Министерство образования и науки Российской Федерации

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ФИЗИКИ

КАФЕДРА РАДИОФИЗИКИ

ОТЧЕТ

Криптография

Задача №2

Студент 4 курса,

группы 06-110

А.Ф.Хаматзянова

Казань – 2015

Формулировка задания:

1. Написать программу, реализующую генератор M-последовательности на основе полинома обратной связи, выданного преподавателем. Начальные значения битов регистра сдвига задаются случайным образом (с помощью стандартного генератора случайных чисел). В программе должна быть возможность сохранять начальное состояние регистра сдвига в файле key.txt, а также возможность загружать начальное состояние регистра из этого же файла. Предусмотреть возможность вывода первых n двоичных элементов М-последовательности на экран (n задаётся пользователем).
2. Протестировать М-последовательность длиной N бит (N ≤ 10000, задаётся преподавателем) с помощью сериального теста. Использовать серии длиной 2, 3 или 4 бита (по указанию преподавателя).
3. Протестировать М-последовательность с помощью корреляционного теста для k = 1, 2, 8, 9.
4. Реализовать программу шифрования/расшифрования двоичного файла с помощью М-последовательности. В качестве ключа использовать начальное значение регистра сдвига, которое хранится в файле key.txt. Результат шифрования сохранить в файле encoded.txt. Расшифровать файл encoded.txt с помощью того же ключа и убедиться, что расшифрованный и исходный файлы идентичны.
5. Протестировать исходный и зашифрованный файлы с помощью статистических тестов (Задание 2, 3).

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;

namespace second

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private int generate\_M (int [] bits)

{

int tmp = plus(bits[30],bits[5]);

int result = bits[0];

for (int i = 0; i < 30; i++)

{

bits[i]=bits[i+1];

}

bits[30] = tmp;

return result;

}

private int plus(int x1, int x2)

{

if ((x1 == 0) && (x2 == 0))

return 0;

else

{

if ((x1 == 0) && (x2 == 1))

return 1;

else

{

if ((x1 == 1) && (x2 == 1))

return 0;

else

return 1;

}

}

}

private List<byte> FromBinToDec(List<int> M)

{

List<byte> M\_b = new List<byte>();

int k = 0;

while (k < M.Count())

{

int sum = 0;

for (int j = 0; j < 8; j++)

{

sum = sum + M[k] \* (int)Math.Pow(2, j);

k++;

}

byte tmpByte = Convert.ToByte(sum.ToString(), 10);

M\_b.Add(tmpByte);

}

return M\_b;

}

private void SerialTest(List<int> M, int N, int k)

{

double N\_T = k / Math.Pow(2, N);

int St\_Sv = (int)Math.Pow(2, N) - 1;

List<string> N\_E = new List<string>();

List<int> N\_E\_Freq = new List<int>();

int i = 0;

if (k \* N == Length\_M)

{

for (i = 0; i < Length\_M - N + 1; i++)

{

int p = N;

string tmp = "";

for (int j = i; j < i + N; j++)

{

tmp = tmp + Convert.ToString(M[j]);

p--;

}

N\_E.Add(tmp);

i = i + N - 1;

}

}

else

{

for (i = 0; i < N \* (k - 1) + 1; i++)

{

int p = N;

string tmp = "";

for (int j = i; j < i + N; j++)

{

tmp = tmp + Convert.ToString(M[j]);

p--;

}

N\_E.Add(tmp);

i = i + N - 1;

}

}

N\_E.Sort();

i = 0;

while (i != N\_E.Count())

{

int first\_index = N\_E.IndexOf(N\_E[i]);

int last\_index = N\_E.LastIndexOf(N\_E[i]);

N\_E\_Freq.Add(last\_index - first\_index + 1);

i = last\_index + 1;

}

double Pirs = 0;

for (i = 0; i < N\_E\_Freq.Count(); i++)

{

Pirs = Pirs + (Math.Pow(((double)N\_E\_Freq[i] - N\_T), 2)) / N\_T;

}

if (St\_Sv == 3)

{

if ((0.584 < Pirs) && (Pirs < 6.251))

{

richTextBox2.Text = "Сериальный тест пройден успешно";

