Министерство образования Республики Беларусь

Учреждение образование

«Белорусский государственный технологический университет»

Кафедра программной инженерии

**Отчет к лабораторной работе**:

«Исследование блочных шифров»

Выполнил:

студент 3 курса 4 группы ФИТ

Трусов Всеволод Сергеевич

Минск 2023

**Теоретические сведения**

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

В качестве начального варианта нового алгоритма рассматривался Lucifer

разработка компании IBM начала семидесятых годов. В основе указанного алгоритма использовались два запатентованных [1971 г](https://wiki2.org/ru/1971_%D0%B3%D0%BE%D0%B4). [Хорстом Фейстел](https://wiki2.org/ru/%D0%A4%D0%B5%D0%B9%D1%81%D1%82%D0%B5%D0%BB%D1%8C%2C_%D0%A5%D0%BE%D1%80%D1%81%D1%82)ем (Horst Feistel) устройства, реализующие различные [алгоритмы](https://wiki2.org/ru/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B) [шифрования](https://wiki2.org/ru/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5),

позже получившие *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 году](https://wiki2.org/ru/1987_%D0%B3%D0%BE%D0%B4) были разработаны алгоритмы [FEAL](https://wiki2.org/ru/FEAL) и [RC2](https://wiki2.org/ru/RC2). Сети Фейстеля получили широкое распространение в [1990-е годы](https://wiki2.org/ru/1990-%D0%B5) — в годы появления таких алгоритмов, как [*Blowfish*](https://wiki2.org/ru/Blowfish)(1993), [TEA](https://wiki2.org/ru/TEA) (1994), [RC5](https://wiki2.org/ru/RC5) (1994), [CAST-128](https://wiki2.org/ru/CAST-128) (1996), [XTEA](https://wiki2.org/ru/XTEA) (1997), [XXTEA](https://wiki2.org/ru/XXTEA) (1998), [RC6](https://wiki2.org/ru/RC6) (1998) и других. На основе сети Фейстеля в [1990 году](https://wiki2.org/ru/1990_%D0%B3%D0%BE%D0%B4) в СССР был принят в качестве [ГОСТ 28147-89](https://wiki2.org/ru/%D0%93%D0%9E%D0%A1%D0%A2_28147-89) стандарт шифрования.

Предполагалось, что DES будет сертифицироваться каждые 5 лет. Срок

действия последнего сертификата на территории США истек практически к концу 20 века. 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, т. е. являются комбинационными. БШ относятся также к классу симметричных.

[Блочное зашифрование](https://www.sites.google.com/site/anisimovkhv/learning/kripto/lecture/tema3#klassCS) (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*.

**Сеть Фейстеля**

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 (*L*0) и правый (*R*0). Далее в каждом *i*-ом раунде выполняются преобразования в соответствии с формальным представлением ячейки сети Фейстеля

**Базовые операции сложения чисел в блочных шифрах**

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 2*n*). Понятно, что числа *А* и *В* также являются *n*-разрядных.

Реализация второй из указанных операций является более сложной.

Вспомним основные ее особенности.

Во-первых. Самое большое слагаемое меньше 2*n*. Например, при *n=*3 самое большое слагаемое в двоичном виде – это 111 (или 7), а 2*n* = 8.

Во-вторых. Результатом сложения также должно быть *n*-разрядное число.

В-третьих. Побитовое сложение предусматривает известную взаимосвязь между соседними символами (порядками).

В-пятых. В силу известных правил модулярной1 арифметики результат вычисления *А* + *В* (mod 2*n*) – это остаток от деления: (*А* + *В*)/ 2*n*.

Рассмотрим примеры.

*Пример 1*. Пусть *n=*4, *А* = 9, *В =* 7, т. е. 2*n* = 16. Легко подсчитать, *А* + *В =* 16; 9 + 7 (mod 16) = 0.

Представим слагаемые в двоичной форме: *А* = 1001, *В =* 0111; *А* + *В =*1001

+ 0111 = 1**0000**. Для получения нужного результата – вычисления 9 + 7 (mod

1. – следует взять младшие 4 разряда (*n=*4) суммы: 0000.

*Пример 2*. Пусть *n=*4, *А* = 4, *В =* 10. Выполним вычисления по аналогии с примером 1.

Получаем *А* + *В =* 14, результирующая сумма меньше 16; 14 (mod 16) = 14. Представим слагаемые в двоичной форме: *А* = 0100, *В =* 1010; *А* + *В =*0100

+ 1010 = **1110**. Таким образом, итоговое число состоит из требуемых 4-х разрядов.

*Пример 3*. Пусть *n=*4, *А* = 10, *В =* 11. Их сумма по модулю 16 дает 5. В двоичной форме: 1010 + 1011 = 1**0101**. Искомое двоичное число – 4 младших разряда.

Если для данных в каждом из рассмотренных примеров мы проведем операцию деления (*А* + *В*)/ 24, получим те же 4 бита (выделены жирным).

**Алгоритм DES**

В силу причин, перечисленных в п.5.1.1, данный алгоритм является, по- жалуй, наиболее исследованным.

Алгоритм строится на основе сети Фейстеля.

Входной блок данных, состоящий из 64 бит, преобразуется в выходной блок идентичной длины. В алгоритме широко используются *рассеивания* (под- становки) и *перестановки* битов текста, о которых мы упоминали выше. Ком- бинация 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 дан- ных на правую (*R*0) и левую (*L*0) половины длиной по 32 бита выполняются 16 раундов одинаковых действий (см. рис. 5.5.). Функционал этих действий по- дробно рассмотрен в п. 5.1 пособия [2].

В таблице 5.1 показан принцип первоначальной перестановки разрядов (*IP*) входного 64-битового слова.

