МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Лабораторная работа № 7**

**ИССЛЕДОВАНИЕ БЛОЧНЫХ ШИФРОВ**

Разработала: Некрасова А.П.

ФИТ 3 курс 5 группа

Преподаватель: Савельева М.Г.

Минск 2023

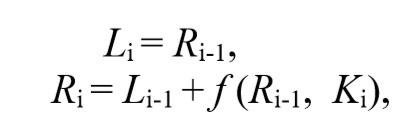
**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации блочных шифров (рассчитана на 4 часа аудиторных занятий).

**Задачи:**

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости блочных шифров.
2. Разработать приложение для реализации указанных преподавателем методов блочного зашифрования/расшифрования.
3. Выполнить анализ криптостойкости блочных шифров.
4. Оценить скорость зашифрования/расшифрования реализованных шифров.
5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Теоретические сведения**

Само название конструкции Фейстеля (сеть) означает ее ячеистую топологию. Формально одна ячейка сети соответствует одному раунду зашифрования или расшифрования сообщения. При зашифровании сообщение разбивается на блоки одинаковой (фиксированной) длины (как правило – 64 или 128 бит). Полученные блоки называются входными. В случае, если длина входного блока меньше, чем выбранный размер, то блок удлиняется установленным способом. Каждый входной блок шифруемого сообщения изначально делится на два подблока одинакового размера: левый (*L0*) и правый (*R0*). Далее в каждом i-ом раунде выполняются преобразования в соответствии с формальным представлением ячейки сети Фейстеля:

 (1)

По какому-либо математическому правилу вычисляется раундовый ключ *Ki*. В приведенном выражение знак «+» соответствует поразрядному суммированию на основе «*XOR*». Расшифрование происходит так же, как и зашифрование, с той лишь разницей, что раундовые ключи будут использоваться в обратном порядке по отношению к зашифрованию. В своей статье Х. Фейстель описывает два блока преобразований с использованием функции *f* (*Ri*-1, *Ki*):

• блок подстановок (*S*-блок, англ. *S*-box);

• блок перестановок (*P*-блок, англ. *P*-box).

Блок подстановок состоит из:

• дешифратора, преобразующего n-разрядное двоичное число в одноразрядное сигнал по основанию 2*n*;

• внутреннего коммутатора;

• шифратора, преобразующего сигнала из одноразрядного 2*n*-ричного в *n-*разрядный двоичный.

В основе криптостойкости блочных шифров лежит идея К. Шеннона в представлении составного шифра таким образом, чтобы от обладал двумя важными свойствами: рассеянием и перемешиванием. Рассеивание должно скрыть отношения между зашифрованным текстом и исходным текстом. Рассеивание подразумевает, что каждый символ (символ или бит) в зашифрованном тексте зависит от одного или всех символов в исходном тексте.

Другими словами, если единственный символ в исходном тексте изменен, несколько или все символы в зашифрованном тексте будут также изменены. Идея относительно перемешивания заключается в том, что оно должно скрыть отношения между зашифрованным текстом и ключом.

Как было указано выше, в основе сети Фейстеля лежит простейшая операция суммирования 2-*х* (*А* + *В*) *n*-разрядных чисел – *XOR*: *А* + *В* (*mod n*). Помимо этой операции некоторые алгоритмы (*Blowfish*, *IDEA*, ГОСТ и др.) предусматривают выполнение операций сложения чисел по модулю более высоких порядков: *XOR*: *А + В* (*mod 2n*).

В DES входной блок данных, состоящий из 64 бит, преобразуется в выходной блок идентичной длины. В алгоритме широко используются рассеивания (подстановки) и перестановки битов текста, о которых мы упоминали выше. Комбинация двух указанных методов преобразования образует фундаментальный строительный блок DES, называемый раундом или циклом. Один блок данных подвергается преобразованию (и при зашифровании, и при расшифровании) в течение 16 раундов. После первоначальной перестановки и разделения 64-битного блока данных на правую (*R0*) и левую (*L0*) половины длиной по 32 бита выполняются 16 раундов одинаковых действий.

