Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

[Кафедра информационных](https://www.belstu.by/fakultety/fit/vm) систем и технологий

Специальность 1-40 01 01 Программное обеспечение информационных технологий

**Отчёт по лабораторной работе №7**

по дисциплине Информационная безопасность

Тема: Исследование блочных шифров

        Студент: Корнелюк. В. В.

ФИТ 3 курс 4 группа

Преподаватель: Нистюк О.А.

Минск 2025

**Лабораторная работа №7**

**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

# Теоретические сведения

Все 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.

В основе криптостойкости блочных шифров лежит идея К. Шеннона о представлении составного шифра таким образом, чтобы он обладал двумя важными свойствами: рассеиванием и перемешиванием. Рассеивание должно скрыть отношения между зашифрованным текстом и исходным текстом. Рассеивание подразумевает, что каждый 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

Алгоритм DES строится на основе сети Фейстеля. Входной блок данных, состоящий из 64 битов, преобразуется в выходной блок идентичной длины. В алгоритме широко используются рассеивания (подстановки) и перестановки битов текста. 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.

# Ход работы

Для выполнения шифрования и дешифрования с помощью алгоритма DES использовалась библиотека node-forge.

Функция encrypt принимает на вход шифруемый текст, ключ и выполняет зашифрование. Код функции представлен на рисунке 2.1.

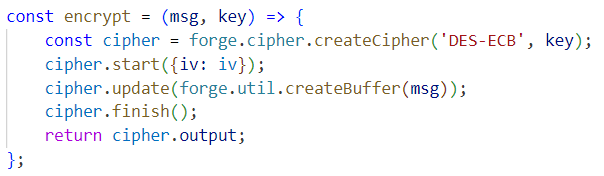


Рисунок 2.1 – Код функции шифрования

Функция EEE2Encrypt для шифрования с помощью алгоритма DES-EEE2 принимает на вход шифруемы текст и два ключа. Код функции представлен на рисунке 2.2

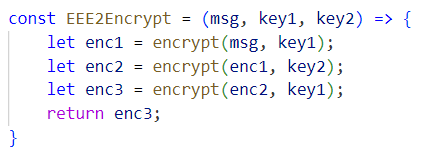


Рисунок 2.2 – Код функции шифрования

Для дешифрования была разработаны функции decrypt и EEE2Decrypt. Код функций представлен на рисунках 2.3 и 2.4 соответственно.

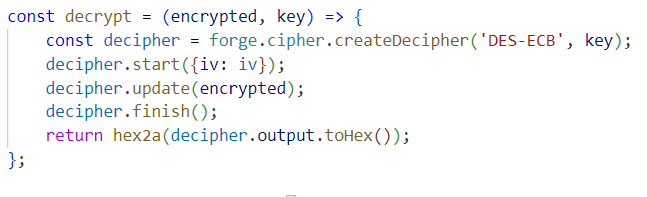


Рисунок 2.2 – Код функции дешифрования

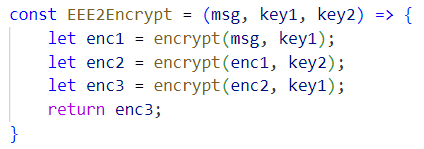


Рисунок 2.2 – Код функции дешифрования

Результат работы приложения с исходным текстом «valik» и ключами «kroneliu» и «valentin» приведен на рисунке 2.3.

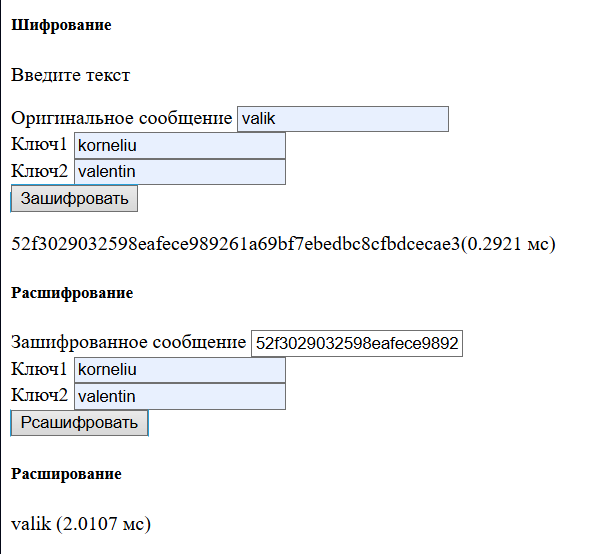


Рисунок 2.3 – Результат шифрования и дешифрования текста

Функция avalancheEffect принимает исходный текст и шифрует его. Затем изменяет один бит исходного текста, снова шифрует его и подсчитывает количество измененных бит и процентах. Код функции представлен на рисунке 2.4.



