Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 01 01 Программное обеспечение информационных технологий

**Отчёт по лабораторной работе №7**

**«Исследование блочных шифров»**

**Вариант 3**

Исполнитель:

Студент 3 курса группы 4

Гурина К. С.

Руководитель:

Ассистент Сазонова Д. В.

1. **Цель и задачи работы**

**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

**2. 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.

**3. Ход работы**

**Практическое задание:**

1. Разработать авторское приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться готовыми библиотеками либо программными кодами, реализующими некоторые блочные алгоритмы, из приложения в. 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.

2. Проанализировать влияние слабых ключей (табл. 5.3) и полуслабых ключей (табл. 5.4) на конечный результат зашифрования и на лавинный эффект.

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

4. Результаты оформить в виде отчета по установленным правилам.

**Ход работы**

В начале работы определяются несколько констант, необходимых для реализации алгоритма DES (Data Encryption Standard). Эти константы включают перестановочные таблицы, S-блоки, таблицы расширения и сжатия ключа, начальной и финальной перестановки, а также таблицу количества сдвигов для генерации ключей:

* PC1 и PC2 - таблицы для сжатия ключа.
* IP и FINAL\_IP - таблицы начальной и финальной перестановок.
* E - таблица расширения.
* S - S-блоки.
* P - таблица перестановки.
* NUM\_OF\_LEFT\_SHIFTS - таблица сдвигов для генерации ключей.

Были разработаны вспомогательные функции:

* chunkString(str, len) - разбивает строку на части длиной len.
* hexToBin(hex) - преобразует шестнадцатеричное число в двоичное.
* decToBin(dec) - преобразует десятичное число в двоичное.
* binaryToHex (bin) - преобразует двоичное число в шестнадцатеричное.
* bin(key) - преобразует строку шестнадцатеричных символов в строку двоичных символов.
* shiftString(str, shift) - сдвигает строку на заданное количество позиций.

Для генерации ключей была разработана функция keySchedule. Функция keySchedule(key) генерирует 16 подключей из основного ключа. Сначала применяется начальная перестановка PC1 к ключу, потом ключ делится на две половины C0 и D0. В цикле выполняются сдвиги и применяются перестановки PC2 для получения 16 подключей. Код функции представлен на рисунке 3.1.



Рисунок 3.1 – Код функции keySchedule

Для расширения 32-битный блока до 48 бит с помощью таблицы E используется функция expandBlock(block), представленная на рисунке 3.2.



Рисунок 3.2 – Код функции expandBlock

Функция stringXOR(str1, str2, len) выполняет побитовое сложение двух строк str1 и str2 длиной len.

Функция sBoxOutput(bits) обрабатывает 48-битный блок через S-блоки и возвращает 32-битный результат.

Функция des(msg, key, subkeys) выполняет шифрование или расшифрование сообщения msg с помощью ключа key и подключей subkeys. Сначала сообщение подвергается начальной перестановке IP. Затем выполняется 16 раундов алгоритма Feistel, где данные разделяются на левую и правую половины, и для каждой половины выполняются операции расширения, XOR с подключом, обработка через S-блоки и перестановка P. Наконец, результат подвергается финальной перестановке FINAL\_IP и преобразуется в шестнадцатеричную строку. Код функции можно увидеть на рисунке 3.3.

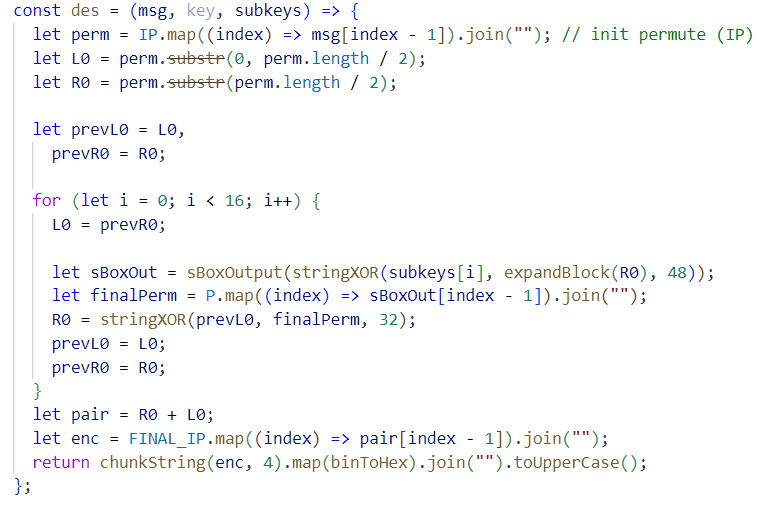


Рисунок 3.3 – Код функции des

Функция encode(msg, key) - выполняет шифрование сообщения msg с помощью ключа key и функции des. Функция decode(msg, key) – выполняет расшифрование сообщения msg с помощью ключа key и реверсированных подключей.

