.Text);

signedhash = ds.CreateSignature(hash1);

MessageBox.Show("Подпись успешно создана!");

}

catch(Exception ex){MessageBox.Show(ex.Message);}

}

private void button2\_Click(object sender, EventArgs e)

{

try//проверка подписи

{

MD5HashHelper md5 = new MD5HashHelper();

hash2 = md5.GetHash(textBox2.Text);

if (ds.VerifySignature(hash2, signedhash))

{

MessageBox.Show("Ура! Подпись правильная!");

}

else

{

MessageBox.Show("Ой! Подпись неверная!");

}

}

catch (Exception ex) { MessageBox.Show(ex.Message); }

}

}

}

namespace lab5

{

class DigitalSignatureHelper

{

private RSAParameters m\_public;

public byte[] CreateSignature(byte[] hash)

{

RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();

RSAPKCS1SignatureFormatter RSAFormatter = new RSAPKCS1SignatureFormatter(RSA);

RSAFormatter.SetHashAlgorithm("MD5");

m\_public = RSA.ExportParameters(false);

return RSAFormatter.CreateSignature(hash);

}

public bool VerifySignature(byte[] hash, byte[] signedhash)

{

RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();

RSAParameters RSAKeyInfo = new RSAParameters();

RSAKeyInfo.Modulus = m\_public.Modulus;

RSAKeyInfo.Exponent = m\_public.Exponent;

RSA.ImportParameters(RSAKeyInfo);

RSAPKCS1SignatureDeformatter RSADeformatter = new RSAPKCS1SignatureDeformatter(RSA);

RSADeformatter.SetHashAlgorithm("MD5");

return RSADeformatter.VerifySignature(hash, signedhash);

}

}

}

namespace lab5

{

class MD5HashHelper

{

public byte[] GetHash(string message)

{

byte[] data;

data = UTF8Encoding.ASCII.GetBytes(message);

MD5CryptoServiceProvider md5 = new MD5CryptoServiceProvider();

return md5.ComputeHash(data, 0, data.Length);

}

public bool VerifyHash(string message, byte[] hash)

{

byte[] data;

data = UTF8Encoding.ASCII.GetBytes(message);

MD5CryptoServiceProvider md5 = new MD5CryptoServiceProvider();

byte[] hashTemp = md5.ComputeHash(data, 0, data.Length);

for (Int32 counter = 0; counter <= hash.Length - 1; counter += 1)

{

if (hash[counter] != hashTemp[counter])

{

return false;

}

}

return true;

}

}

}

Вывод: В результате выполнения лабораторной работы были получены навыки по разработке алгоритмов выработки и проверки цифровой подписи сообщений пользователей; написана, отлажена и протестирована программа генерации и проверки электронных подписей.