}

else

richTextBox2.Text = "Сериальный тест не пройден";

}

else

{

if (St\_Sv == 7)

{

if ((2.833 < Pirs) && (Pirs < 12.017))

{

richTextBox2.Text = "Сериальный тест пройден успешно";

}

else

richTextBox2.Text = "Сериальный тест не пройден";

}

else

{

if (St\_Sv == 15)

{

if ((8.547 < Pirs) && (Pirs < 22.307))

{

richTextBox2.Text = "Сериальный тест пройден успешно";

}

else

richTextBox2.Text = "Сериальный тест не пройден";

}

}

}

}

private void CorrelationTest(List<int> M, int k)

{

int Sx = 0;

int Sy = 0;

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

for (int i = k; i < Length\_M; i++)

{

y.Add(M[i]);

}

for (int i = 0; i < k; i++)

{

y.Add(M[i]);

}

for (int i = 0; i < Length\_M; i++)

{

Sx = Sx + M[i];

}

for (int j = 0; j < Length\_M; j++)

{

Sy = Sy + y[j];

}

int Sxy = 0;

for (int i = 0; i < Length\_M; i++)

{

Sxy = Sxy + M[i] \* y[i];

}

int Sx2 = 0;

for (int i = 0; i < Length\_M; i++)

{

Sx2 = Sx2 + M[i] \* M[i];

}

int Sy2 = 0;

for (int j = 0; j < Length\_M; j++)

{

Sy2 = Sy2 + y[j] \* y[j];

}

double R1 = (double)(Length\_M \* Sxy - Sx \* Sy);

double R2 = Math.Sqrt(((double)(Length\_M \* Sx2 - Sx \* Sx)) \* (Length\_M \* Sy2 - Sy \* Sy));

double R = R1 / R2;

double R\_T = ((double)(1)) / ((double)(Length\_M - 1) + (2 / (Length\_M - 2)) \* Math.Sqrt((Length\_M \* (Length\_M - 3)) / (Length\_M + 1)));

if ((-R\_T >= R) && (R <= R\_T))

{

richTextBox3.Text = "Корреляционный тест пройден успешно";

}

else

{

richTextBox3.Text = "Корреляционный тест не пройден";

}

}

private List<byte> EncryptDecrypt(List <byte> M\_b,List <byte> textToEnc)

{

List<byte> resOfEnc = new List<byte>();

for (int i = 0; i < textToEnc.Count; i++)

{

byte tmp = (byte)(textToEnc[i] ^ M\_b[i]);

resOfEnc.Add(tmp);

}

return resOfEnc;

}

private List<int> FromDecToBin(List<byte> TextT)

{

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

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

foreach (var i in TextT)

{

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

int result = Convert.ToInt32(BinaryCode);

code.Add(result);

}

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

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

Bits = digits1.ToList();

return Bits;

}

int[] bits;

List <byte> textToDec=new List<byte>();

List<byte> textToEnc=new List<byte>();

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

List<byte> X=new List<byte>();

List<byte> M\_b = new List<byte>();

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

int Length\_M;

int n;

int N;

//генерация М-последовательности

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

{

Length\_M = Convert.ToInt32(textBox2.Text);

for (int i = 0; i < 62; i++)

{

int tmp= generate\_M(bits);

}

for (int i = 0; i < Length\_M; i++)

{

M.Add(generate\_M(bits));

}

}

//случайная инициализация

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

{

bits = new int[31];

Random rand = new Random();

// начальная инициализация

for (int i = 0; i < 31; i++)

{

bits[i] = rand.Next(0, 2);

}

// Папка для начальной инициализации

string Dir = "";

FolderBrowserDialog folderBrowserDialog1 = new FolderBrowserDialog();

//folderBrowserDialog1.RootFolder = Environment.SpecialFolder.Personal;

folderBrowserDialog1.ShowNewFolderButton = true;

DialogResult dialogresult = folderBrowserDialog1.ShowDialog();

folderBrowserDialog1.Description = "Выбор папки для записи файла с начальной инициализацией";

if (dialogresult == DialogResult.OK)

{

Dir = folderBrowserDialog1.SelectedPath;