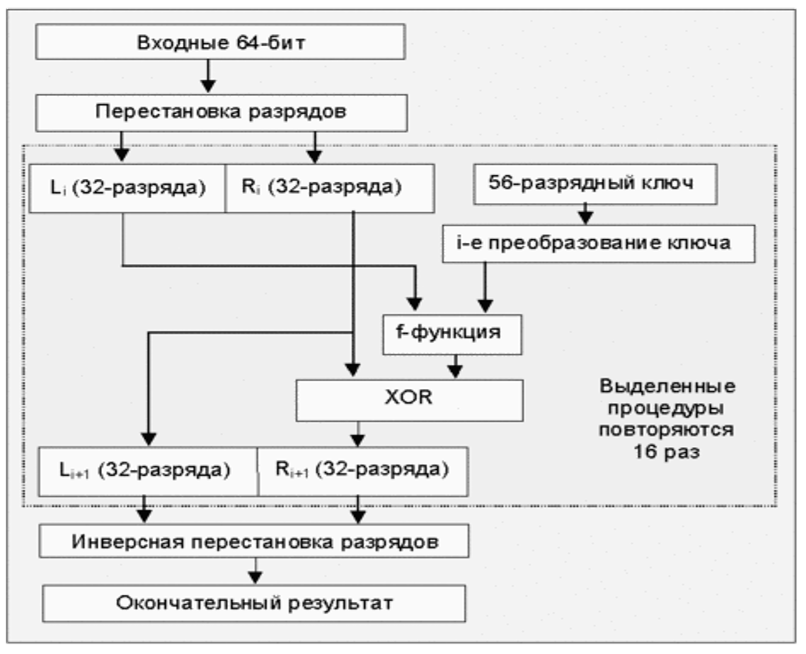


Рисунок 2 – Общая схема алгоритма DES

Функция *f* из себя представляет следующее:



Рисунок 3 – Схема реализации функции f

Некоторые пары ключей при зашифровании переводят открытый текст в идентичный шифртекст. Иными словами, один из ключей пары может расшифровать сообщения, зашифрованные другим ключом пары. Это происходит из-за метода, используемого DES для генерации подключей: вместо 16 различных подключей эти ключи генерируют только два различных подключа. В алгоритме каждый из этих подключей используется восемь раз. Эти ключи, называемые полуслабыми.

Существуют несколько реализаций алгоритма 3DES. Вот некоторые из них:

• DES-EEE3: шифруется 3 раза с 3 разными ключами (операции шифрование-шифрование-шифрование);

• DES-EDE3: 3DES операции шифрование-расшифрование-шифрование с разными ключами:

• DES-EEE2 и DES-EDE2: как и предыдущие, однако, на первом и третьем шаге используется одинаковый ключ.

**Практическая часть**

В связи с поставленными требованиями было разработано приложение, реализующее алгоритм DES.

Алгоритм зашифрования:

public void DESencode(string mes, string key)

{

X = ""; Xbin = ""; k = ""; XIP = "";

Blocks = new string[0];

binBlocks = new string[0];

richTextBox.Text = "";

X = mes;

richTextBox.Text += X + '\n';

X = StringToRightLength(X);

CutStringIntoBlocks(X);

string blockBin;

foreach (string block in Blocks)

{

blockBin = BinaryConvert.ToBinary(BinaryConvert.ConvertToByteArray(block, Encoding.Default));

Xbin += blockBin;

}

richTextBox.Text += Xbin + "\nk = ";

k = Permut(BinaryConvert.ToBinary(BinaryConvert.ConvertToByteArray(key, Encoding.Default)), keyP);

richTextBox.Text += k + '\n';

CutBinaryStringIntoBlocks(Xbin);

foreach (string binBlock in binBlocks)

