МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образование «Белорусский государственный технологический университет»

**«**ИССЛЕДОВАНИЕ БЛОЧНЫХ ШИФРОВ**»**

Студент: Макрица Ф Н

ФИТ 3 курс 7 группа

Вариант 1

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 эксперимент

**Теоретические сведения**

**История:**

В 1972 г. Национальное бюро стандартов США (ныне – Национальный институт стандартов и технологии, National Institute of Standarts & Technology – NIST) инициировало программу защиты каналов связи и компьютерных данных. Одна из целей – разработка единого стандарта криптографического шифрования. Основными критериями оценки алгоритма являлись следующие:

• алгоритм должен обеспечить высокий уровень защиты;

• алгоритм должен быть понятен и детально описан;

• криптостойкость алгоритма должна зависеть только от ключа;

• алгоритм должен допускать адаптацию к различным применениям;

• алгоритм должен быть разрешен для экспорта.

В качестве начального варианта нового алгоритма рассматривался Lucifer – разработка компании IBM начала семидесятых годов. В основе указанного алгоритма использовались два запатентованных в 1971 г. Хорстом Фейстелем (Horst Feistel) устройства, реализующие различные алгоритмы шифрования, позже получившие название шифр (сеть) Фейстеля (Feistel Cipher, Feistel Network). В первой версии проекта Lucifer сеть Фейстеля не использовалась.

После многочисленных согласований, специальных конференций, где рассматривались в основном вопросы криптостойкости алгоритма, подлежащего утверждению в качестве федерального стандарта, в ноябре 1976 г. был утвержден стандарт DES (Data Encryption Standard – стандарт шифрования данных). Предполагалось, что стандарт будет реализовываться только аппаратно.

В 1981 г. ANSI одобрил DES в качестве стандарта для публичного использования (стандарт ANSI Х3.92), назвав его алгоритмом шифрования данных (Data Encryption Algorithm – DEA).

В 1987 г. были разработаны алгоритмы FEAL и RC2. Сети Фейстеля получили широкое распространение в 1990-е гг. – в годы появления таких алгоритмов, как Blowfish (1993), TEA (1994), RC5 (1994), CAST-128 (1996), XTEA (1997), XXTEA (1998), RC6 (1998) и др. На основе сети Фейстеля в 1990 г. в СССР был принят в качестве ГОСТ 28147–89 стандарт шифрования.

Предполагалось, что DES будет сертифицироваться каждые 5 лет. Срок действия последнего сертификата на территории США истек практически к концу ХХ в. К тому времени DES был вскрыт «лобовой атакой».

В 1998 г. NIST объявил конкурс на новый стандарт, который завершился в 2001 г. принятием AES (Advanced Encryption Standard).

Все перечисленные стандарты и алгоритмы блочных шифров (БШ) строятся на основе подстановочных и перестановочных шифров, т. е. являются комбинационными. БШ относятся также к классу симметричных.

Блочное зашифрование (расшифрование) предполагает разбиение исходного открытого (зашифрованного) текста на равные блоки, к которым применяется однотипная процедура зашифрования (расшифрования). Указанная однотипность характеризуется прежде всего тем, что процедура зашифрования (расшифрования) состоит из совокупности повторяющихся наборов преобразований, называемых раундами.

Основные требования к шифрам рассматриваемого класса можно сформулировать следующим образом:

• даже незначительное изменение исходного сообщения должно приводить к существенному изменению зашифрованного сообщения;

• устойчивость к атакам по выбранному тексту;

• алгоритмы зашифрования/расшифрования должны быть реализуемыми на различных платформах;

• алгоритмы должны базироваться на простых операциях;

• алгоритмы должны быть простыми для написания кода, вероятность появления программных ошибок должна быть низкой;

• алгоритмы должны допускать их модификацию при переходе на иные требования по уровню криптостойкости.

Алгоритм DES

Входной блок данных, состоящий из 64 битов, 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 () и левую () половины длиной по 32 бита выполняются 16 раундов одинаковых действий.