}

//запись начальной инициализации в файл

using (StreamWriter writer = new StreamWriter(File.Open(Path.Combine(Dir, "key.txt"), FileMode.Create)))

{

for (int i = 0; i < 31; i++)

{

writer.Write(bits[i]);

}

writer.Close();

}

}

//инициализация из файла

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

{

bits = new int[31];

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.Filter =

"Txt files |\*.txt";

openFileDialog1.Title = "Начальная инициализация";

DialogResult dr = openFileDialog1.ShowDialog();

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

{

using (StreamReader reader = new StreamReader(File.Open(openFileDialog1.FileName,FileMode.Open)))

{

for (int i = 0; i < 31; i++)

{

bits[i] = Math.Abs(48-reader.Read());

}

reader.Close();

}

}

}

// кнопка выхода из программы

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

{

Application.Exit();

}

// вывод n - первых значений на экран

private void textBox1\_TextChanged(object sender, EventArgs e)

{

n = Convert.ToInt32(textBox1.Text);

string tmp = "";

for (int i = 0; i < n; i++)

{

tmp = tmp + Convert.ToString(M[i]);

}

richTextBox1.Text = tmp;

}

//сериальный тест

private void button5\_Click(object sender, EventArgs e)

{

N=Convert.ToInt32(textBox3.Text);

int k = Length\_M / N;

//сериальный тест

SerialTest(M, N, k);

}

//корреляционный тест

private void button6\_Click(object sender, EventArgs e)

{

int k =Convert.ToInt32(textBox4.Text);

//корреляционный тест

CorrelationTest(M,k);

}

//чтение файла для кодирования и генерация М-последовательности

private void button7\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog2 = new OpenFileDialog();

openFileDialog2.Filter =

"Txt files |\*.txt";

openFileDialog2.Title = "Текст для шифрования";

DialogResult dr = openFileDialog2.ShowDialog();

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

{

byte[] array = File.ReadAllBytes(openFileDialog2.FileName);

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

{

textToEnc.Add(array[i]);

}

}

Length\_M = textToEnc.Count()\*8;

for (int i = 0; i < 62; i++)

{

int z = generate\_M(bits);

}

for (int i = 0; i < Length\_M; i++)

{

M.Add(generate\_M(bits));

}

M\_b = FromBinToDec(M);

}

//шифрование

private void button8\_Click(object sender, EventArgs e)

{

//шифрование и дешифрование

List<byte> resOfEnc = new List<byte>();

resOfEnc = EncryptDecrypt(M\_b,textToEnc);

//запись результата в текстовый файл и выход сообщения об этом

string Dir = "";

FolderBrowserDialog folderBrowserDialog2 = new FolderBrowserDialog();

folderBrowserDialog2.ShowNewFolderButton = true;

DialogResult dialogresult = folderBrowserDialog2.ShowDialog();

folderBrowserDialog2.Description = "Выбор папки для записи результата шифрования";

if (dialogresult == DialogResult.OK)

{

Dir = folderBrowserDialog2.SelectedPath;

}

byte[] resEnc = resOfEnc.ToArray();

char[] chars = Encoding.Default.GetChars(resEnc);

using (StreamWriter writer = new StreamWriter(Path.Combine(Dir, "encoded.txt"), false, Encoding.Default))

{

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

{

writer.Write(chars[i]);

}

writer.Close();

}

richTextBox4.Text = "Результат шифрования записан в выбранную папку в файл /encoded.txt/ ";

}

//ключ для дешифрации

private void button9\_Click(object sender, EventArgs e)

{

bits = new int[31];

OpenFileDialog openFileDialog3 = new OpenFileDialog();

openFileDialog3.Filter =

"Txt files |\*.txt";

openFileDialog3.Title = "Ключ для дешифрации";

DialogResult dr = openFileDialog3.ShowDialog();

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

{

using (StreamReader reader = new StreamReader(File.Open(openFileDialog3.FileName, FileMode.Open)))

{

for (int i = 0; i < 31; i++)

{

bits[i] = Math.Abs(48 - reader.Read());

}

reader.Close();

}

}

}

//чтение файла для декодирования и генерация М-последовательности

private void button10\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog2 = new OpenFileDialog();

openFileDialog2.Filter =

"Txt files |\*.txt";

openFileDialog2.Title = "Текст для шифрования";

