Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**Криптографические методы защиты информации**

**Лабораторная работа №7**

ИССЛЕДОВАНИЕ БЛОЧНЫХ ШИФРОВ

Вариант №6

Выполнил:

Студент 3 курса 4 группы ФИТ

Левша Марк Сергеевич

Минск 2023

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации блочных шифров.

**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.

**Практическое задание**

Было разработано приложение для шифрования текстов на английском языке алгоритмом DES.

Перед зашифрованием и расшифрованием необходимо подготовить раундовые ключи (всего в алгоритме 16 раундов и 16 раундовых ключей):

1. Генерируется 64 битный ключ

2. 64-битный ключ с помощью таблицы перестановок превращается в 56-битовый.

[57, 49, 41, 33, 25, 17, 9 , 1 , 58, 50, 42, 34, 26, 18,

10, 2 , 59, 51, 43, 35, 27, 19, 11, 3 , 60, 52, 44, 36,

63, 55, 47, 39, 31, 23, 15, 7 , 62, 54, 46, 38, 30, 22,

14, 6 , 61, 53, 45, 37, 29, 21, 13, 5 , 28, 20, 12, 4 ]

3. 56-битный ключ разбивается на два блока по 28 бит - C0 и D0 блоки (первые 28 битов - C0 блок, остальные биты - D0 блок)

4. Блок C0 и D0 сдвигаются циклически влево на

[1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]

После всех этих сдвигов получаем 16 пар С и D блоков по 28 бит.

5. Пары С0 и D0 блоков соединяем вместе - получаем 16 56-битных ключа.

6. Каждый 56-битный ключ сжимаем с помощью таблицы:

[14, 17, 11, 24, 1 , 5 , 3, 28,

15, 6 , 21, 10, 23, 19, 12, 4,

26, 8 , 16, 7 , 27, 20, 13, 2,

41, 52, 31, 37, 47, 55, 30, 40,

51, 45, 33, 48, 44, 49, 39, 56,

34, 53, 46, 42, 50, 36, 29, 32]

На выходе получаем 16 раундовых ключей по 48 бит. Эти ключи далее используются для зашифрования/расшифрования.

Зашифрование

1. Разбиваем сообщение на блоки по 64-бита

2. Перемешиваем каждый 64-битный блок в соответствии с таблицей

[58, 50, 42, 34, 26, 18, 10, 2,

60, 52, 44, 36, 28, 20, 12, 4,

62, 54, 46, 38, 30, 22, 14, 6,

64, 56, 48, 40, 32, 24, 16, 8,

57, 49, 41, 33, 25, 17, 9 , 1,

59, 51, 43, 35, 27, 19, 11, 3,

61, 53, 45, 37, 29, 21, 13, 5,

63, 55, 47, 39, 31, 23, 15, 7]

3. Разбиваем каждый 64-битный блок на два 32-битных блока *L* и *R*

4. Прохождение 16 раундов:

4.1. *L* остаётся тем же, *R* расширяется до 48-битов в соответствии с таблицей

[32, 1 , 2 , 3 , 4 , 5 ,

4 , 5 , 6 , 7 , 8 , 9 ,

8 , 9 , 10, 11, 12, 13,

12, 13, 14, 15, 16, 17,

16, 17, 18, 19, 20, 21,

20, 21, 22, 23, 24, 25,

24, 25, 26, 27, 28, 29,

28, 29, 30, 31, 32, 1 ]

4.2. L остаётся тем же, R = R XOR «соответствующий раундовый ключ» (для первого раунда - первый раундовый ключ, для второго раунда - второй раундовый и т.д.)