{

string binBlockIP = Permut(binBlock, IP);

XIP += binBlockIP;

}

CutBinaryStringIntoBlocks(XIP);

string L, R, L1, R1, R2, K, Y, mess = "";

foreach (string binBlock in binBlocks)

{

richTextBox.Text += $"BLOCK {Array.IndexOf(binBlocks, binBlock)}" + '\n';

Y = "";

string block = binBlock;

for (int i = 1; i <= shiftKey.Length; i++)

{

L = block.Substring(0, 32);

R = block.Substring(32);

K = GenKey(k, i);

R1 = f(R, K);

L1 = R;

R2 = XOR(L, R1);

Y = String.Concat(L1, R2);

richTextBox.Text += $"Round {i} = " + Y + '\n';

block = Y;

}

Y = Permut(Y, FP);

richTextBox.Text += $"Final Permutation = " + Y + '\n';

mess += Y;

}

encoded = BinaryConvert.ConvertBytesToString(BinaryConvert.GetBytes(mess), Encoding.Default);

richTextBox.Text += $"ENCODED " + encoded + '\n';

textBox1.Text = "";

foreach (char symbol in encoded)

{

textBox1.Text += symbol;

}

}

Листинг 1 – Реализация шифрования алгоритмом DES

Алгоритм расшифрования:

public void DESdecode(string mes, string key)

{

X = ""; Xbin = ""; k = ""; XIP = "";

Blocks = new string[0];

binBlocks = new string[0];

X = mes;

X = StringToRightLength(X);

CutStringIntoBlocks(X);

string blockBin;

foreach (string block in Blocks)

{

blockBin = BinaryConvert.ToBinary(BinaryConvert.ConvertToByteArray(block, Encoding.Default));

Xbin += blockBin;

}

k = Permut(BinaryConvert.ToBinary(BinaryConvert.ConvertToByteArray(key, Encoding.Default)), keyP);

CutBinaryStringIntoBlocks(Xbin);

foreach (string binBlock in binBlocks)

{

string binBlockIP = Permut(binBlock, IP);

XIP += binBlockIP;

}

CutBinaryStringIntoBlocks(XIP);

string L, R, L1, R1, L2, K, Y, mess = "";

foreach (string binBlock in binBlocks)

{

richTextBox.Text += $"BLOCK {Array.IndexOf(binBlocks, binBlock)}" + '\n';

Y = "";

string block = binBlock;

for (int i = shiftKey.Length; i > 0; i--)

{

L = block.Substring(0, 32);

R = block.Substring(32);

K = GenKey(k, i);

L1 = f(L, K);

R1 = L;

L2 = XOR(R, L1);

Y = String.Concat(L2, R1);

richTextBox.Text += $"Round {i} = " + Y + '\n';

block = Y;

}

Y = Permut(Y, FP);

richTextBox.Text += $"Final Permutation = " + Y + '\n';

mess += Y;

}

encoded = BinaryConvert.ConvertBytesToString(BinaryConvert.GetBytes(mess), Encoding.Default);

richTextBox.Text += $"DECODED " + encoded + '\n';

textBox1.Text = encoded;

}

Листинг 2 – Реализация расшифрования алгоритмом DES

Результат выполнения программы представлен на рисунках 4, 5, 6:

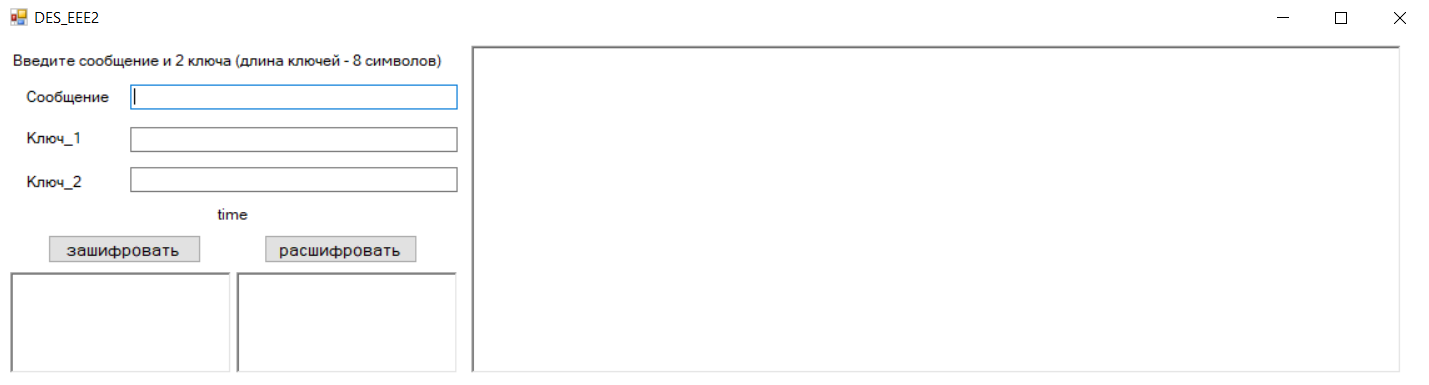


Рисунок 4 – Запуск приложения

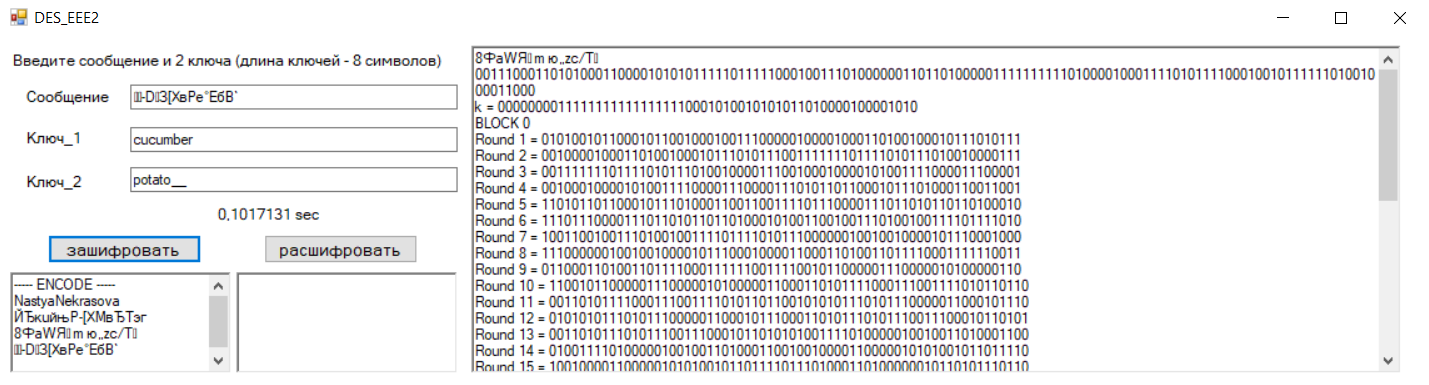


Рисунок 5 – Этапы зашифрования

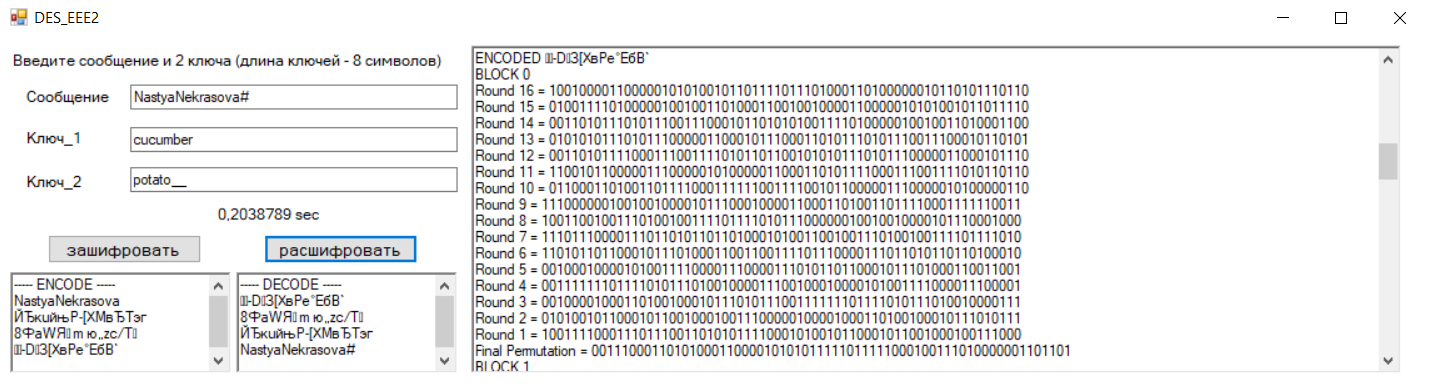


Рисунок 6 – Этапы расшифрования

Как видно из результатов работы приложения исходный и зашифрованный текст совпали, значит алгоритмы шифрования и дешифрования работают исправно.

Лавинный эффект:

Рассмотрим 16 этапов преобразования блоков:

Round 1 = 0011111110110110111100001100100110110101010010100100111101101101

Round 2 = 1011010101001010010011110110110110101011111010100011000110001101

Round 3 = 1010101111101010001100011000110100000100111011101100101001110001

Round 4 = 0000010011101110110010100111000100011100111101011101101100101010

Round 5 = 0001110011110101110110110010101001101010011000000010111001000101

Round 6 = 0110101001100000001011100100010101111010110010101101111110011110

Round 7 = 0111101011001010110111111001111011010101101000000110111100001110

Round 8 = 1101010110100000011011110000111001001010010110011101100010011001

Round 9 = 0100101001011001110110001001100110110001011111001111101001010000