DialogResult dr = openFileDialog2.ShowDialog();

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

{

byte[] array = File.ReadAllBytes(openFileDialog2.FileName);

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

{

textToDec.Add(array[i]);

}

}

Length\_M = textToDec.Count()\*8;

for (int i = 0; i < 62; i++)

{

int z = generate\_M(bits);

}

for (int i = 0; i < Length\_M; i++)

{

M.Add(generate\_M(bits));

}

M\_b = FromBinToDec(M);

}

//дешифрование

private void button11\_Click(object sender, EventArgs e)

{

//перевести байты в инт

List<byte> resOfDec = new List<byte>();

resOfDec = EncryptDecrypt(M\_b,textToDec);

//запись результата в текстовый файл и выход сообщения об этом

string Dir = "";

FolderBrowserDialog folderBrowserDialog2 = new FolderBrowserDialog();

folderBrowserDialog2.ShowNewFolderButton = true;

DialogResult dialogresult = folderBrowserDialog2.ShowDialog();

folderBrowserDialog2.Description = "Выбор папки для записи результата дешифрования";

if (dialogresult == DialogResult.OK)

{

Dir = folderBrowserDialog2.SelectedPath;

}

byte[] resDec = resOfDec.ToArray();

char[] chars = Encoding.Default.GetChars(resDec);

using (StreamWriter writer = new StreamWriter(Path.Combine(Dir, "decoded.txt"), false,Encoding.Default))

{

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

{

writer.Write(chars[i]);

}

writer.Close();

}

richTextBox5.Text = "Результат расшифрования записан в выбранную папку в файл /decoded.txt/ ";

}

//сериальный тест для текстового файла

private void button12\_Click(object sender, EventArgs e)

{

N = Convert.ToInt32(textBox5.Text);

List<byte> Text = new List<byte>();

OpenFileDialog openFileDialog2 = new OpenFileDialog();

openFileDialog2.Filter =

"Txt files |\*.txt";

openFileDialog2.Title = "Текст для сериального теста";

DialogResult dr = openFileDialog2.ShowDialog();

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

{

byte[] array = File.ReadAllBytes(openFileDialog2.FileName);

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

{

Text.Add(array[i]);

}

}

TextToTest = FromDecToBin(Text);

int k = TextToTest.Count / N;

//сериальный тест

SerialTest(TextToTest, N, k);

}

//корреляционный тест для текстового файла

private void button13\_Click(object sender, EventArgs e)

{

int k = Convert.ToInt32(textBox6.Text);

List<byte> Text = new List<byte>();

OpenFileDialog openFileDialog2 = new OpenFileDialog();

openFileDialog2.Filter =

"Txt files |\*.txt";

openFileDialog2.Title = "Текст для сериального теста";

DialogResult dr = openFileDialog2.ShowDialog();

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

{

byte[] array = File.ReadAllBytes(openFileDialog2.FileName);

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

{

Text.Add(array[i]);

}

}

TextToTest = FromDecToBin(Text);

CorrelationTest(TextToTest, k);

}

}

}

Результат работы программы:

Результатом работы программы являются 3 файла:

* key.txt – ключ;
* encoded.txt – зашифрованный текст;
* decoded.txt – расшифрованный текст.

