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 №6

По дисциплине “Защита информации” на тему:

«Работа алгоритма DES в режиме CFB»

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 особенностями работы алгоритма DES в режиме CFB.

# Задание на работу

В данной лабораторной работе необходимо программно реализовать алгоритмы шифрования и дешифрования DES в режиме CBC. 64-битовый начальный вектор C*0* следует сгенерировать, используя датчик ПСЧ, разработанный в лабораторной работе №3.

# Основные сведения

В отличие от режимов ECB и CBC, оперирующих 64-битовыми блоками, режимы CFB и OFB оперируют *k*-битовыми блоками (1<*k*<63). Это, в частности, позволяет шифровать текстовые данные посимвольно (для кода EBCDIC достаточно положить *k*=8, для кода ASCII - *k*=7).

Шифрование в режиме CFB происходит следующим образом. Вначале на вход поступает случайным образом сгенерированный 64-битовый вектор C*0*, который шифруется алгоритмом DES с некоторым ключом K.

Левые *k* бит результата шифрования гаммируются с левыми *k* битами открытого текста и полученный таким образом *k*-битовый шифртекст поступает на дальнейшую обработку (передачу по каналу связи, хранение и т.п.).

Существенно, что *k*-битовый вектор, гаммируемый с *k* битами открытого текста, загружается в *k* правых блоков входного блока посредством сдвига его содержимого на *k* позиций влево, после чего описанные операции повторяются применительно к следующим *k* битам открытого текста и т.д. Обратим внимание на то, что упомянутый сдвиг содержимого входного блока не является циклическим, в связи с чем левые *k* бит этого содержимого на каждом шаге “выталкиваются” за пределы блока.

Процесс дешифрования в режиме CFB аналогичен описанному и поэтому очевиден из правой половины рис. 1.

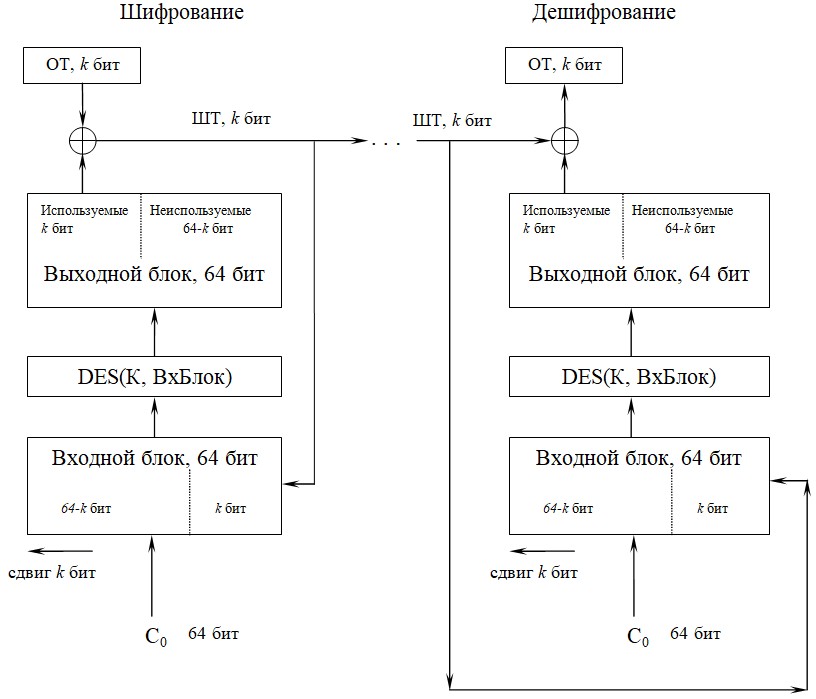


Рисунок 1 – Шифрование DES в режиме CFB

**Практическая часть**

# Текст программы

using System;

using System.Security.Cryptography;

using System.Text;

class Program

{

static void Main()

{

int A = 25, C = 37, T0 = 7, B = 256;

byte[] C0 = GenerateIV(A, C, T0, B); // Генерация начального вектора

Console.Write("Введите строку открытого текста: ");

string plaintext = Console.ReadLine();

// Генерация ключа

byte[] key = Encoding.UTF8.GetBytes("12345678"); // 8 байт для DES

Console.WriteLine("\nИзначальный ключ: " + ConvertToBinaryString(key));

Console.WriteLine("Начальный вектор C0: " + ConvertToBinaryString(C0));

// Шифрование

byte[] encrypted = EncryptDES(plaintext, key, C0);

Console.WriteLine("\nШифртекст (в base64): " + Convert.ToBase64String(encrypted));

// Дешифрование

string decrypted = DecryptDES(encrypted, key, C0);

Console.WriteLine("Дешифрованный текст: " + decrypted);

}

static byte[] GenerateIV(int A, int C, int T0, int B)

{

byte[] iv = new byte[8]; // 64-битовый начальный вектор

int gamma = T0;

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

{

gamma = (A \* gamma + C) % B;

iv[i] = (byte)gamma;

