Министерство образования и науки Российской Федерации

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ФИЗИКИ

КАФЕДРА РАДИОФИЗИКИ

ОТЧЕТ

Криптография

Задача №4

Студент 4 курса,

группы 06-110

А.Ф.Хаматзянова

Казань – 2015

Формулировка задания:

1. Написать программу, вычисляющую с помощью алгоритма MD4 хэш-код строки, введённой пользователем. Убедиться в наличии лавинного эффекта. Для этого в исходной символьной строке M изменить один произвольный бит и получить строку M ` . Вычислить и сравнить хэш-коды h = H(M) и h ` = H(M `). Определить, какое количество битов в h 1 изменилось по сравнению с h.

Код программы:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

using Digests;

namespace md4

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

byte[] data;

byte[] data1;

private void button1\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.Filter =

"Txt files |\*.txt";

openFileDialog1.Title = "Текст для хеширования";

DialogResult dr = openFileDialog1.ShowDialog();

if (dr == System.Windows.Forms.DialogResult.OK)

{

data = File.ReadAllBytes(openFileDialog1.FileName);

}

}

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

{

Digests.MD4Digest d = new Digests.MD4Digest();

d.Reset();

d.BlockUpdate(data, 0, data.Length);

byte[] output =new byte [16];

d.DoFinal(output,0);

string tmp = "";

tmp = tmp + BitConverter.ToString(output);

MessageBox.Show(tmp);

}

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

{

Application.Exit();

}

private void button4\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.Filter =

"Txt files |\*.txt";

openFileDialog1.Title = "Текст для хеширования";

DialogResult dr = openFileDialog1.ShowDialog();

if (dr == System.Windows.Forms.DialogResult.OK)

{

data1 = File.ReadAllBytes(openFileDialog1.FileName);

}

byte [] data2=new byte [data1.Length];

data1.CopyTo(data2, 0);

Digests.MD4Digest d1 = new Digests.MD4Digest();

d1.Reset();

d1.BlockUpdate(data1, 0, data1.Length);

byte[] output1 = new byte [16];

d1.DoFinal(output1, 0);

int tmp = data2[0] + 1;

data2[0] = (byte)tmp;

Digests.MD4Digest d2 = new Digests.MD4Digest();

d2.Reset();

d2.BlockUpdate(data2, 0, data2.Length);

byte[] output2 = new byte [16];

d2.DoFinal(output2, 0);

byte [] result= new byte [16];

for (int i = 0; i < output1.Length; i++)

{

result[i] = (byte)(output1[i] ^ output2[i]);

}

List<int> code = new List<int>();

foreach (var i in result)

{

string BinaryCode = Convert.ToString(i, 2);

int res = Convert.ToInt32(BinaryCode);

code.Add(res);

}

string codetxt = string.Join("", code.ToArray());

Console.WriteLine("Двоичный код исходного текста" + "\n" + codetxt);

int[] digits1 = codetxt.ToString().Select(c => (int)char.GetNumericValue(c)).ToArray();

int sum=0;

for (int i = 0; i < digits1.Length; i++)

{

if (digits1[i] == 1)

sum++;

}

MessageBox.Show("digits\_Count= " + Convert.ToString(digits1.Length) + " " + "1\_Count= " + Convert.ToString(sum));

}

}

}

Функции, использованные при реализации алгоритма

public override void Reset()

{

base.Reset();

H1 = unchecked((int) 0x67452301);

H2 = unchecked((int) 0xefcdab89);

H3 = unchecked((int) 0x98badcfe);

H4 = unchecked((int) 0x10325476);

xOff = 0;

for (int i = 0; i != X.Length; i++)

{

X[i] = 0;

}

}

private int RotateLeft(

int x,

int n)

{

return (x << n) | (int) ((uint) x >> (32 - n));

}

/\*

\* F, G, H and I are the basic MD4 functions.

\*/

private int F(

int u,

int v,

int w)

{

return (u & v) | (~u & w);

}

private int G(

int u,

int v,

int w)

{

return (u & v) | (u & w) | (v & w);

}

private int H(

int u,

int v,

int w)

{

return u ^ v ^ w;

}

internal override void ProcessBlock()

{

int a = H1;

int b = H2;

int c = H3;

int d = H4;

//

// Round 1 - F cycle, 16 times.