В табл. 1.1 показан принцип первоначальной перестановки разрядов (IP) входного 64-битного слова.

Таблица 1.1

**Начальная перестановка**

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

Выполненная перестановка означает, например, что первый бит входного блока сообщения будет размещен на 40-й позиции (цифра «1» выделена жирным), а 58-й (выделено жирным с подчеркиванием) – на первой и т. д. Из беглого анализа выполненной перестановки легко понять принцип. Алгоритм перестановки разрабатывался для облегчения загрузки блока входного сообщения в специализированную микросхему. Вместе с тем эта операция придает некоторую «хаотичность» исходному сообщению, снижая возможность использования криптоанализа статистическими методами.

Левая и правая ветви каждого промежуточного значения обрабатываются как отдельные 32-битные значения, обозначенные и .

Вначале правая часть блока расширяется до 48 битов с использованием таблицы, которая определяет перестановку плюс расширение на 16 битов. Эта операция приводит размер правой половины в соответствие с размером ключа для выполнения операции .

Кроме того, за счет выполнения этой операции быстрее возрастает зависимость всех битов результата от битов исходных данных и ключа (это называется «лавинный эффект»).

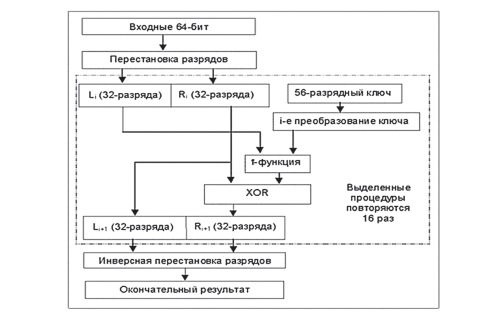


Рисунок 1 – Общая схема алгоритма DES

После выполнения перестановки с расширением для полученного 48-битного значения выполняется операция *XOR* с 48-битным подключом . Затем полученное 48-битное значение подается на вход блока подстановки *S* (от англ. *substitution* – подстановка), результатом которой является 32-битное значение. Подстановка выполняется в восьми блоках подстановки или восьми *S*-блоках (*S-boxes*). При выполнении этой операции 48 битов данных делятся на восемь 6-битных подблоков, каждый из которых по соответствующей таблице замен замещается четырьмя битами. Подстановка с помощью *S*-блоков является одним из важнейших этапов *DES*. Таблицы замен для этой операции специально спроектированы так, чтобы обеспечивать максимальную криптостойкость. В результате выполнения этого этапа получаются восемь 4-битных блоков, которые вновь объединяются в единое 32-битное значение.

Далее полученное 32-битное значение обрабатывается с помощью перестановки *Р* (от англ. *permutation* – перестановка), которая не зависит от используемого ключа. Целью перестановки является такое максимальное переупорядочивание битов, чтобы в следующем раунде шифрования каждый бит с большой вероятностью обрабатывался другим *S*-блоком.

И наконец, результат перестановки объединяется с помощью операции *XOR* с левой половиной первоначального 64-битного блока данных. Затем левая и правая половины меняются местами, и начинается следующий раунд.

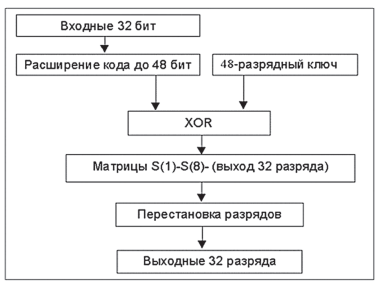


Рисунок 2 – Схема реализации функции f

После выполнения 16-раундового зашифрования 64-битного блока данных осуществляется конечная перестановка ( ). Она является обратной к перестановке *IP*. Конечная перестановка определяется табл. 1.2.

Таблица 1.2

**Конечная перестановка**

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

Каждый 8-й бит исходного 64-битного ключа отбрасывается.

