МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Чувашский государственный университет имени И.Н. Ульянова»

Факультет информатики и вычислительной техники

Кафедра вычислительной техники

Лабораторная работа №4

По дисциплине “Защита информации” на тему:

«Классический криптографический алгоритм DES»

Вариант 6

Выполнил: ст. гр. ИВТ-11-21

Еремеев С.А.

Проверил: канд. техн. наук. Ковалев С.В.

Чебоксары 2024

**Теоретическая часть**

# Цель работы

Целью работы является знакомство с классическим криптографическим алгоритмом DES и его работой в режиме ECB.

# Основные сведения

Для достижения потребного уровня криптостойкости реальные алгоритмы шифрования, применяемые в информационно-вычислительных сетях общего пользования, включают многократно повторяемые шаги перестановки (транспозиции), гаммирования и нелинейной подстановки. Классическим примером такого подхода является криптографический алгоритм DES (Data Encryption Standard).

Существуют четыре режима работы DES:

* ECB (Electronic Code Book), электронный кодоблокнот;
* CBC (CipherBlock Chaining), сцепление блоков шифра;  CFB (Cipher FeedBack), обратная связь по шифртексту;  OFB (Output FeedBack), обратная связь по выходу.

Настоящая лабораторная работа посвящена изучению работы криптоалгоритма DES в режиме ECB.

Режим ECB является базовым. Для него характерны разбиение открытого текста на блоки по 64 бита и их независимое шифрование при помощи одного и того же 64-битового ключа. Блок шифртекста, как и соответствующий ему блок открытого текста, имеет длину 64 бита. Для шифрования используются 56 бит ключа из 64; оставшиеся 8 бит используются для контроля. Схема реализации данного алгоритма приведена на рисунках 1-3.