//

a = RotateLeft((a + F(b, c, d) + X[ 0]), S11);

d = RotateLeft((d + F(a, b, c) + X[ 1]), S12);

c = RotateLeft((c + F(d, a, b) + X[ 2]), S13);

b = RotateLeft((b + F(c, d, a) + X[ 3]), S14);

a = RotateLeft((a + F(b, c, d) + X[ 4]), S11);

d = RotateLeft((d + F(a, b, c) + X[ 5]), S12);

c = RotateLeft((c + F(d, a, b) + X[ 6]), S13);

b = RotateLeft((b + F(c, d, a) + X[ 7]), S14);

a = RotateLeft((a + F(b, c, d) + X[ 8]), S11);

d = RotateLeft((d + F(a, b, c) + X[ 9]), S12);

c = RotateLeft((c + F(d, a, b) + X[10]), S13);

b = RotateLeft((b + F(c, d, a) + X[11]), S14);

a = RotateLeft((a + F(b, c, d) + X[12]), S11);

d = RotateLeft((d + F(a, b, c) + X[13]), S12);

c = RotateLeft((c + F(d, a, b) + X[14]), S13);

b = RotateLeft((b + F(c, d, a) + X[15]), S14);

//

// Round 2 - G cycle, 16 times.

//

a = RotateLeft((a + G(b, c, d) + X[ 0] + 0x5a827999), S21);

d = RotateLeft((d + G(a, b, c) + X[ 4] + 0x5a827999), S22);

c = RotateLeft((c + G(d, a, b) + X[ 8] + 0x5a827999), S23);

b = RotateLeft((b + G(c, d, a) + X[12] + 0x5a827999), S24);

a = RotateLeft((a + G(b, c, d) + X[ 1] + 0x5a827999), S21);

d = RotateLeft((d + G(a, b, c) + X[ 5] + 0x5a827999), S22);

c = RotateLeft((c + G(d, a, b) + X[ 9] + 0x5a827999), S23);

b = RotateLeft((b + G(c, d, a) + X[13] + 0x5a827999), S24);

a = RotateLeft((a + G(b, c, d) + X[ 2] + 0x5a827999), S21);

d = RotateLeft((d + G(a, b, c) + X[ 6] + 0x5a827999), S22);

c = RotateLeft((c + G(d, a, b) + X[10] + 0x5a827999), S23);

b = RotateLeft((b + G(c, d, a) + X[14] + 0x5a827999), S24);

a = RotateLeft((a + G(b, c, d) + X[ 3] + 0x5a827999), S21);

d = RotateLeft((d + G(a, b, c) + X[ 7] + 0x5a827999), S22);

c = RotateLeft((c + G(d, a, b) + X[11] + 0x5a827999), S23);

b = RotateLeft((b + G(c, d, a) + X[15] + 0x5a827999), S24);

//

// Round 3 - H cycle, 16 times.

//

a = RotateLeft((a + H(b, c, d) + X[ 0] + 0x6ed9eba1), S31);

d = RotateLeft((d + H(a, b, c) + X[ 8] + 0x6ed9eba1), S32);

c = RotateLeft((c + H(d, a, b) + X[ 4] + 0x6ed9eba1), S33);

b = RotateLeft((b + H(c, d, a) + X[12] + 0x6ed9eba1), S34);

a = RotateLeft((a + H(b, c, d) + X[ 2] + 0x6ed9eba1), S31);

d = RotateLeft((d + H(a, b, c) + X[10] + 0x6ed9eba1), S32);

c = RotateLeft((c + H(d, a, b) + X[ 6] + 0x6ed9eba1), S33);

b = RotateLeft((b + H(c, d, a) + X[14] + 0x6ed9eba1), S34);

a = RotateLeft((a + H(b, c, d) + X[ 1] + 0x6ed9eba1), S31);

d = RotateLeft((d + H(a, b, c) + X[ 9] + 0x6ed9eba1), S32);

c = RotateLeft((c + H(d, a, b) + X[ 5] + 0x6ed9eba1), S33);

b = RotateLeft((b + H(c, d, a) + X[13] + 0x6ed9eba1), S34);

a = RotateLeft((a + H(b, c, d) + X[ 3] + 0x6ed9eba1), S31);

d = RotateLeft((d + H(a, b, c) + X[11] + 0x6ed9eba1), S32);

c = RotateLeft((c + H(d, a, b) + X[ 7] + 0x6ed9eba1), S33);

b = RotateLeft((b + H(c, d, a) + X[15] + 0x6ed9eba1), S34);