4.3. L остаётся тем же, 48 битный Rблок разбивается на 8 блоков по 6 бит и каждый блок обрабатывает с помощью *S*-блоков (таблица ниже):

0: [[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 6, 7, 0, 7],

[0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],

[4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],

[15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13]],

1: [[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10],

[3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5],

[0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15],

[13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9]],

2: [[10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8],

[13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1],

[13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7],

[1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12]],

3: [[7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15],

[13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9],

[10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4],

[3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14]],

4: [[2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 14, 13, 0, 14, 9],

[14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6],

[4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14],

[11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3]],

5: [[12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11],

[10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8],

[9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6],

[4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13]],

6: [[4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1],

[13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6],

[1, 4, 11, 13, 12, 3, 7, 13, 10, 15, 6, 8, 0, 5, 9, 2],

[6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12]],

7: [[13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7],

[1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2],

[7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8],

[2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11]]

На выходе получаем 8 блоков по 4 бита - соединяем в 32-битный блок - R блок

4.4. L остаётся тем же, Rблок перемешивается в соответствии с таблицей:

[16, 7, 20, 21, 29, 12, 28, 17,

1, 15, 23, 26, 5, 18, 31, 19,

2, 8, 24, 14, 32, 27, 3, 9,

19, 13, 30, 6, 22, 11, 4, 25]

4.5. R = L XOR R, L = R

4.6. Выполнить 4.1-4.6 ещё 15 раз

5. L и R блок соединить в 64 битный блок и перемещать с помощью таблицы:

[40, 8, 48, 16, 56, 24, 64, 32,

39, 7, 47, 15, 55, 23, 63, 31,

38, 6, 46, 14, 54, 22, 62, 30,

37, 5, 45, 13, 53, 21, 61, 29,

36, 4, 44, 12, 52, 20, 60, 28,

35, 3, 43, 11, 51, 19, 59, 27,

34, 2, 42, 10, 50, 18, 58, 26,

33, 1, 41, 9, 49, 17, 57, 25]

6. Все 64 битные блоки соединяются в 1 блок, что и будет являться зашифрованным сообщением.

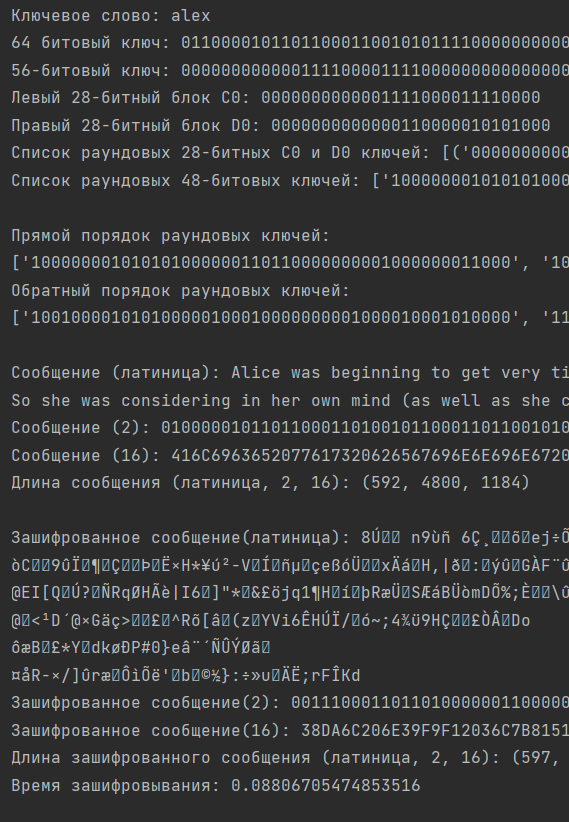
На рисунке 1 представлен консольный вывод работы алгоритма DES.

Рис. 1 – Консольный вывод программы

Для оценки степени сжатия открытого текста и зашифрованного текста было использован алгоритм Лемпеля-Зива. Эффективность сжатия текстов представлена на рисунках 2 и 3

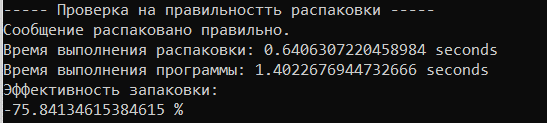


Рис. 2 – Эффективность сжатия открытого текста

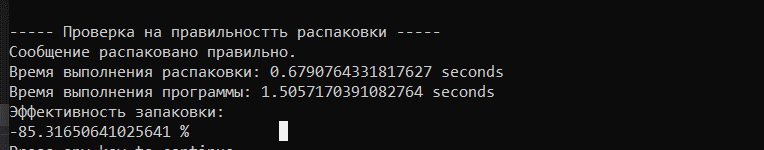


Рис. 3 – Эффективность сжатия зашифрованного текста

Эффективность сжатия уменьшилась после зашифровки т.к. увеличилась энтропия сообщения.