Round 10 = 1011000101111100111110100101000011111100100111000001011110000100

Round 11 = 1111110010011100000101111000010001100100110000100000000011000001

Round 12 = 0110010011000010000000001100000111010000010001010110110010100001

Round 13 = 1101000001000101011011001010000111110110111111100011111111001100

Round 14 = 1111011011111110001111111100110001001001010011110010111111001000

Round 15 = 0100100101001111001011111100100000011011001101100100011001010001

Round 16 = 0001101100110110010001100101000100101010000011000100000000111011

Final Permutation = 0100001111010110001101001110001001010011100100100000110100000000

В символьном виде на входе мы получаем: «ѓеЎЦ№­\_n», а на выходе «CЦ4вS’».

Таблица 1 – Тестирование ф-ций шифрования/расшифрования

|  |  |  |  |
| --- | --- | --- | --- |
| Исходное сообщение | Ключ\_1 | Ключ\_2 | Зашифрованное сообщение |
| nastyanekrasova | potato\_\_ | cucumber | JїЊОёЁХ e—™оPi |
| Nastyanekrasava | potato\_\_ | cucumber | 2Іo‰^yMt! ќ6 |
| nastyanekrasova | patato\_\_ | cucumber | Іµ„aАИHтrqЇ&Ч |

Как видим начальное и конечное значения полностью отличаются, что свидетельствует о наличии лавинного эффекта в шифре DES.