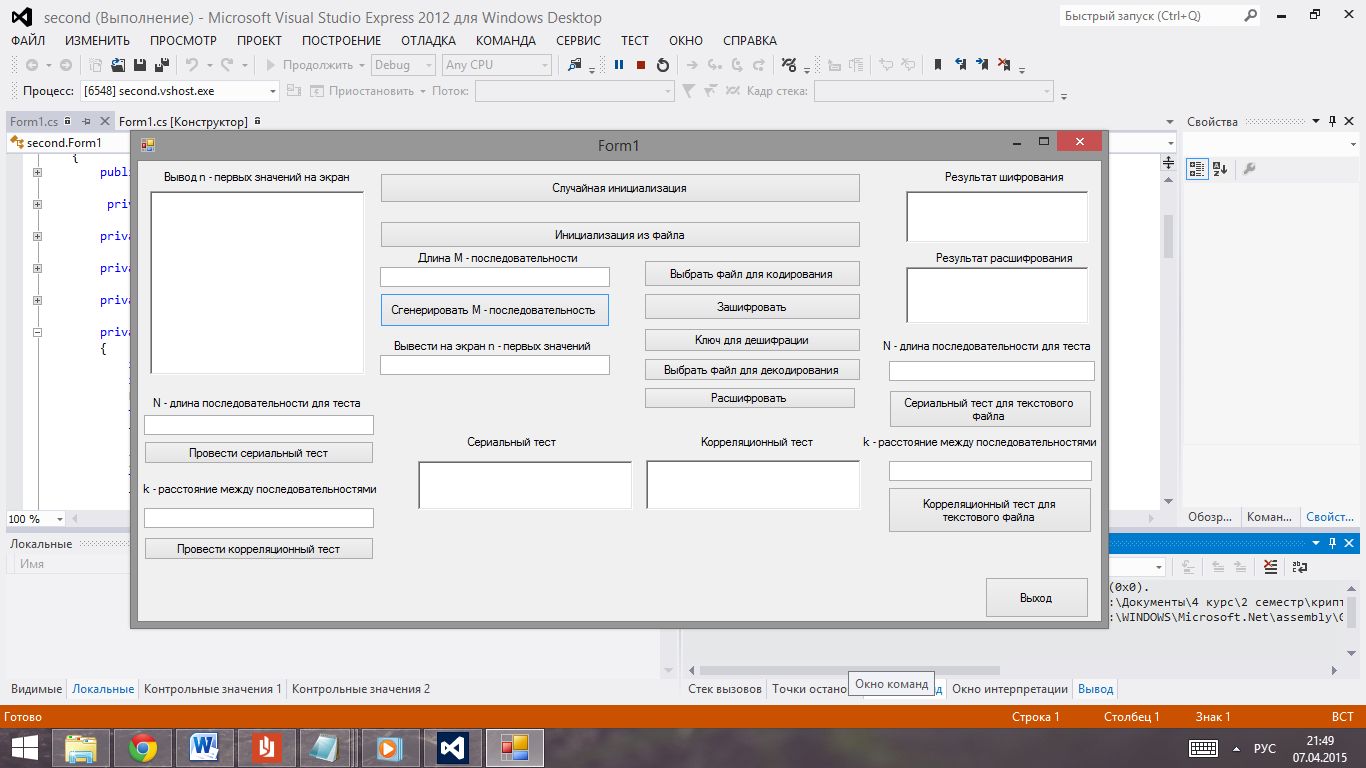


Рисунок 1. Окно запуска программы

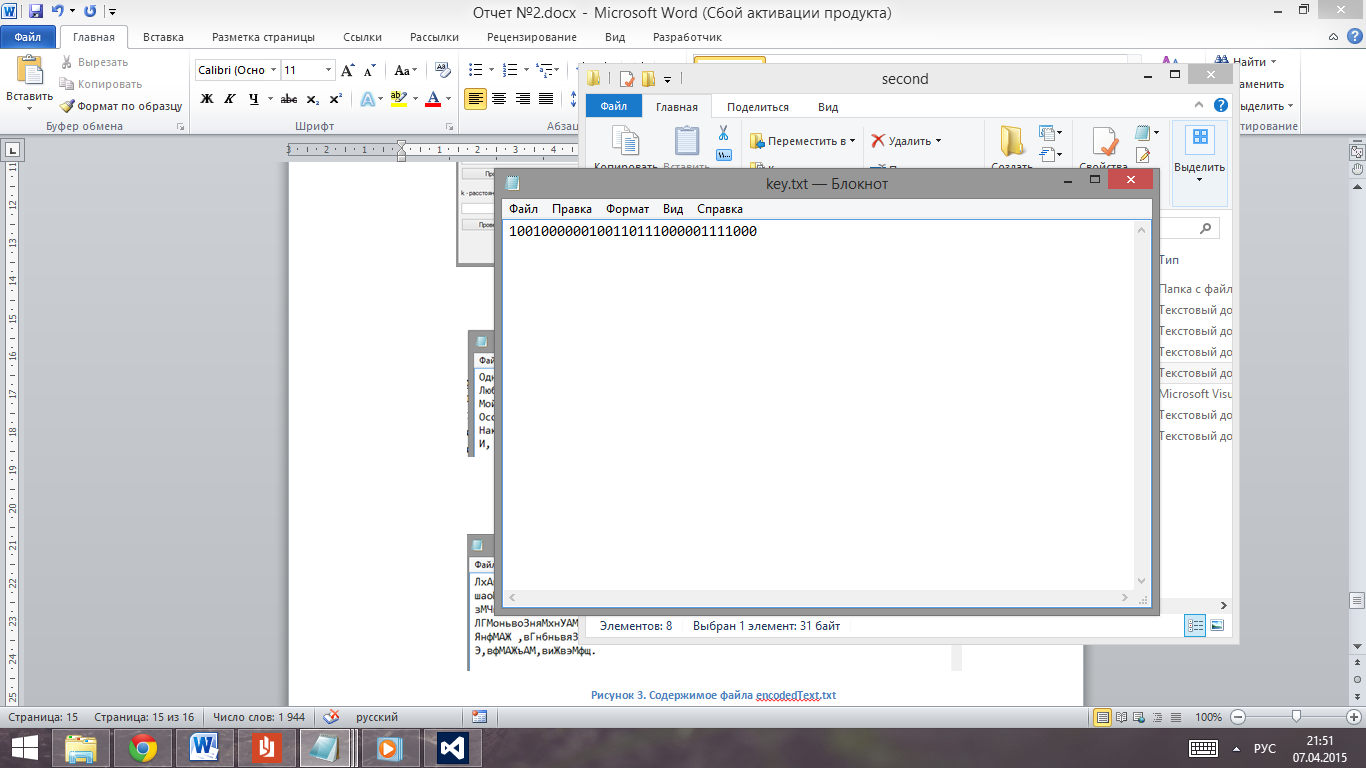