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

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



Рисунок 1.1 Общая схема алгоритма DES

Вначале правая часть блока *R*i расширяется до 48 битов, используя таблицу, которая определяет перестановку плюс расширение на 16 битов. Эта операция приводит размер правой половины в соответствие с размером ключа для выполнения операции XOR. Кроме того, за счет выполнения этой операции быстрее возрастает зависимость всех битов результата от битов исходных данных и ключа (это называется «*лавинным эффектом*»).

После выполнения перестановки с расширением для полученного 48-бит- ного значения выполняется операция XOR с 48-битным подключом *K*i. Затем полученное 48-битное значение подается на вход блока подстановки *S* (от англ. *S*ubstitution – 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 этапом 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. *P*ermutation – перестановка), которая не зависит от используемого ключа. Целью перестановки является максимальное переупорядочивание битов такое, 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-битового блока данных. Затем левая и правая половины меняются местами, и начинается следующий раунд (см. рис. 5.6).

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 (см. лабораторные работы №№ 4-6 из [1]). 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 *K*i. Ключ *K*16 используется в первом раунде, *K*1 – в последнем.

После последнего раунда процесса расшифрования две половины выхода меняются местами так, чтобы вход заключительной перестановки был состав- лен из подблоков *R*16 и *L*16. Выходом этой стадии является расшифрованный текст.

***Слабые и полуслабые ключи***. Из-за того, что первоначальный ключ изменяется при получении подключа для каждого раунда алгоритма, определенные

первоначальные ключи являются *слабыми* [4]. Вспомним, что первоначальное значение разделяется на две половины, каждая из которых сдвигается независимо. Если все биты каждой половины равны 0 или 1, то для всех раундов алгоритма используется один и тот же ключ. Это может произойти, если ключ состоит из одних 1, из одних 0, или если одна половина ключа состоит из одних 1, а другая – из одних 0.

***Криптоанализ DES***. *Дифференциальный криптоанализ* базируется на таблице неоднородных дифференциальных распределений *S*-блоков в блочном шифре. Криптоанализ шифртекстов на основе рассматриваемого стандарта «работает» с парами шифртекстов, открытые тексты которых имеют определенные разности, как это отмечалось в материалах к лабораторной работе № 2. Метод анализирует эволюцию этих разностей в процессе прохождения открытых текстов раундов DES при шифровании одним и тем же ключом.

Для DES термин «разность» определяется с помощью операции XOR. Затем, используя разности полученных шифртекстов, присваивают различные вероятности различным ключам. В процессе дальнейшего анализа следующих пар шифртекстов один из ключей станет наиболее вероятным.

*Линейный криптоанализ*. Для того, чтобы найти линейное приближение для DES нужно найти «хорошие» однораундовые линейные приближения и объединить их. Обратим внимание на *S*-блоки. У них 6 входных битов и 4 выходных. Входные биты можно объединить с помощью операции XOR 63 способами (26 - 1), а выходные биты – 15 способами. Теперь для каждого *S*-блока можно оценить вероятность того, что для случайно выбранного входа входная комбинация XOR равна некоторой выходной комбинации XOR. И т.д.

DES давно характеризуется низкой криптостойкостью: в январе 1999 г. закодированное посредством DES сообщение было взломано с помощью связанных через Internet в единую сеть 100 тыс. персональных компьютеров за 24 часа. Данному алгоритму присуща проблема так называемых «слабых» и «частично слабых» ключей [4].

Основное достоинство DES – относительно высокая скорость (из-за малой длины ключа); бесплатное распространение по всему миру; общедоступность и отсутствие необходимости лицензионных отчислений.

Модификацией DES является 3DES. Создан У. Диффи, М. Хеллманом, У. Тачманном в 1978 г.

Формальная запись:

*C*i = *f* (*M*j,(DES (*К*3, (DES(*К*2, (DES, *К*1)))))).

Расшифрование происходит, как и в простом DES, в обратном порядке по отношению к процедуре зашифрования.

3DES с тремя ключами реализован во многих Интернет-приложениях. Например, в PGP (Pretty Good Privacy) – позволяет выполнять операции шифрования и цифровой подписи сообщений, файлов и другой информации, представленной в электронном виде, например, на жёстком диске ); в S/mime для обеспечения криптографической безопасности электронной почты.

3DES используется при управлении ключами в стандартах ANSI X9.17 (метод генерации 64-битных ключей) и ISO 8732 (управление ключами в банковском деле), а также в PEM (Privacy Enhanced Mail).

**Практическая часть**

В данной лабораторной работе необходимо было произвести программно реализовать алгоритм DES EDE3: Общий интерфейс программы представлен на рисунке 1.

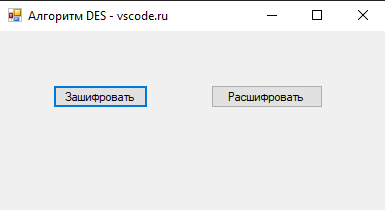


Рисунок 1 – Общий интерфейс программы

Пример работы шифрования DES3 приведен на рисунке (Рисунок 2)

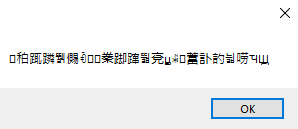


Рисунок 2 – Шифрование DES EDE3

Пример расшифрования DES EDE3 приведен на рисунке 3.

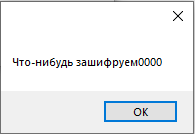


Рисунок 3 – Расшифрование DES EDE3

Вывод: В данной лабораторной работе были изучены теоретические сведения об блочных шифрах, а также разработано программное средство в соответствии с заданием реализующее работу алгоритма DES EDE3.