H1 += a;

H2 += b;

H3 += c;

H4 += d;

xOff = 0;

for (int i = 0; i != X.Length; i++)

{

X[i] = 0;

}

}

public override int DoFinal(

byte[] output,

int outOff)

{

Finish();

UnpackWord(H1, output, outOff);

UnpackWord(H2, output, outOff + 4);

UnpackWord(H3, output, outOff + 8);

UnpackWord(H4, output, outOff + 12);

Reset();

return DigestLength;

}

internal override void ProcessWord(

byte[] input,

int inOff)

{

X[xOff++] = (input[inOff] & 0xff) | ((input[inOff + 1] & 0xff) << 8)

| ((input[inOff + 2] & 0xff) << 16) | ((input[inOff + 3] & 0xff) << 24);

if (xOff == 16)

{

ProcessBlock();

}

}

internal override void ProcessLength(

long bitLength)

{

if (xOff > 14)

{

ProcessBlock();

}

X[14] = (int)(bitLength & 0xffffffff);

X[15] = (int)((ulong) bitLength >> 32);

}

private void UnpackWord(

int word,

byte[] outBytes,

int outOff)

{

outBytes[outOff] = (byte)word;

outBytes[outOff + 1] = (byte)((uint) word >> 8);

outBytes[outOff + 2] = (byte)((uint) word >> 16);

outBytes[outOff + 3] = (byte)((uint) word >> 24);

}

Результат работы программы:

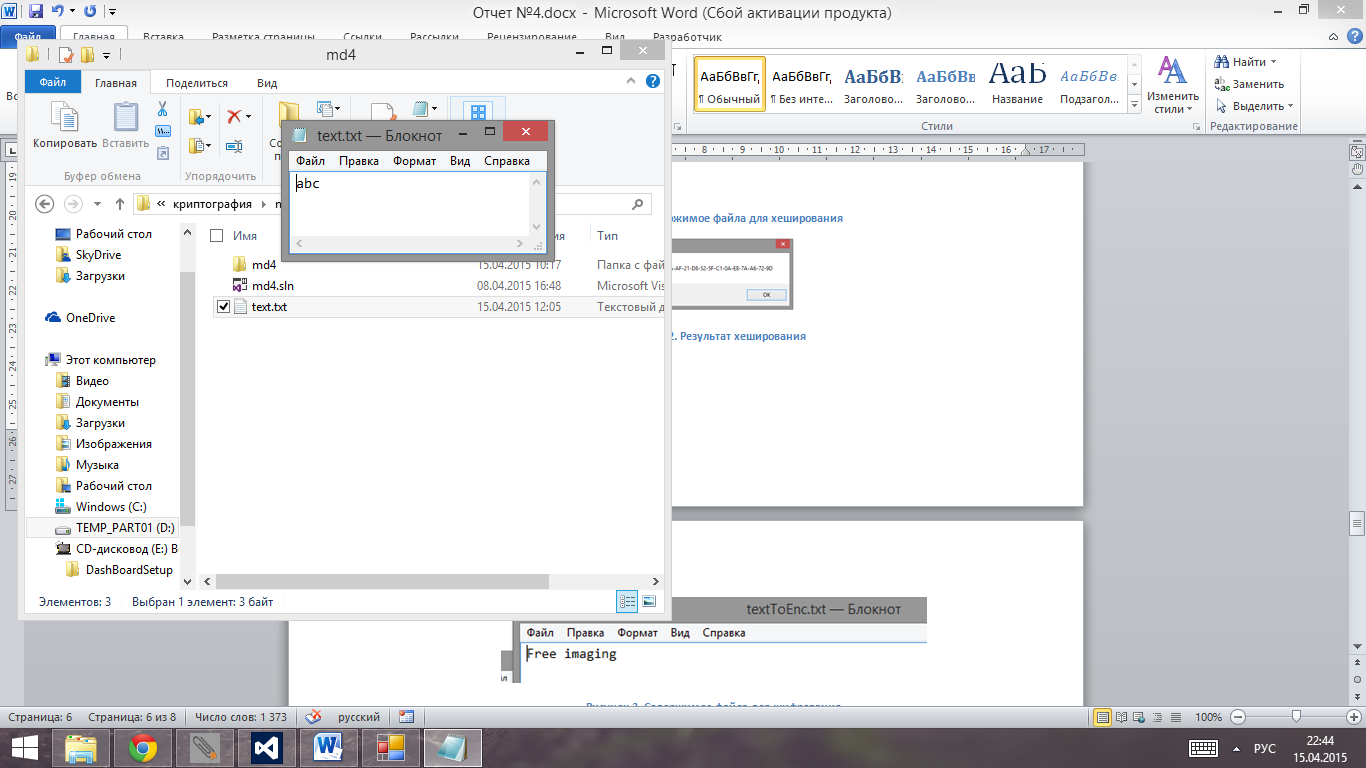


Рисунок . Содержимое файла для хеширования

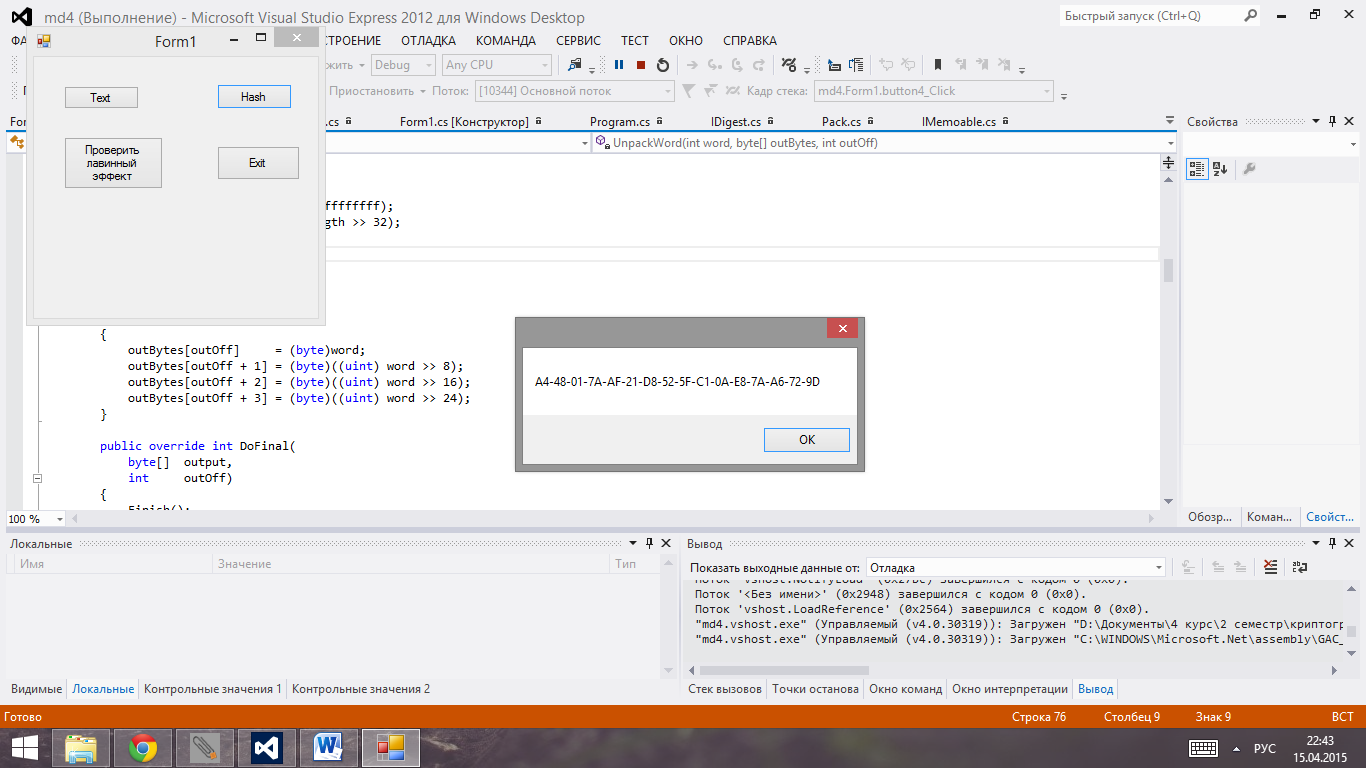