Рисунок . Содержимое файла key.txt

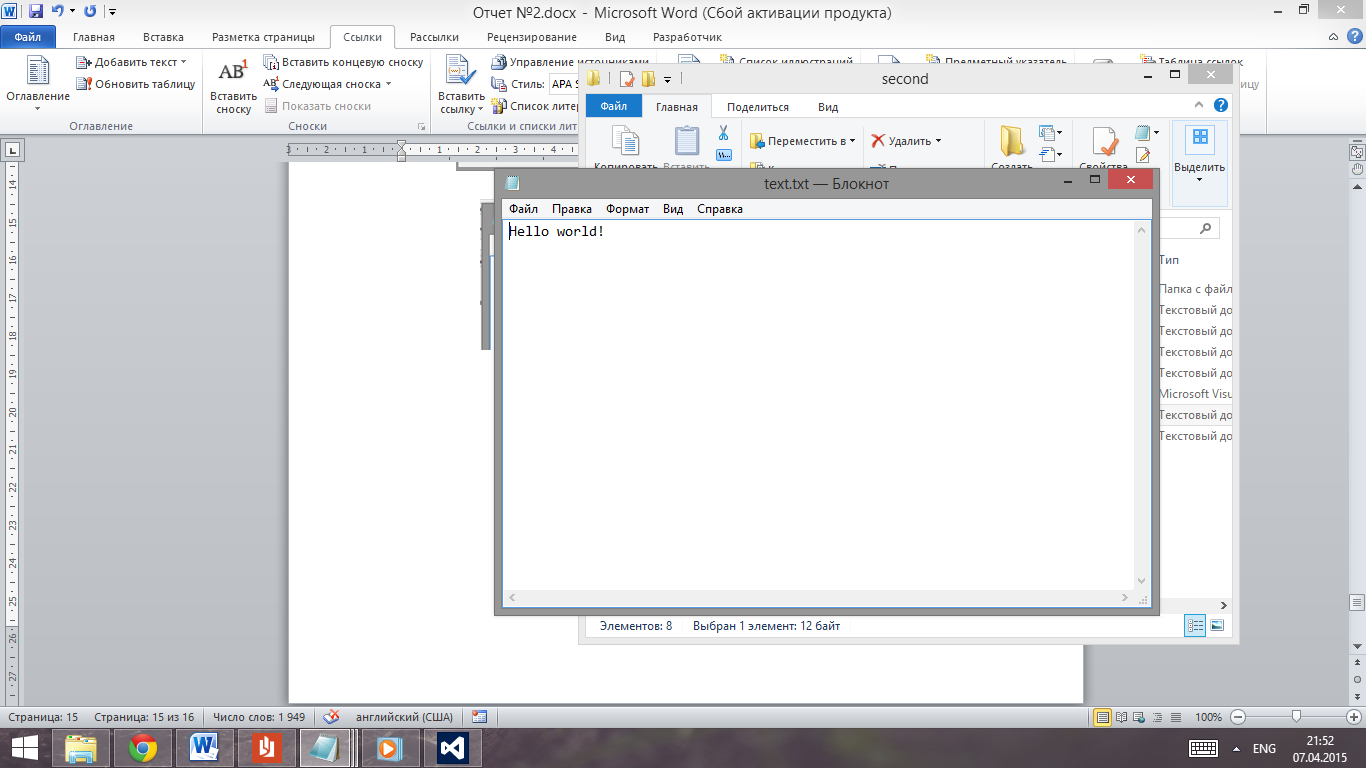


