Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

СИММЕТРИЧНЫЕ АЛГОРИТМЫ ШИФРОВАНИЯ

Отчет по лабораторной работе №2

По дисциплине

«Информационная безопасность»

Студент гр. 431-3

_____ Д.П. Андреев

«___»_____ 2024 г.

Проверил: старший преподаватель кафедры АСУ.

_____Я.В.Яблонский

«___»_____ 2024 г.

Томск 2024

1 Цель работы

Познакомиться и научиться работать с симметричными алгоритмами шифрования.

2 Задание на лабораторную работу

Вариант 1. Два друга хотят обмениваться зашифрованными сообщениями, но у них нет подходящей программы. Напишите программу позволяющую шифровать и расшифровывать сообщения с использованием алгоритма симметричного шифрования TEA. Входные и выходные данные запишите в файл типа .txt.

3 Описание алгоритма шифрования

TEA – это блочный шифр. Он использует 128-битный ключ и работает с 64-битными блоками данных. Сначала происходит инициализация ключа. 128 битный ключ, делим на четыре 32-битных блока (K0, K1, K2, K3). После данные которые мы будем шифровать, разбиваем на 64-битные блоки, после каждый блок делим на два 32-битных блока (t0, t1). Алгоритм проходит 32 раунда, где каждый раунд включает в себя сложение, сдвиги, XOR-операции для обновления значений t0 и t1 и использование ключевых слов для перемешивания данных. На каждом шаге увеличиваем значение sum на фиксированное значение Delta, которое равно 0x9E3779B9. Это значение происходит от золотого сечения и используется для обеспечения диффузии в шифре.

Процесс расшифровки является обратным процессом шифрования. Используется та же логика, но в обратном порядке. Для этого необходимо начать с sum, инициализировав его значением Delta умножить на 32. Алгоритм проходит 32 раунда, выполняя аналогичные операции, но вычитая вместо сложения.

4 Листинг программы

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

namespace ИБ_2
{
   class Program
   {
      const uint Delta = 0x9E3779B9;
      public static uint[] Encryption(uint[] text, uint[] key)
      {
         uint t0 = text[0], t1 = text[1];
         uint k0 = key[0], k1 = key[1], k2 = key[2], k3 = key[3];
```

```
    uint sum = 0;
    for (int i = 0; i < 32; i++)
    {
        sum += Delta;
        t0 += ((t1 << 4) + k0) ^ (t1 + sum) ^ ((t1 >> 5) + k1);
        t1 += ((t0 << 4) + k2) ^ (t0 + sum) ^ ((t0 >> 5) + k3);
    }
    return new uint[] { t0, t1 };
}
public static uint[] Decryption(uint[] text, uint[] key)
{
    uint t0 = text[0], t1 = text[1];
    uint k0 = key[0], k1 = key[1], k2 = key[2], k3 = key[3];
    ulong sumL = (ulong)Delta * 32;

    uint sum = (uint)sumL;
    for (int i = 0; i < 32; i++)
    {
        t1 -= ((t0 << 4) + k2) ^ (t0 + sum) ^ ((t0 >> 5) + k3);
        t0 -= ((t1 << 4) + k0) ^ (t1 + sum) ^ ((t1 >> 5) + k1);
        sum -= Delta;
    }
    return new uint[] { t0, t1 };
}

static void Main(string[] args)
{
    //приём сообщения
    byte[] bytesMessage;
    using (FileStream MessageFile = new FileStream("IN.txt", FileMode.Open))
    using (BinaryReader reader = new BinaryReader(MessageFile))
    {
        //Console.WriteLine(reader.CurrentEncoding.WebName); //вывод формата входного текста
        //string Message = reader.ReadToEnd();
        //bytesMessage = Encoding.UTF8.GetBytes(Message);
        bytesMessage = reader.ReadBytes(reader.PeekChar());
    }
    if (bytesMessage.Length % 8 != 0)
    {
        Array.Resize(ref bytesMessage, (bytesMessage.Length / 8 + 1) * 8);
    }
    //приём сообщения