Рисунок . Результат хеширования

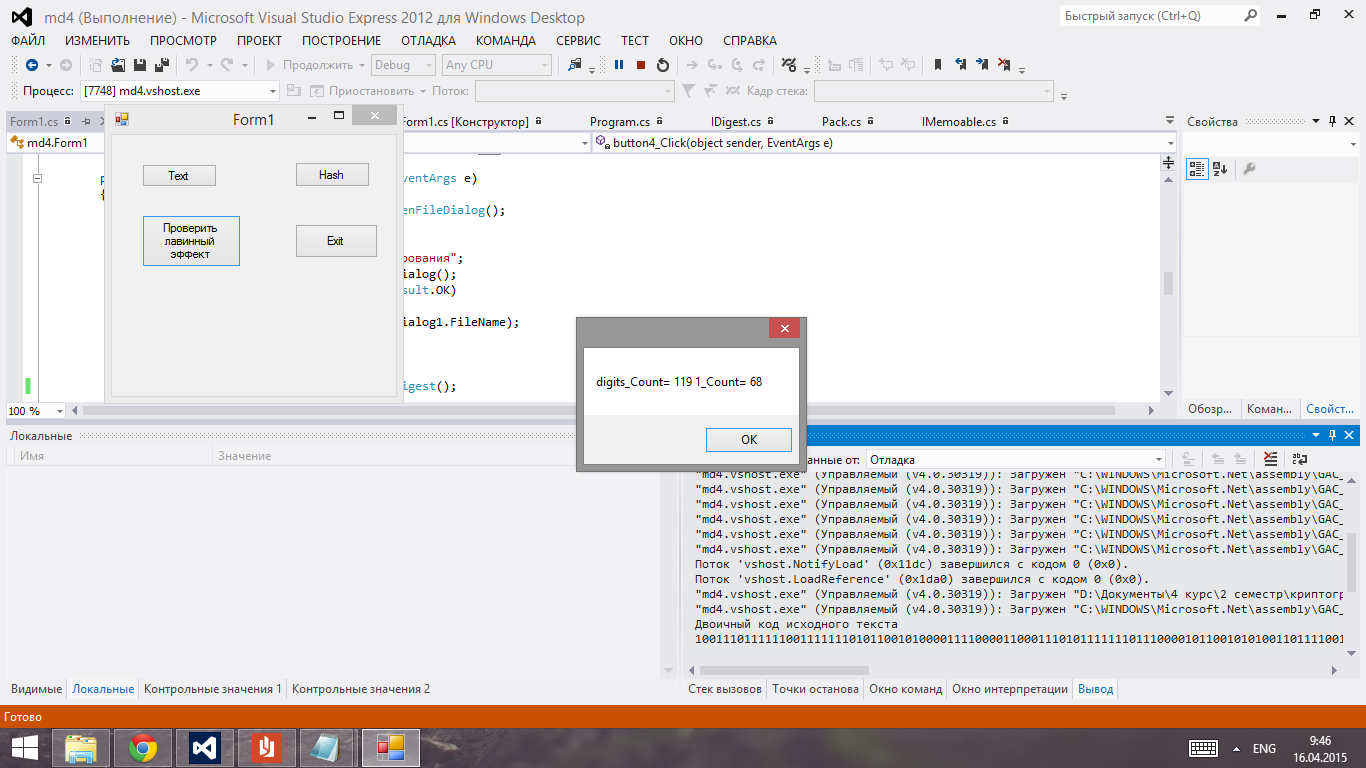


Рисунок . Результат проверки лавинного эффекта