Рисунок 3. Содержимое файла для шифрования

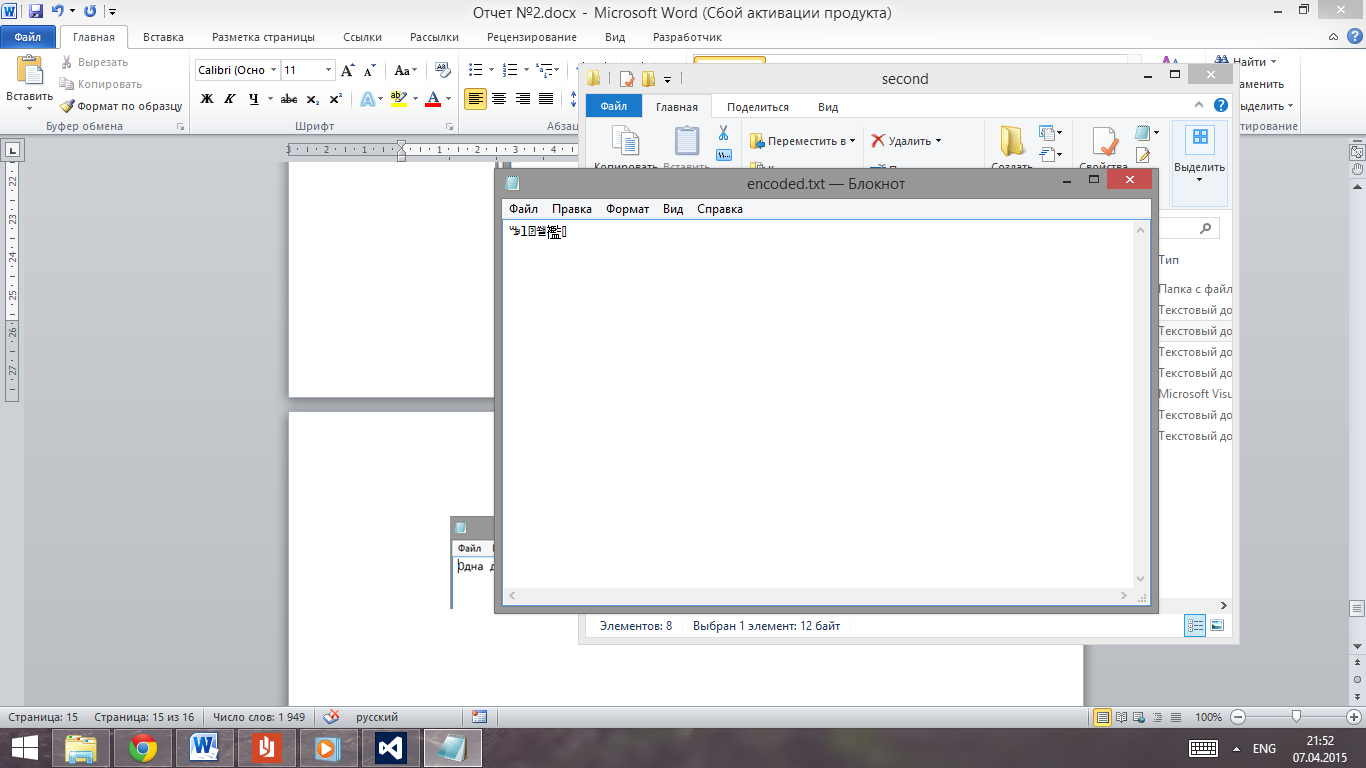


Рисунок 4. Содержимое