Слабые ключи:

Четыре ключа из 256 *возможных ключей* называются **слабыми ключами**. Слабые ключи — это одни из тех, которые после операции удаления проверочных бит состоят из всех нулей или всех единиц или половины нулей и половины единиц. Такие ключи показаны в таблице 2.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Таблица 2. Слабые ключи   |  |  | | --- | --- | | **Ключи до удаления проверочных бит (64 бита)** | **Действующие ключи (56 бит)** | | 0101 0101 0101 0101 | 0000000 0000000 | | 1F1F 1F1F 1F1F 1F1F | 0000000 FFFFFFF | | E0E0 E0E0 E0E0 E0E0 | FFFFFFF 0000000 | | FEFE FEFE FEFE FEFE | FFFFFFF FFFFFFF | |

Ключи раунда, созданные от любого из этих слабых ключей, — те же самые и имеют тот же самый тип, что и ключ шифра. Например, эти шестнадцать ключей раунда создают первый ключ, который состоит из всех нулей или всех единиц или наполовину из нулей и единиц.

Это происходит по той причине, что алгоритм генерирования ключей сначала делит ключ шифра на две половины. Смещение или перестановка блока не изменяют блок, если он состоит из всех нулей, или всех единиц, или наполовину из нулей и единиц.

В чем опасность использования слабых ключей? Если мы зашифровали блок слабым ключом и впоследствии расшифровали результат тем же самым слабым ключом, мы получаем первоначальный блок. Процесс создает один и тот же первоначальный блок, если мы расшифровываем блок дважды. Другими словами, каждый слабый ключ есть инверсия самого себя: как это показано на [рис. 8.11](https://intuit.ru/studies/professional_retraining/940/courses/408/lecture/9362?page=4#image.8.11).

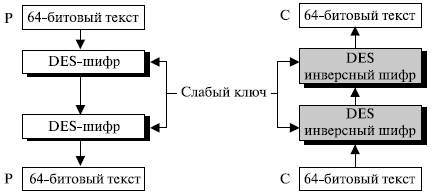


Рис. 8.11.Двойное шифрование и дешифрование со слабым ключом

Слабых ключей надо избегать, потому что противник может легко распознать их на перехваченном шифре. Если после двух этапов дешифрации результат тот же самый, противник определяет, что он нашел ключ.

Пример:

Ключ: 0x0000 0000 0000 0000

Зашифрованный текст: WmNЊ„‑¬0

Исходный текст: nastyanekrasova

Ключ: 0x0000 0000 0000 0000

Исходный текст: WmNЊ„‑¬0

Зашифрованный текст: nastyanekrasova

На лавинный эффект использование слабых ключей в моём случае не повлияло.

Сжатие файлов:

Таблица 3 – Сжатие файлов

|  |  |
| --- | --- |
|  |  |
|  |  |

Файл с исходным текстом получилось сжать на 46%.

Файл с зашифрованным текстом расширился на 5%.

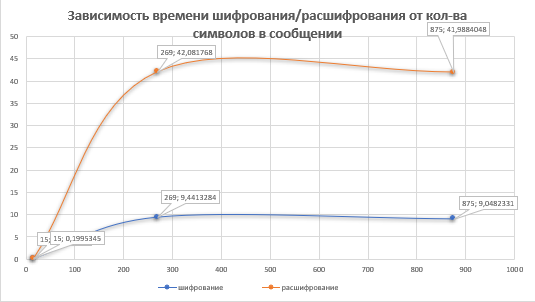
****

Рис 8 – Зависимость времени шифрования/расшифрования от кол-ва символов

**Вывод:**

DES — алгоритм для симметричного шифрования, разработанный фирмой IBM и утверждённый правительством США в 1977 году как официальный стандарт. Размер блока для DES равен 64 битам. В основе алгоритма лежит сеть Фейстеля с 16 циклами и ключом, имеющим длину 56 бит.

В основе алгоритма лежит сеть Фейстеля с 16 циклами (раундами) и ключом, имеющим длину 56 бит. Алгоритм использует комбинацию нелинейных (S-блоки) и линейных (перестановки E, IP, IP-1) преобразований.

Лавинный эффект означает, что небольшие изменения в исходном тексте (или ключе) могут вызвать значительные изменения в зашифрованном тексте. Было доказано, что DES имеет все признаки этого свойства.