Эти 8 битов, находящих в позициях 8, 16, 24, 32, 40, 48, 56, 64, изначально добавляются в исходный ключ таким образом, чтобы каждый байт содержал четное число единиц. Это используется для обнаружения ошибок при обмене и хранении ключей по известным алгоритмам избыточного кодирования. Один избыточный бит в ключе DES формируется, как видим, в соответствии с кодом простой четности. Этот код позволяет в кодовом слове (в нашем случае – в каждом байте ключа) обнаруживать ошибки, количество которых нечетно.

При расшифровании на вход алгоритма подается зашифрованный текст. Единственное отличие состоит в обратном порядке использования частичных ключей . Ключ *K16* используется в первом раунде, *K1* – в последнем.

После последнего раунда процесса расшифрования две половины выхода меняются местами так, чтобы вход заключительной перестановки был составлен из подблоков *R16* и *L16*. Выходом этой стадии является расшифрованный текст.

**Практическое задание**

Приложение должно реализовывать следующие операции:

• разделение входного потока данных на блоки требуемой длины с необходимым дополнением последнего блока;

• выполнение требуемых преобразований ключевой информации;

• выполнение операций зашифрования/расшифрования;

• оценка скорости выполнения операций зашифрования/расшифрования;

•пошаговый анализ лавинного эффекта с подсчетом количества изменяющихся символов по отношению к исходному слову.

Код для шифрования:

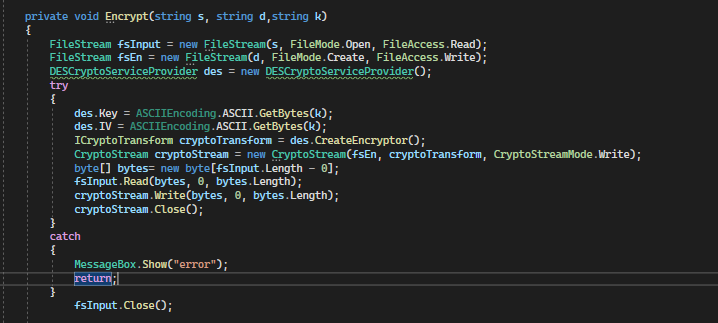


Рисунок 3.1 – Код для шифрования

Код для дешифрования:

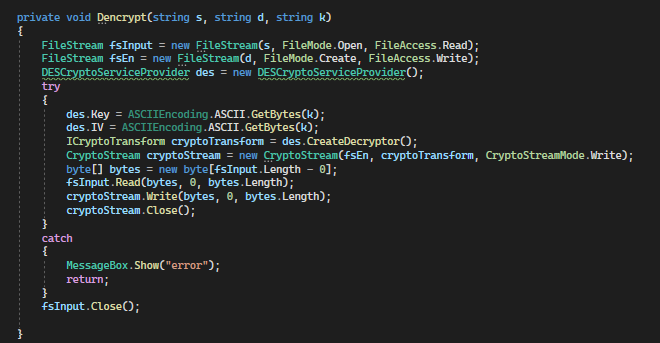


Рисунок 3.2 – Код для дешифрования

Исследуемый метод шифрования и ключевая информация – в соответствии с вариантом из таблицы:

|  |  |  |
| --- | --- | --- |
| Вариант | Алгоритм | Ключ |
| **1** | DES | Первые 8 символов собственных фамилииимени |

