Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Исследование блочных шифров**

Студент: Раченок И.А.

ФИТ 3 курс 4 группа

Преподаватель:

Сазонова Дарья Владимировна

1. **Описание приложения и заданий**

Приложение должно выполнять следующие функции:

* 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;

1. **Выполнение поставленных задач**

В соответствии с вариантом был выбран 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-битногоблока данных на правую (R0) и левую (L0) половины длиной по 32 бита выполняются 16 раундов одинаковых действий. На входе текст расширяется до 64 битов и подвергается начальной перестановке, где биты в соответствии с таблицей меняют свой порядок бит, рисунок реализации в программе 2.1.

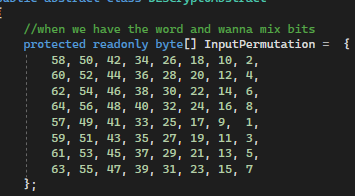


Рисунок 2.1 – Функция начальной перестановки

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

После выполнения перестановки с расширением для полученного расширенного 48-битного значения выполняется операция XOR с 48-битным подключом . Затем полученное 48-битное значение подается на вход блока подстановки S, результатом которой является 32-битное значение.

Подстановка выполняется в восьми блоках подстановки или восьми *S*-блоках. 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, 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.

После выполнения 16-раундового зашифрования 64-битного блока данных осуществляется конечная перестановка. Она является обратной к перестановке IP.

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 *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 составлен из подблоков *R*16 и *L*16. Выходом этой стадии является расшифрованный текст. Ниже на рисунке 2.2 отображены переменные преобразований, необходимые для алгоритма DES.

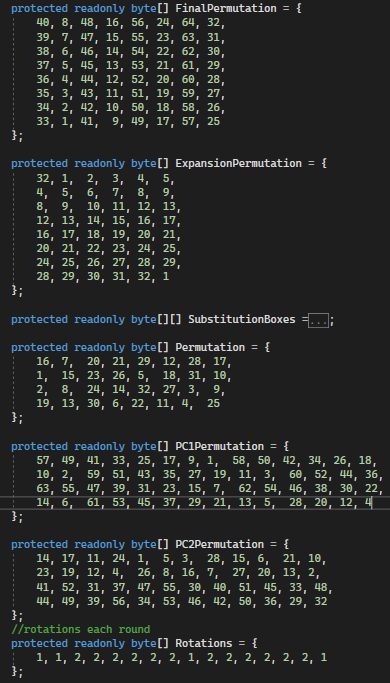


Рисунок 2.2 – Переменные преобразования, необходимые для алгоритма

Далее идет пояснение этих переменных. InputPermutation: Данный массив представляет начальную перестановку, применяемую к блоку входных данных перед началом алгоритма. FinalPermutation: Данный массив представляет конечную перестановку, применяемую к блоку выходных данных после завершения алгоритма. ExpansionPermutation: Данный массив представляет расширенную перестановку, применяемую к правой половине данных в процессе шифрования. SubstitutionBoxes: Это двумерный массив, представляющий 8 S-блоков, используемых в алгоритме DES. *S*-блоки используются для замены битов в процессе шифрования. Permutation: Данный массив представляет перестановку, применяемую к результату замен на *S*-блоках. PC1Permutation: Данный массив представляет перестановку, применяемую к начальному 64-битному ключу для генерации 56-битного ключа для каждого раунда шифрования. PC2Permutation: Данный массив представляет перестановку, применяемую к 56-битному ключу для генерации 48-битного подключа для каждого раунда шифрования. Rotations: Данный массив представляет количество циклических сдвигов влево, применяемых к половинкам 28-битного ключа на каждом раунде шифрования.

Далее перейдем к реализации функций (EncodeBlock) шифрования данными алгоритмом DES, алгоритм которого описан выше, рисунок 2.3.

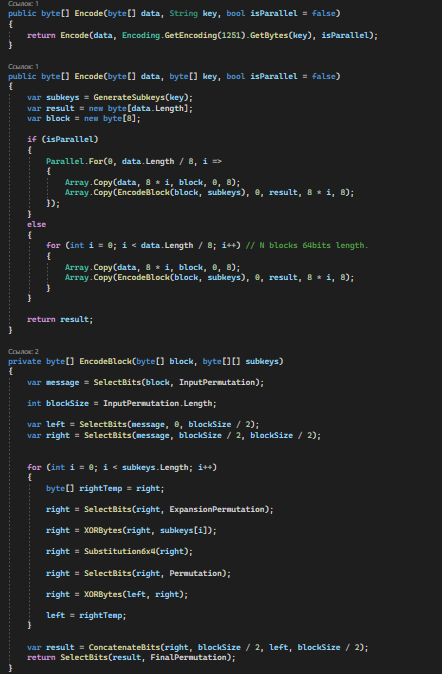


Рисунок 2.3 – EncodeBlock

Разберем теперь функцию расшифровки сообщения алгоритмом DES.