При использовании слабых и полуслабых ключей значения блоков значение ключей на разных шагах могут не отличаться или почти не отличаться, это приводит к тому, тому, что на нескольких шагах, идущих подряд, будут одни и те же значения левого и правого блока.

Так же было оценено время выполнения операций расшифрования и зашифрования, в зависимости от длины сообщения. График зависимости представлен на рисунке 4 и имеет экспоненциальный вид.

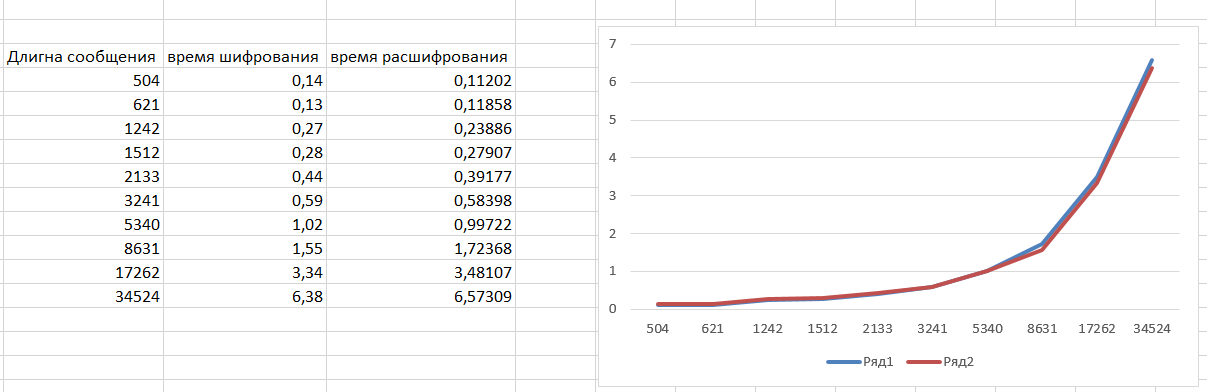


Рис. 4 – Зависимость времени шифрования и расшифрования от длины сообщения

Что касаемо лавинного эффекта, при длине сообщения 93 зашифрованное сообщение в бинарном виде выглядит так

Если к исходному сообщению в начале припишем символ «*A*», то длинна сообщения станет 94, а зашифрованное сообщение станет:

110011111100100011111110011100011110110001100101000011010011111011101001001110111111101010011001110001101111011110000011100010110011111101011010010100100101101001000000001111011011101101111100001100111100110011110111101001000111010000110011010111101101000110111101111010011011011001000001000010110001011110111011110001001001101000100010000010100000000100110100011001111110111011010000101110110111111000111100011110011111010101101000011100001101101101011111010101100000010000011101110111110111001001001000100000101101011100011001101011100100100010111111001011111111101001111011011110110110001011001011010000101110111010110101110000001110010011111000011001010101011101010101000111110000010010110011001111001000101101111010111101001100111101000010101001010100011100100011

Если смотреть по битам, то своё значение поменяли большое количество бит, что говорит о наличии лавинного эффекта.

Так же для изучения лавинного эффекта была построена гистограмма количества отличающихся бит на каждом шаге начального сообщения и сообщения с изменённым первым битом. Как видно из графика разница битов растет и уже к двадцатому шагу разница составляет 35-40 битов, учитывая то, что начальное сообщение состоит из 27 символов т.е. 216 битов что говорит о лавинном эффекте.