    //приём ключа
    uint[] key = new uint[4];
    using (FileStream KeyFile = new FileStream("KEY.txt", FileMode.Open))
    using (StreamReader reader = new StreamReader(KeyFile))
    {
        string keyString = reader.ReadToEnd();
        for (int i = 0; i < 4; i++)
        {
            key[i] = Convert.ToUInt32(keyString.Substring(i * 8, 8), 16);
        }
    }
    //приём ключа

    //шифровка и расшифровка сообщения
    using (FileStream EncryptionFile = new FileStream("ENCRYPTION.txt", FileMode.Create))
    using (StreamWriter encryptedWriter = new StreamWriter(EncryptionFile))
    using (FileStream DecryptionFile = new FileStream("OUT.txt", FileMode.Create))
    using (BinaryWriter decryptedWriter = new BinaryWriter(DecryptionFile))
    {
```

```
            for (int i = 0; i < bytesMessage.Length; i += 8)
            {
                //Разбиение
                if (i + 4 >= bytesMessage.Length) break; // Защита от выхода за границы
                uint[] t = new uint[2];
                t[0] = BitConverter.ToUInt32(bytesMessage, i);
                t[1] = BitConverter.ToUInt32(bytesMessage, i + 4);

                // Шифрование
                uint[] encrypted = Encryption(t, key);
                encryptedWriter.Write("{0:X}{1:X}", encrypted[0], encrypted[1]);

                // Расшифровка
                uint[] decrypted = Decryption(encrypted, key);
                byte[] decryptedBytes = new byte[8];
                BitConverter.GetBytes(decrypted[0]).CopyTo(decryptedBytes, 0);
                BitConverter.GetBytes(decrypted[1]).CopyTo(decryptedBytes, 4);
                decryptedWriter.Write(decryptedBytes);
            }
        }
        //шифровка и расшифровка сообщения