**Ключ: МакрицаФ**

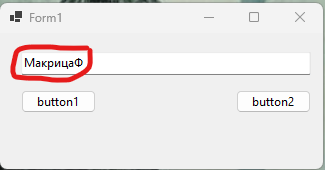
****

Рисунок 3.3 – введение ключа в программу

**Исходный текст:**

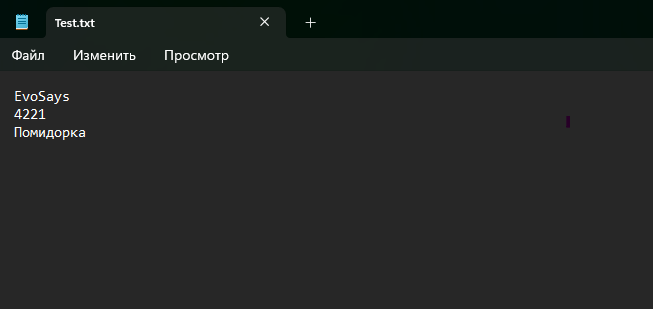
****

Рисунок 3.4–Исходный текст

**Зашифрованное сообщение**

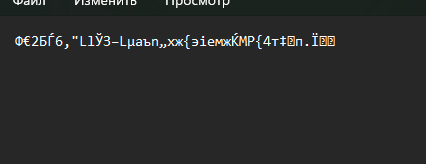


Рисунок 3.5 – Зашифрованное сообщение

**Дешифрованное сообщение**

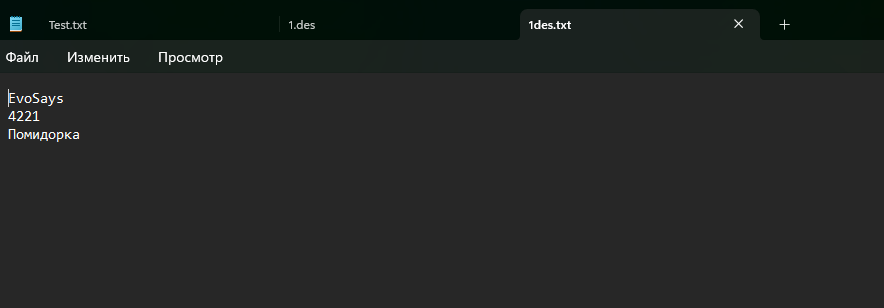
****

Рисунок 3.6 – Дешифрованное сообщение

Время шифрования теста с 1701 символов равна 13 мс.

Время дешифрования теста с 1701 символов равна 5 мс.

Время шифрования теста с 2255 символов равна 16 мс.

Время дешифрования теста с 2255 символов равна 2 мс.

Время шифрования теста с 3958 символов равна 17 мс.

Время дешифрования теста с 3958 символов равна 2 мс.

Рисунок 8– График времени выполнения операций шифрования и расшифрования текста

Пошаговый анализ лавинного эффекта с подсчетом количества изменяющихся символов по отношению к исходному слову.

исходное слово " Привет" было зашифровано алгоритмом, и результатом стало слово DЌ°юл„$®‘е1ЬУ•Ї , то можно проанализировать изменения каждого символа:

Первый символ П заменился на символ DЌ°

Второй символ р заменился на символ юл„

Третий символ и заменился на символ $®‘

Четвертый символ в заменился на символ е1Ь

Пятый символ е заменился на символ У•Ї

Шестой символ т заменился на <-тут пробел

2. Проанализировать влияние слабых ключей (табл. 2.1) и полу слабых ключей (табл. 2.2) на конечный результат зашифрования и на лавинный эффект.

Таблица 2.1

**Слабые ключи DES**

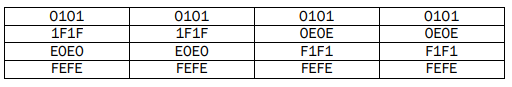
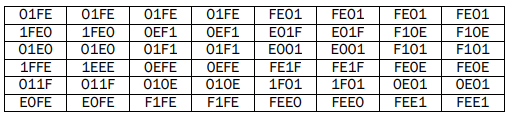
****

Таблица 2.2

**Полу слабые ключи DES**

****

Результат зашифрования текста при помощи слабого ключа:

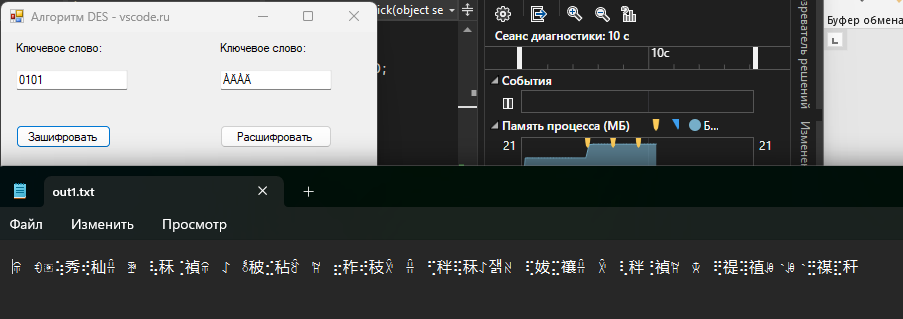


Рисунок 3.7 – Результат зашифрования текста при помощи слабого ключа 0101

Результат расшифровки текста:

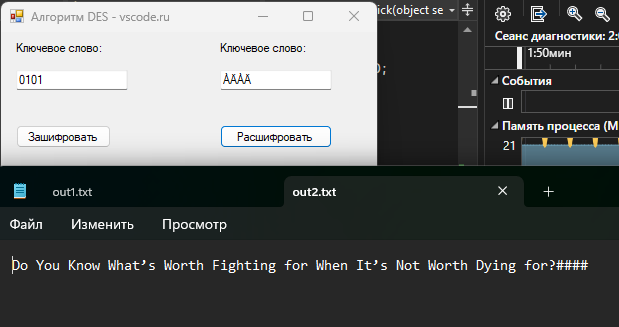


Рисунок 3.8 – Результат расшифровки текста

Результат зашифрования текста при помощи полу слабого ключа:

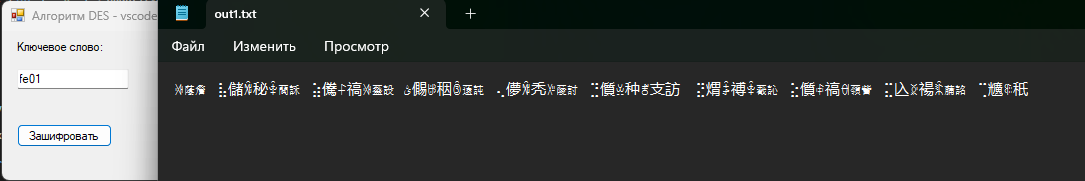


Рисунок 3.9 – Результат зашифрования текста при помощи полу слабого ключа fe01

Результат расшифровки текста:

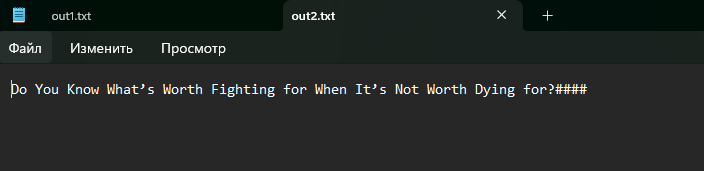


Рисунок 4 – Результат расшифровки текста

* довести исходное сообщение до такого размера (в битах), чтобы оно нацело делилось на размер блока (sizeOfBlock = 128 бит);
* разделить исходное сообщение на блоки;
* довести длину ключа до длины половины блока;
* перевести ключ в бинарный формат (в нули и единицы);
* провести над каждым блоком прямое преобразование сетью Фейстеля в течении 16-ти раундов. После каждого раунда необходимо выполнять циклический сдвиг ключа на заданное количество символов;
* соединить все блоки вместе; таким образом получим сообщение, зашифрованное алгоритмом DES.

Расшифровка DES производится по аналогии. Используется обратное преобразование сетью Фейстеля.

3. Оценить степень сжатия (используя любой доступный архиватор) открытого текста и соответствующего зашифрованного текста. Дать пояснения к полученному результату.



Рисунок 5 – Результат сжатия зашифрованного текста



Рисунок 6 – Результат сжатия открытого текста

Степень сжатия исходного теста равна: или же

Полученный результат показывает, насколько сильно размер зашифрованного текста уменьшился по сравнению с открытым текстом. Чем выше степень сжатия, тем лучше сжатие и тем меньше размер зашифрованного текста по сравнению с открытым. Однако, следует учитывать, что для DES не является главной целью сжатие, а задача - защита информации.

Вывод:

Я изучил и приобрёлл практические навыки разработки и использования приложений для реализации блочных шифров (рассчитана на 4 часа аудиторных занятий).