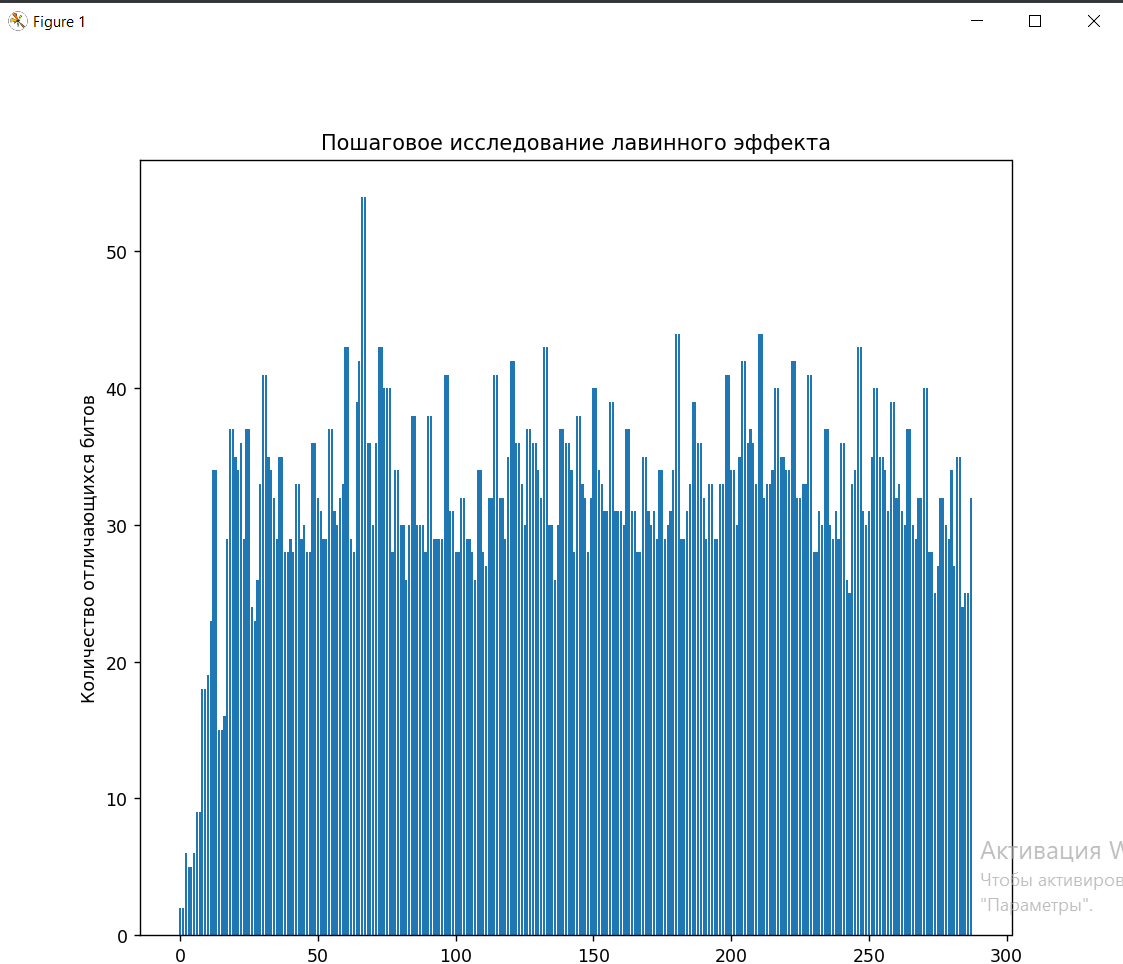


Рис. 5– количество отличающихся бит на каждом шаге начального сообщения и сообщения с изменённым первым битом

Начальное и зашифрованное в 16 виде сообщения были архивированы и оценены степени сжатия:

При сжатии исходного сообщения в формате txt имеет размер 138 байта, а после сжатия 109 байт, сообщение сжалось на 0,21%. При сжатии зашифрованного сообщения начальный размер составлял 367 байт, а после сжатия 315 байта, сообщение сжалось на 0,14%.

**Вывод**: в ходе лабораторной работы был разобран и реализован метод DES, был сделан вывод, что энтропия зашифрованного сообщения в DES увеличивается и так же рассмотрены разные типы ключей.