Алгоритм DES использует обратную последовательность операций для расшифровки сообщения. В процессе расшифровки, используется та же самая логика и те же самые компоненты, что и при шифровании, но операции применяются в обратном порядке.

Например, при расшифровке используется обратная конечная перестановка (FinalPermutation) для восстановления исходного блока данных. Расширенная перестановка (ExpansionPermutation) также используется для расширения правой половины данных перед применением обратных *S*-боксов и перестановок.

Также обратные операции применяются к ключам. PC1Permutation и PC2Permutation используются для генерации обратных ключей, и Rotations выполняют обратные циклические сдвиги вправо вместо влево.

Таким образом, процесс расшифровки DES заключается в применении тех же операций, что и при шифровании, но в обратном порядке для восстановления исходного сообщения.

Основная функция расшифровки представлена на рисунке 2.4.

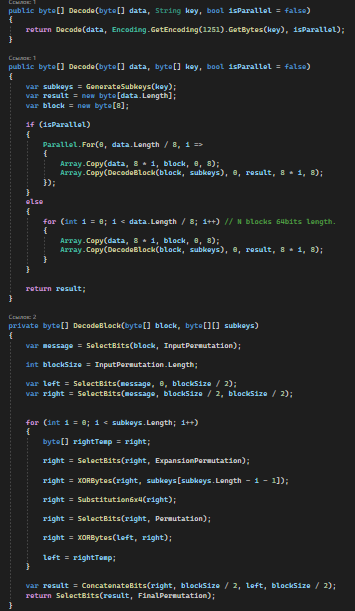


Рисунок 2.4 – DecodeBlock функция дешифровки

Основной код и результат работы алгоритма DES представлены на рисунках 2.5 и 2.6.

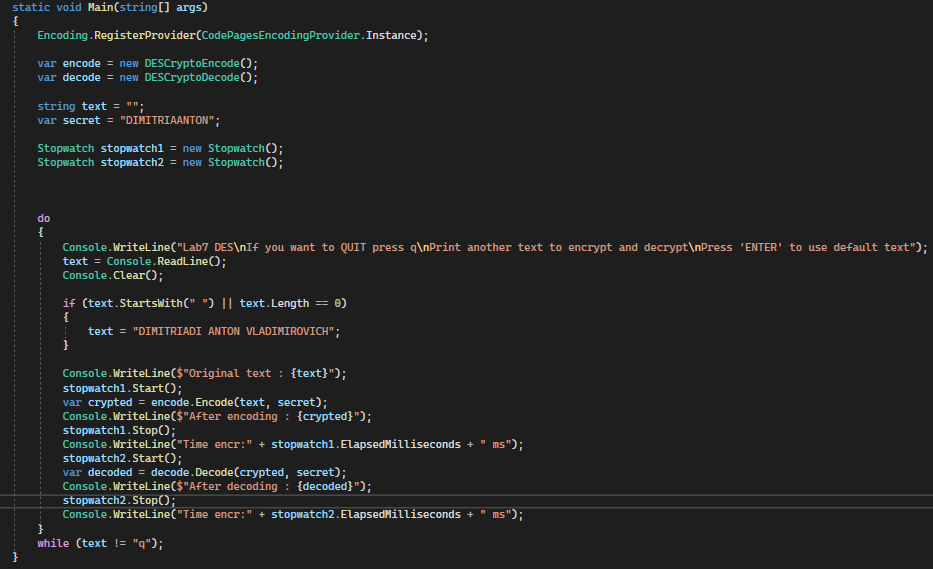


Рисунок 2.5 – Основной код

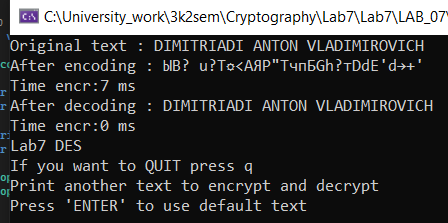


Рисунок 2.6 – Результат работы

Также ниже приведен оценочный график времени выполнения зашифрования и расшифрования блочного алгоритма, рисунок 2.7.