        Console.WriteLine("Шифрование и расшифровка завершены. Результаты записаны в файлы.");
        Console.ReadKey();
    }
}
```

5 Пример работы программы

Зашифруем текст из входного текстового файла IN.txt используя ключ из файла KEY.txt (рисунок 5.1-5.2).

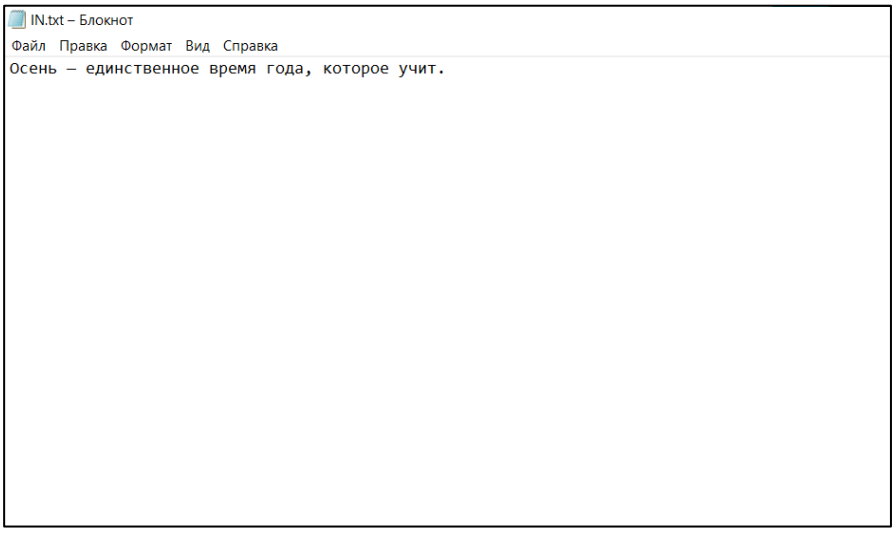

Рисунок 5.1 – Входные данные файла IN.txt

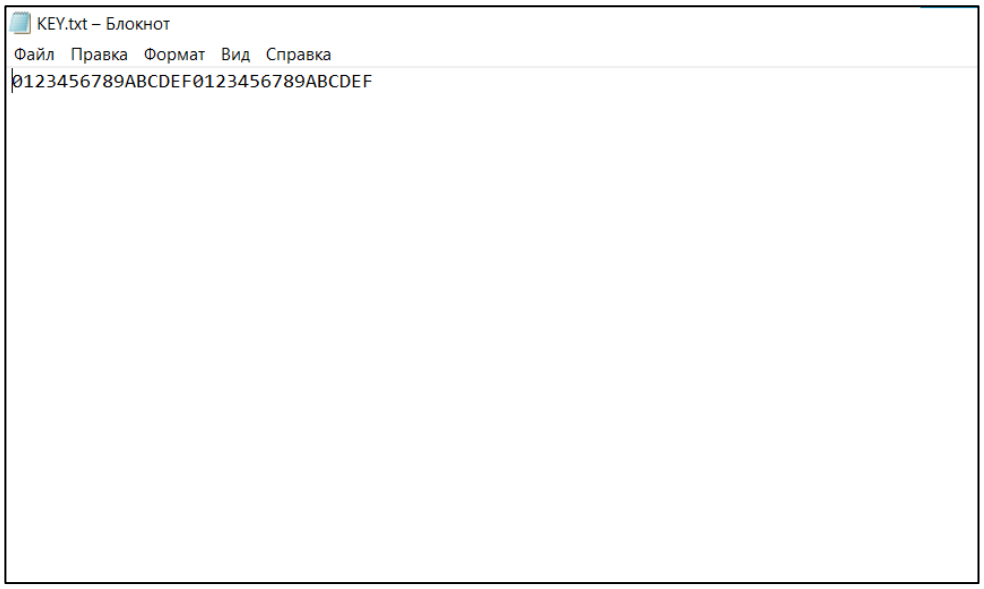

Рисунок 5.2 – Ключ из файла KEY.txt

Запускаем программу и видим, что после выполнения всех операций нам выводится сообщение об завершении всех действий и записи результатов в файлы (рисунок 5.3).