Рисунок 2.4 – Код функции, рассчитывающей лавинный эффект

Функция принимает исходный текст и два ключа. Сначала выполняется первое зашифрование и результат заносится в переменную. Аналогично выполняются второе и третье шифрование. После этого в исходном тексте меняется последний бит и снова выполняется зашифрование три раза и результата шифрования выносится в переменные. После этого результат шифрования на каждом шаге до изменения последнего бита и после сравнивается и высчитывается процент отличающихся битов. Результат анализа лавинного эффекта представлен на рисунке 2.5, где step1 – процент отличающихся битов после первого зашифрования, step2 и step3 – после второго и третьего шага соответсвенно.

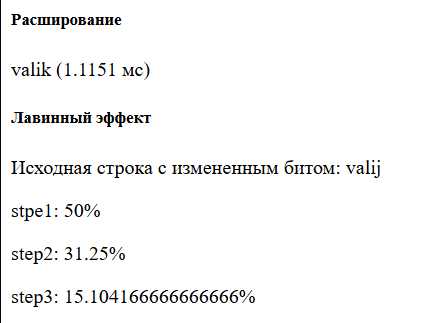


Рисунок 2.5 – Эффект лавины для исходного текста «korneliukvalentine»

Кроме того, нужно было проверить влияние слабых и полуслабых ключей на лавинный эффект. Пример работы приложения со слабым ключом 0101010101010101 приведен на рисунке 2.6.

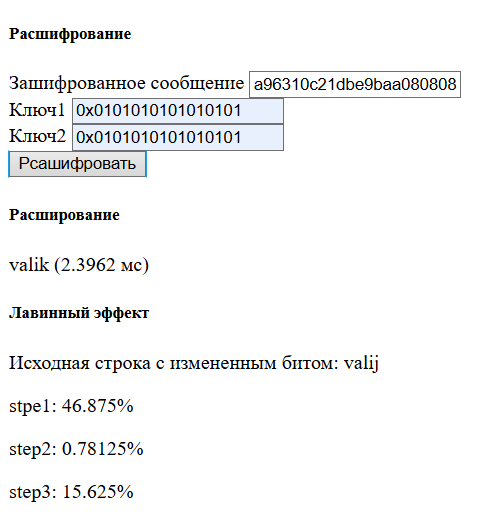


Рисунок 2.6 – Эффект лавины для исходного текста при шифровании со слабым ключом

Пример работы приложения со полуслабым ключом E01FE01FE01FE01F приведен на рисунке 2.7.

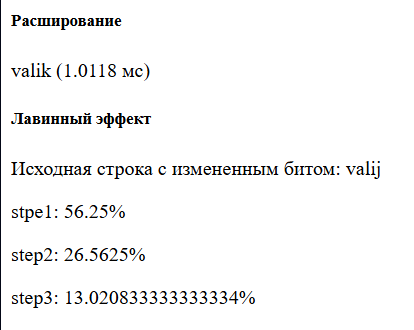


Рисунок 2.7 – Эффект лавины для исходного текста при шифровании с полуслабым ключом

Можно заметить, что эффект лавины значительно уменьшается, поскольку при разделении первоначального значения ключа на две половины, получаются части, в которые все биты равны 0 или 1, которые в последующем не изменяются в каждом раунде алгоритма. В случае полуслабых ключей, каждый из подключей используется 8 раз, что также снижает действие лавинного эффекта.

Также было необходимо измерить скорость шифрования и дешифрования. Шифрование и дешифрование алгоритмом DES выполняется достаточно быстро из-за малой длины ключа.

Далее, была проведена оценка степени сжатия открытого и зашифрованного текста. Результат приведен на рисунке 2.8.

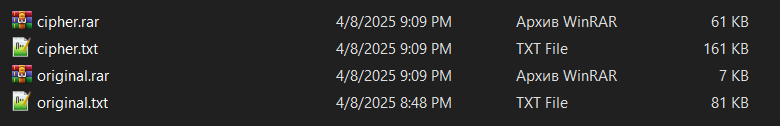


Рисунок 2.8 – Степень сжатия текста

Можно заметить, что исходный текст сжался почти на 92%, в то время как зашифрованный только на 60%. Это происходит потому, что шифрование преобразует информацию таким образом, что создается псевдослучайная последовательность битов, к которым практически невозможно применить существующие алгоритмы сжатия.

**Вывод:** в ходе выполнения лабораторной работы был изучен принцип работы блочного алгоритма DES. Также было разработано приложение для реализации зашифрования и расшифрования с помощью этого алгоритма. Была оценена скорость шифрования и расшифрования алгоритмом DES-EEE2.