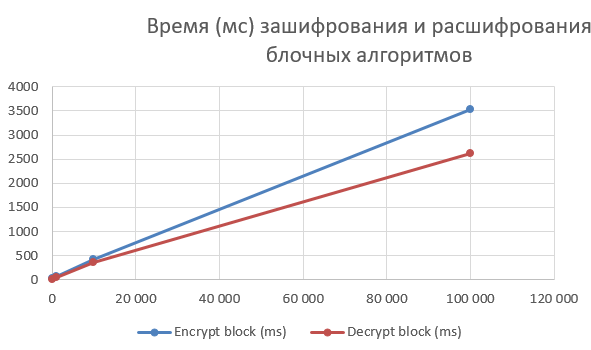


Рисунок 2.7 – Оценочный график

Лавинный эффект является феноменом, связанным с работой криптографических алгоритмов, особенно алгоритмов блочного шифрования, таких как DES (Data Encryption Standard) или AES (Advanced Encryption Standard). Он описывает свойство, при котором даже небольшое изменение входных данных приводит к значительным изменениям в выходных данных шифрования.

При использовании блочных шифров, данные разбиваются на блоки фиксированного размера, например, 64 или 128 бит. Каждый блок обрабатывается независимо от других блоков, используя определенный ключ и алгоритм шифрования. Лавинный эффект проявляется в том, что даже незначительное изменение в одном бите входных данных приводит к тому, что множество бит в выходных данных изменяются.

Этот эффект является желательным свойством криптографических алгоритмов, так как он усиливает стойкость шифрования. Для хорошего алгоритма шифрования, даже небольшие изменения входных данных должны приводить к случайным и непредсказуемым изменениям выходных данных. Это делает атаки на шифр с использованием методов анализа и статистики гораздо сложнее, поскольку изменения в данных не позволяют получить информацию о ключе или исходных данных.

Лавинный эффект достигается за счет использования нелинейных и необратимых операций, таких как подстановки и перестановки битов, внутри алгоритма шифрования. Эти операции распределяют изменения входных данных по всему блоку, что ведет к распространению эффекта и усилению случайности выходных данных.

Обеспечение сильного лавинного эффекта является одной из важных целей при разработке криптографических алгоритмов, поскольку это способствует повышению безопасности и защиты данных при шифровании.

Далее были проведена операция, которая посчитала на какое количество бит будут отличаться зашифрованные тексты, при отличии ключа на один символ, рисунок 2.8.

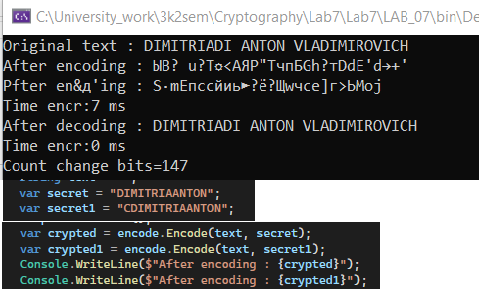


Рисунок 2.8 – Оценка лавинного эффекта

Алгоритм DES (Data Encryption Standard) имеет некоторые слабые ключи, которые могут привести к уязвимостям в процессе шифрования. Слабые ключи DES - это ключи, которые приводят к шифрованию данных с недостаточным уровнем безопасности.

Одним из примеров слабого ключа DES является ключ, состоящий из всех нулей (все 64 бита ключа установлены в 0). В этом случае, все раунды шифрования DES будут производить одни и те же преобразования для каждого блока данных, что делает шифрование предсказуемым и уязвимым к атакам.

Еще одним примером слабого ключа DES является ключ, состоящий из всех единиц (все 64 бита ключа установлены в 1). В этом случае также происходит повторение преобразований на каждом раунде, что делает шифрование уязвимым.

Кроме того, существуют несколько слабых ключей DES, которые обладают определенными свойствами, такими как повторение половин ключа или других структурных особенностей, которые могут существенно снизить безопасность шифрования.

Как можно увидеть на рисунке 2.9, пример слабого ключа, лавинный эффект будет меньшим чем при более стойком ключе, а это значит, что изменения между двумя зашифрованными текстами меньше, то есть проще взломать алгоритм.