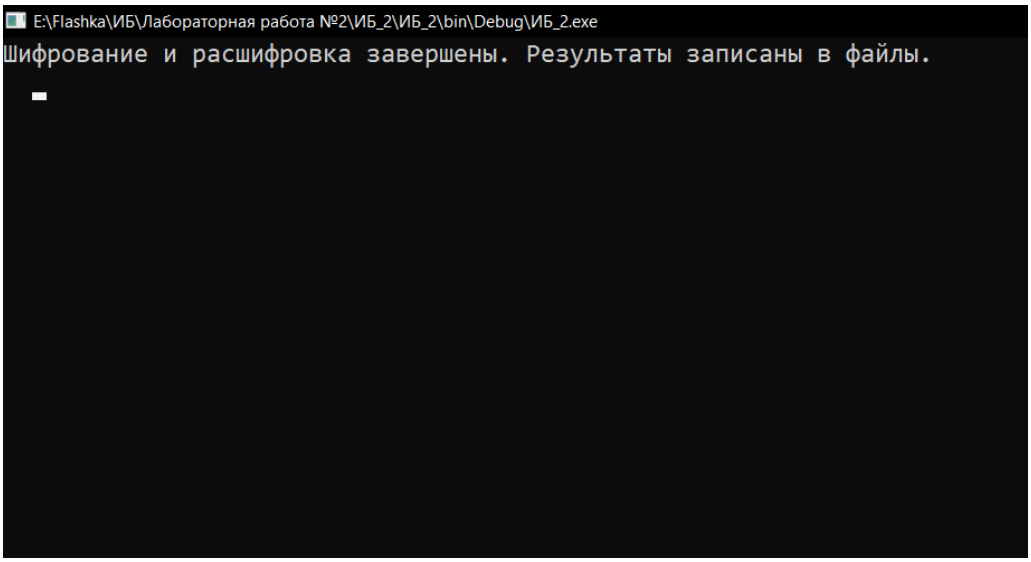

Рисунок 5.3 – Вывод сообщения об завершении всех действий и записи результатов в файлы

В результате работы программы были созданы два файла формата txt. В файле ENCRYPTION.txt записаны зашифрованные данные (рисунок 5.4). В файле OUT.txt записаны изначальные данные, полученные из файла IN.txt (рисунок 5.5).

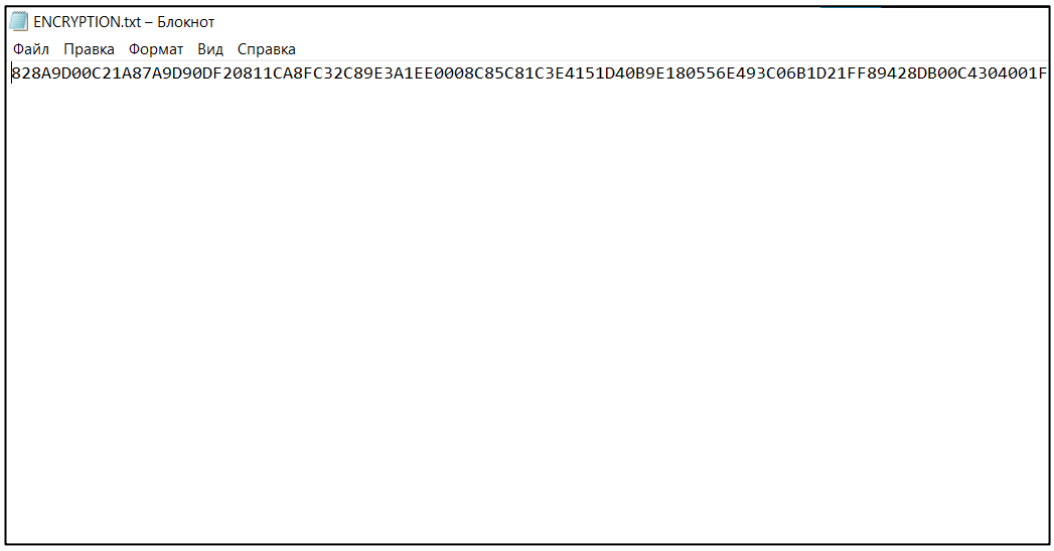

Рисунок 5.4 – Содержимое файла ENCRYPTION.txt

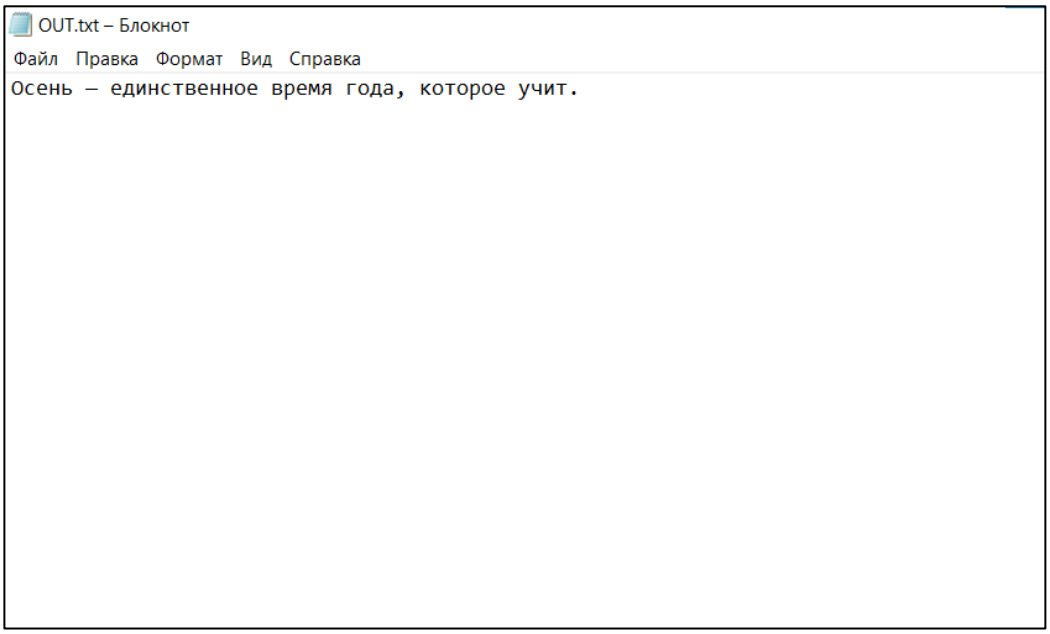