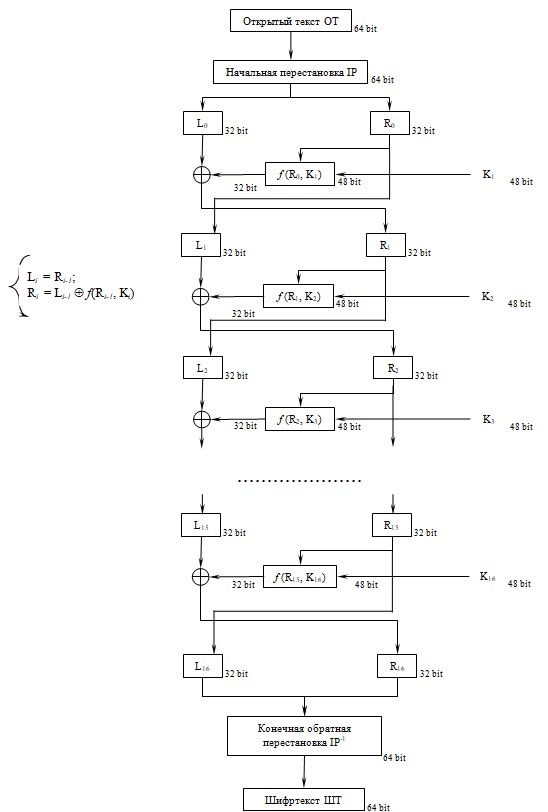


Рисунок 1 – Структурная схема алгоритма шифрования DES

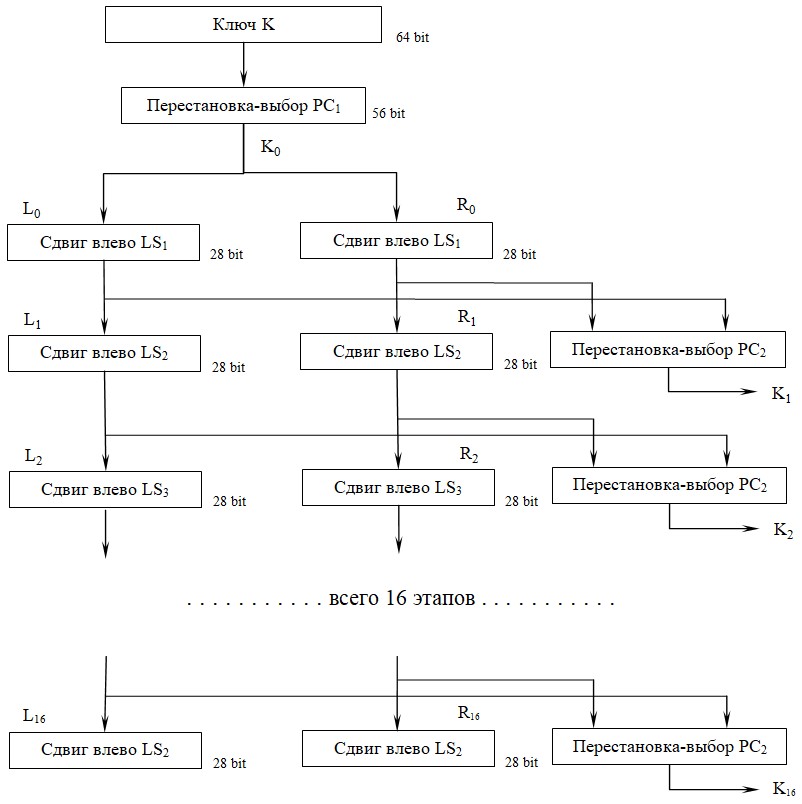


Рисунок 2 – Схема формирования ключей

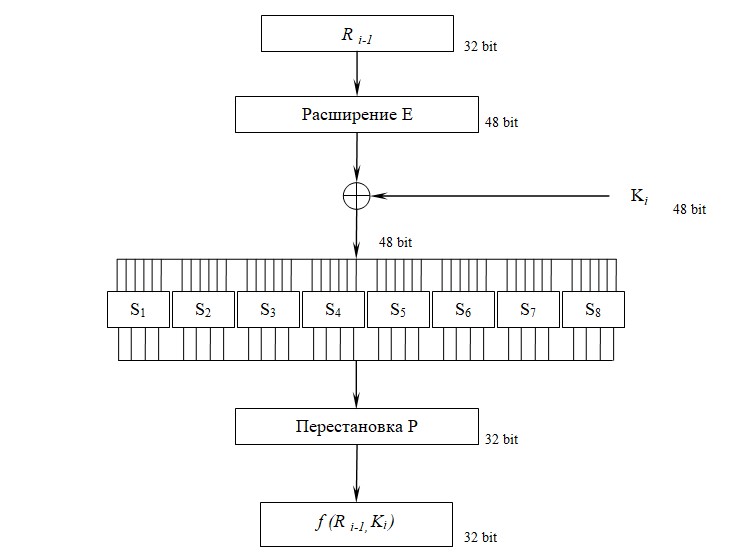


Рисунок 3 – Схема вычисления *f*(R*i-1*, K*i*).

**Практическая часть**

# Текст программы

using System.Security.Cryptography;

using System.Text;

Console.Write("Введите строку: ");

string inputString = Console.ReadLine();

// Генерация DES-ключа

byte[] key = GenerateDESKey();

string keyBinary = ToBinaryString(key);

string truncatedKeyBinary = TruncateKeyTo56Bits(keyBinary);

// Вывод начальных данных

Console.WriteLine("Введённая строка: " + inputString);

Console.WriteLine("Генераторный ключ: " + keyBinary);

Console.WriteLine("Ключ после перестановки PC-1: " + truncatedKeyBinary + " Размер ключа: 56");

// Генерация раундовых ключей

string[] roundKeys = GenerateRoundKeys(truncatedKeyBinary);

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

{

    Console.WriteLine($"K-{i + 1} ключ: {roundKeys[i]}");

}

// Шифрование текста

byte[] encryptedBytes = Encrypt(inputString, key);

string encryptedText = Convert.ToBase64String(encryptedBytes);

Console.WriteLine("Зашифрованный текст (Base64): " + encryptedText);

// Расшифрование текста

string decryptedText = Decrypt(encryptedBytes, key);

Console.WriteLine("Расшифрованный текст: " + decryptedText);

static byte[] GenerateDESKey()

{

    using (DESCryptoServiceProvider des = new DESCryptoServiceProvider())

    {

        des.GenerateKey();

        return des.Key;

    }

}

static string ToBinaryString(byte[] bytes)

{

    StringBuilder binaryString = new StringBuilder();

    foreach (byte b in bytes)

    {

        binaryString.Append(Convert.ToString(b, 2).PadLeft(8, '0'));

    }

    return binaryString.ToString();

}

static string TruncateKeyTo56Bits(string keyBinary)

{

    // Удаляем каждый 8-й бит (контроль четности)

    StringBuilder truncatedKey = new StringBuilder();

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

    {

        if ((i + 1) % 8 != 0) // Пропускаем каждый 8-й бит

        {

            truncatedKey.Append(keyBinary[i]);