}

return iv;

}

static byte[] EncryptDES(string plaintext, byte[] key, byte[] iv)

{

using (DES des = DES.Create())

{

des.Key = key;

des.IV = iv;

des.Mode = CipherMode.CBC;

des.Padding = PaddingMode.PKCS7;

ICryptoTransform encryptor = des.CreateEncryptor();

byte[] plaintextBytes = Encoding.UTF8.GetBytes(plaintext);

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

}

}

static string DecryptDES(byte[] ciphertext, byte[] key, byte[] iv)

{

using (DES des = DES.Create())

{

des.Key = key;

des.IV = iv;

des.Mode = CipherMode.CBC;

des.Padding = PaddingMode.PKCS7;

ICryptoTransform decryptor = des.CreateDecryptor();

byte[] plaintextBytes = decryptor.TransformFinalBlock(ciphertext, 0, ciphertext.Length);

return Encoding.UTF8.GetString(plaintextBytes);

}

}

static string ConvertToBinaryString(byte[] data)

{

StringBuilder binaryString = new StringBuilder();

foreach (byte b in data)

{

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

}

return binaryString.ToString().Trim();

}

}

Функция шифрования:

static byte[] EncryptDES(string plaintext, byte[] key, byte[] iv)

{

using (DES des = DES.Create())

{

des.Key = key;

des.IV = iv;

des.Mode = CipherMode.CBC;

des.Padding = PaddingMode.PKCS7;

ICryptoTransform encryptor = des.CreateEncryptor();

byte[] plaintextBytes = Encoding.UTF8.GetBytes(plaintext);

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

}

}

Функция дешифрования:

static string DecryptDES(byte[] ciphertext, byte[] key, byte[] iv)

{

using (DES des = DES.Create())

{

des.Key = key;

des.IV = iv;

des.Mode = CipherMode.CBC;

des.Padding = PaddingMode.PKCS7;

ICryptoTransform decryptor = des.CreateDecryptor();

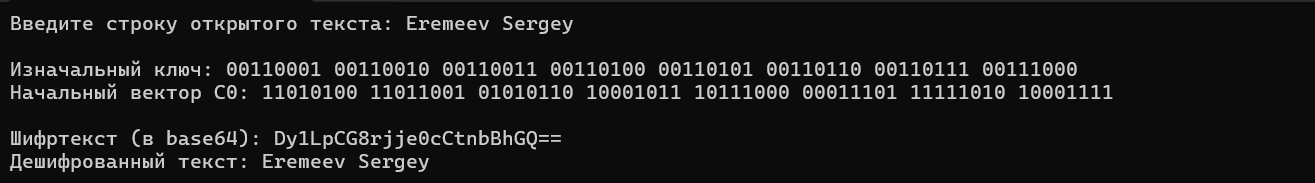
byte[] plaintextBytes = decryptor.TransformFinalBlock(ciphertext, 0, ciphertext.Length);

return Encoding.UTF8.GetString(plaintextBytes);

}

}

# Пример открытого текста и соответствующей ему шифрограммы

Рисунок 2 – Пример 1

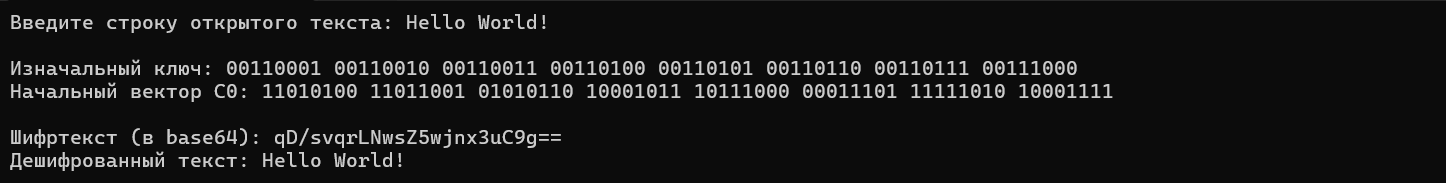
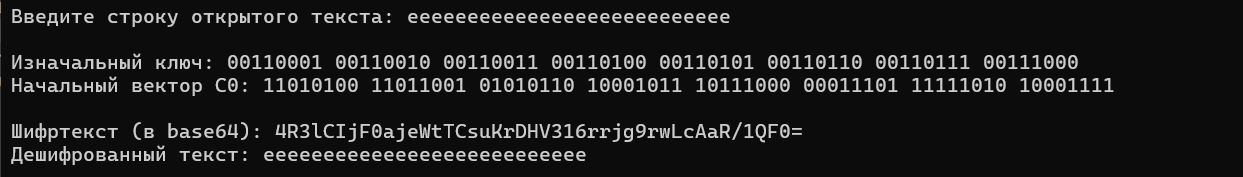


Рисунок 3 – Пример 2

 Рисунок 4 – Пример 3

# Вывод

В ходе данной лабораторной работы был рассмотрен и реализован на языке высокого программирования C# классический криптографический алгоритм DES в режиме CBC.