файла encoded.txt

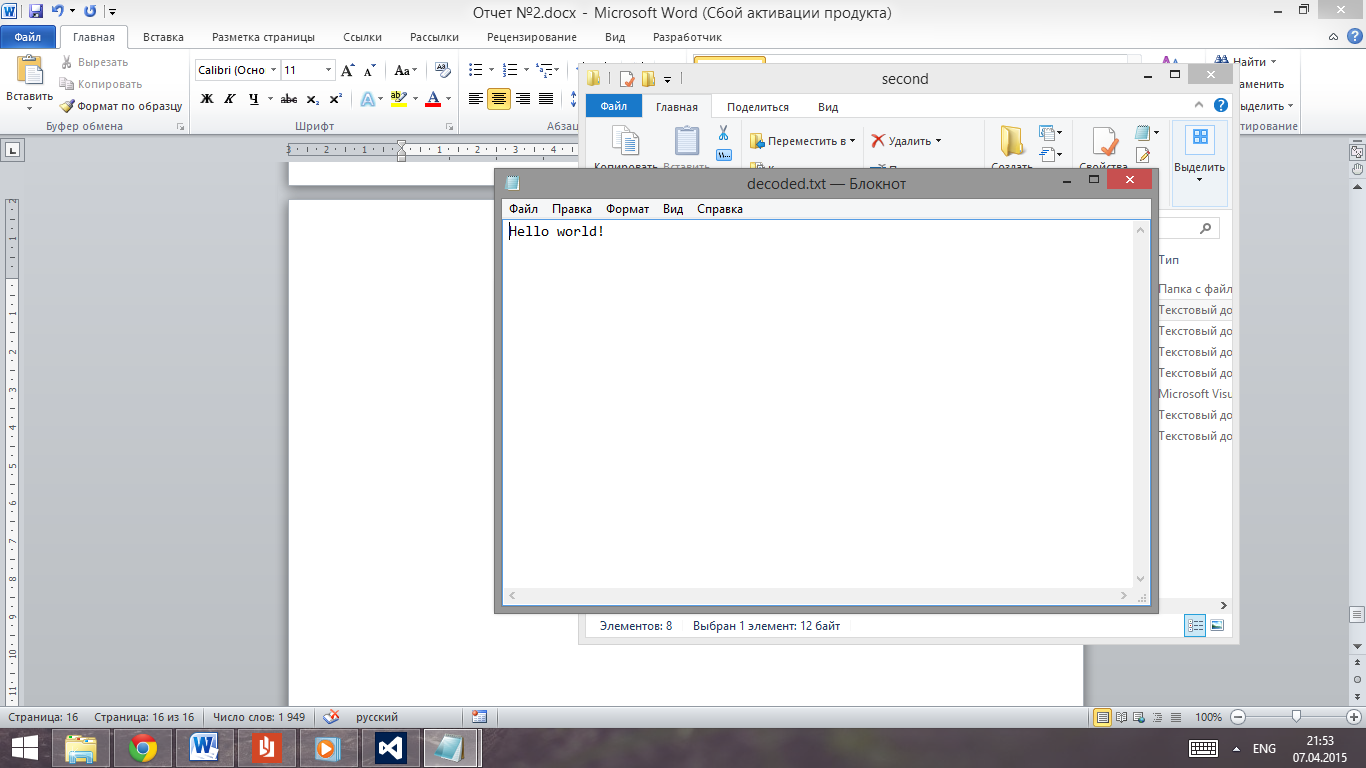


Рисунок 5. Содержимое файла decoded.txt