        }

    }

    return truncatedKey.ToString();

}

static string[] GenerateRoundKeys(string truncatedKeyBinary)

{

    // Делим ключ на две части

    string C = truncatedKeyBinary.Substring(0, 28);

    string D = truncatedKeyBinary.Substring(28, 28);

    // Массив сдвигов для каждого раунда

    int[] shifts = { 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1 };

    string[] roundKeys = new string[16];

    for (int i = 0; i < 16; i++)

    {

        C = LeftShift(C, shifts[i]);

        D = LeftShift(D, shifts[i]);

        roundKeys[i] = C + D; // Объединяем для раундового ключа

    }

    return roundKeys;

}

static string LeftShift(string input, int shift)

{

    return input.Substring(shift) + input.Substring(0, shift);

}

static byte[] Encrypt(string plainText, byte[] key)

{

    using (DESCryptoServiceProvider des = new DESCryptoServiceProvider())

    {

        des.Key = key;

        des.Mode = CipherMode.ECB; // Можно выбрать другой режим, например, CBC

        des.Padding = PaddingMode.PKCS7;

        ICryptoTransform encryptor = des.CreateEncryptor();

        byte[] plainBytes = Encoding.UTF8.GetBytes(plainText);

        return encryptor.TransformFinalBlock(plainBytes, 0, plainBytes.Length);

    }

}

static string Decrypt(byte[] cipherBytes, byte[] key)

{

    using (DESCryptoServiceProvider des = new DESCryptoServiceProvider())

    {

        des.Key = key;

        des.Mode = CipherMode.ECB; // Должен совпадать с режимом шифрования

        des.Padding = PaddingMode.PKCS7;

        ICryptoTransform decryptor = des.CreateDecryptor();

        byte[] plainBytes = decryptor.TransformFinalBlock(cipherBytes, 0, cipherBytes.Length);

        return Encoding.UTF8.GetString(plainBytes);

    }

}

# Пример открытого текста и соответствующей ему шифрограммы



Рисунок 4 – Пример 1

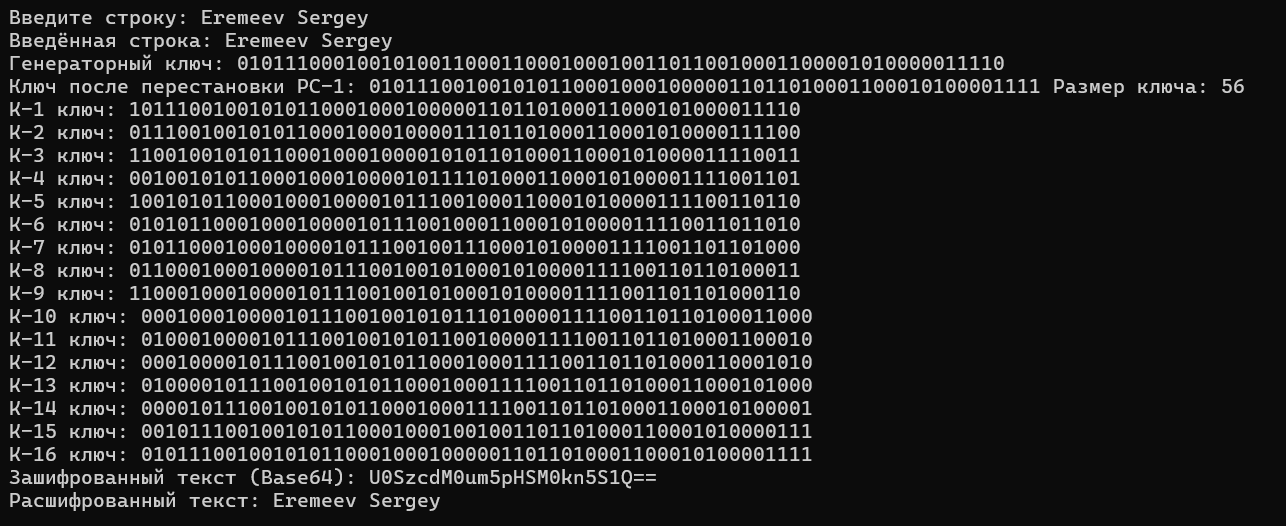


Рисунок 5 – Пример 2



Рисунок 5 – Пример 3

# Вывод

Ознакомился с классическим криптографическим алгоритмом – алгоритмом шифрования данных при помощи перестановки.