Рисунок 5.5 – Содержимое файла OUT.txt

Во втором примере попробуем зашифровать английский текс (рисунок 5.6).

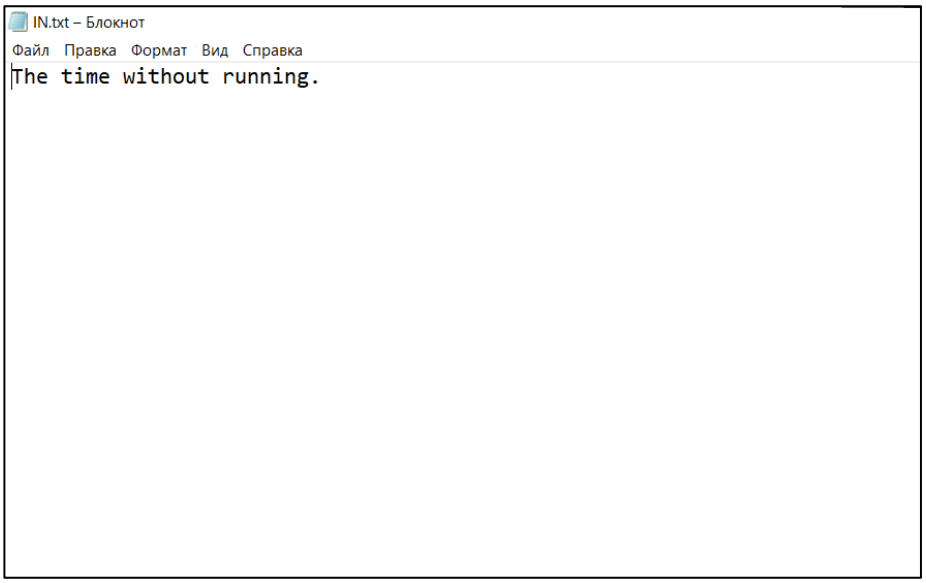

Рисунок 5.6 – Входной файл IN.txt для второго примера

В результате получаем следующие выходные файлы (рисунок 5.7-5.8).

Рисунок 5.7 – Содержимое файла ENCRYPTION.txt

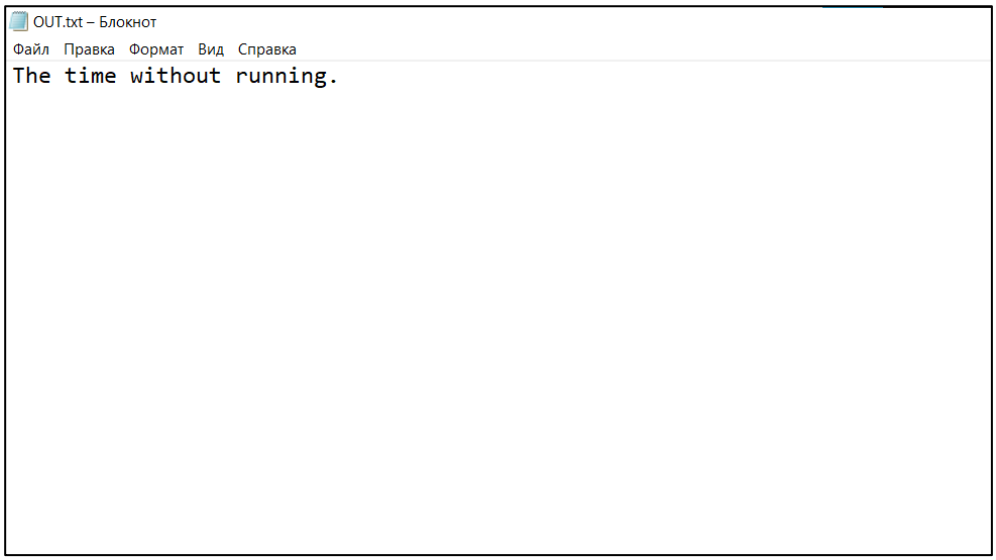

Рисунок 5.8 – Содержимое файла OUT.txt

6 Вывод

В ходе выполнения лабораторной работы я познакомился и научился работать с симметричным алгоритмом шифрования TEA.