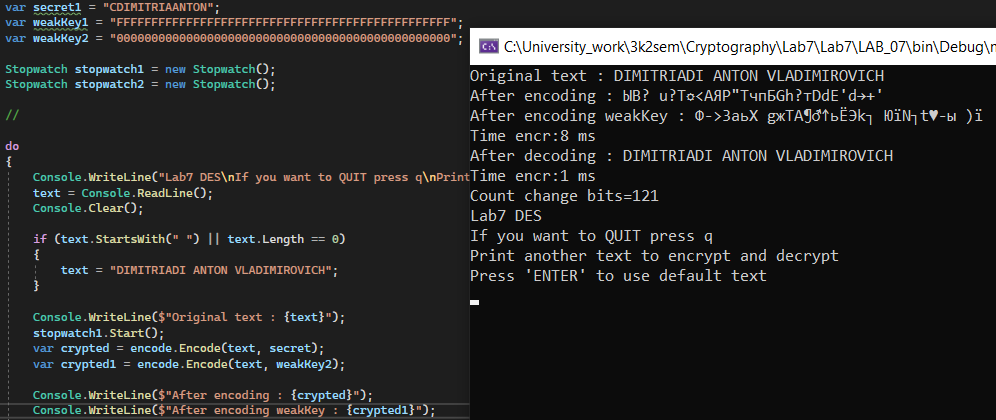


Рисунок 2.9 – Пример работы с слабым ключом

На рисунке 2.10 количество отличающихся битов будет большим чем со слабым ключом, но меньше чем с обычным, что тоже говорит нам о том, что такие ключи дают большие возможности к взлому алгоритма.

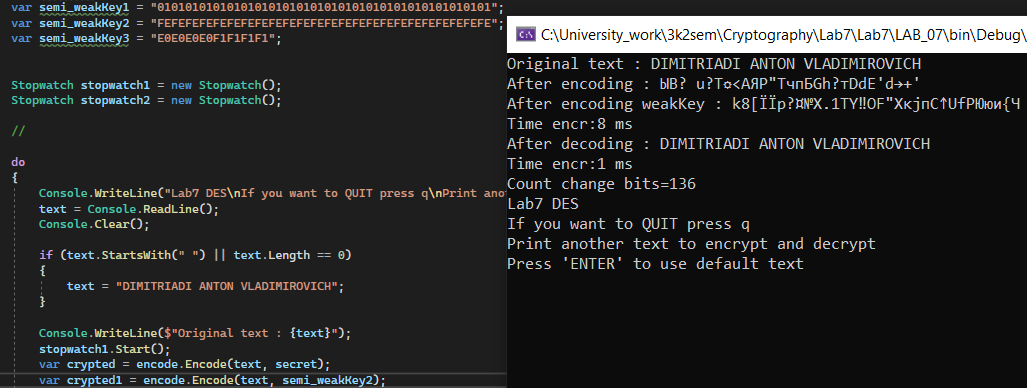


Рисунок 2.10 – Пример работы с полуслабым ключом

Также была проведена оценка размера сжатого файла и исходного на рисунке 2.11.

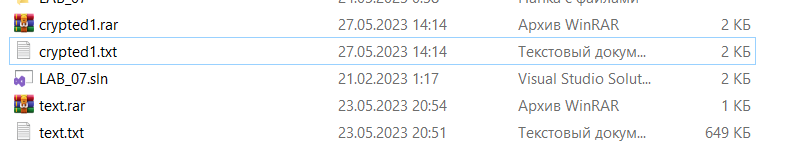


Рисунок 2.11 – Разница между архивом зашифрованного текста и исходного

Ка мы видим на рисунке, зашифрованный текст занимает больше байт чем исходный, а это связано с тем что, что процесс шифрования добавляет дополнительную информацию к исходному тексту, чтобы обеспечить безопасность. Эта дополнительная информация может включать вектор инициализации (IV), данные проверки целостности (MAC) или другие управляющие данные.

Кроме того, при блочном шифровании DES и подобных алгоритмах, размер блока обычно больше размера одного символа исходного текста. Например, DES работает с блоками размером 64 бита (8 байт), в то время как исходный текст может быть гораздо меньшего размера. Это означает, что даже если исходный текст небольшой, он будет дополнен до размера блока.

В результате, зашифрованный текст с использованием DES может занимать больше места, чем исходный текст, из-за добавленных данных шифрования, дополнений до размера блока и возможностей управления шифрованием. Это нормальное явление при использовании шифрования с блочными алгоритмами, и размер зашифрованных данных может быть больше размера исходных данных.

**Вывод**

В данный лабораторной работе были изучены и приобретены практические навыки разработки и использования приложений для реализации блочного шифра DES. Сделаны выводы о том, почему зашифрованный текст занимает больше места, чем исходный.

Разработано приложение, на языке программирования C#, для реализации задач, связанных с шифрованием данных.