Функция des3enc (msg, key1, key2, key3) – реализует тройное шифрование 3DES (шифрование с ключом key1, расшифрование с ключом key2, шифрование с ключом key3). Код функции представлен на рисунке 3.4.

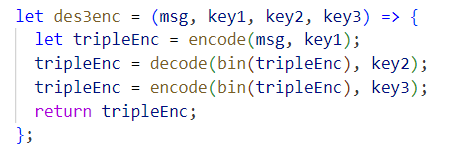


Рисунок 3.4 – Код функции des3enc

Функция des3dec (enc, key1, key2, key3) – реализует тройное расшифрование 3DES (расшифрование с ключом key3, шифрование с ключом key2, расшифрование с ключом key1).

Функция avalancheEffect используется для вычисления лавинного эффекта. Лавинный эффект в криптографии означает, что небольшое изменение в исходных данных или ключе приводит к значительным и непредсказуемым изменениям в результате шифрования. Это свойство важно для обеспечения безопасности криптографических алгоритмов, так как оно затрудняет анализ шифротекста для восстановления исходных данных или ключа.Функция 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 3.5.



Рисунок 3.5 – Код функции avalancheEffect

Для проверки работоспособности приложения в соответствии с условием инициализируем входные данные для алгоритма (рисунок 3.6)

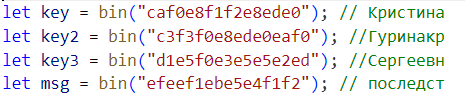


Рисунок 3.6 – Инициализация входных параметров

Результат работы приложения можно увидеть на рисунке 3.7.

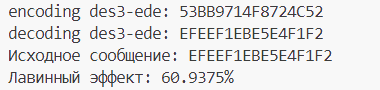


Рисунок 3.7 – Результат работы приложения

Кроме того, нужно было проверить влияние слабых и полуслабых ключей на лавинный эффект. Из-за того, что первоначальный ключ изменяется при получении подключа для каждого раунда алгоритма, определенные первоначальные ключи являются слабыми. 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. Заданные слабые ключи представлены на рисунке 3.8.

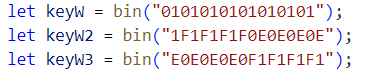


Рисунок 3.8 – Инициализация слабых ключей

Лавинный эффект при использовании слабых ключей показан на рисунке 3.9

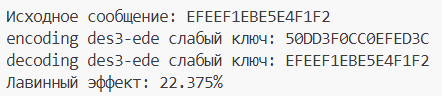


Рисунок 3.9 – Эффект лавины при шифровании со слабыми ключам

Лавинный эффект при использовании полуслабых ключей показан на рисунке 3.10.

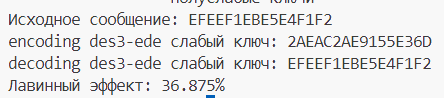


Рисунок 3.10 – Эффект лавины при шифровании с полуслабыми ключам

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 3.11.

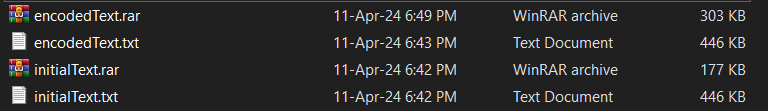
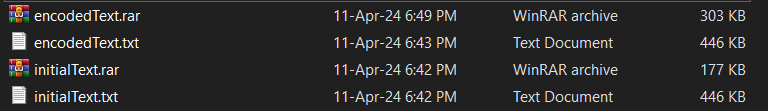


Рисунок 3.11 – Степень сжатия текста

Можно заметить, что исходный текст сжался почти на 60%, в то время как зашифрованный только на 30%. 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 лабораторной работы была исследована реализация блочного шифра DES-EDE3. Были изучены основные этапы шифрования и расшифрования, включая генерацию ключей, перестановки, использование S-блоков и операции XOR. Реализация алгоритма позволила успешно выполнить шифрование и расшифрование сообщения, что подтверждает корректность работы функций. Также был оценен эффект лавины при использовании слабых и полуслабых ключей.