Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Лабораторная работа № 7**

«Исследование блочных шифров»

Выполнил:

Студент: Герман А.Е.

ФИТ 3 курса 5 группы

Преподаватель: Савельева М.Г

Минск 2024

# **1 Описание приложения**

Приложение написано на языке программирования C# и позволяет зашифровать и расшифровать текстовый документ с помощью алгоритма 3DES-EDE2. Приложение реализует следующие операции:

* разделение входного потока данных на блоки требуемой длины с дополнением последнего блока нулями;
* преобразование ключевой информации по алгоритму MD5;
* пошаговый подсчет количества символов по отношению к исходному слову.

Также приложение позволяет оценивать время выполнения операций за(рас)шифрования и записать исходной текст и соответствующий ему зашифрованный текст в выходной файл, чтобы в последующем оценить степень их сжатия.

# **2 Методика выполнения расчетов**

В данной лабораторной работе была поставлена цель создания приложения, реализующее блочное шифрование 3DES-EDE2. На листингах 2.1-2.2 представлены методы, реализующий данную функциональность.

|  |
| --- |
| private static string Encode(string input, string key)  {  byte[] toEncryptArray = Encoding.UTF8.GetBytes(input);  var des = new DESCryptoServiceProvider();  des.Key = Encoding.UTF8.GetBytes(key);  des.Mode = CipherMode.ECB;  des.Padding = PaddingMode.Zeros;  var cTransform = des.CreateEncryptor();  byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0, toEncryptArray.Length);  des.Clear();  return Convert.ToBase64String(resultArray, 0, resultArray.Length);  }  private static string Decode(string input, string key)  {  byte[] toEncryptArray = Convert.FromBase64String(input);  var des = new DESCryptoServiceProvider();  des.Key = Encoding.UTF8.GetBytes(key);  des.Mode = CipherMode.ECB;  des.Padding = PaddingMode.Zeros;  var cTransform = des.CreateDecryptor();  byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0, toEncryptArray.Length);  des.Clear();  return Encoding.UTF8.GetString(resultArray);  } |

Листинг 2.1 – Методы **Encode/Decode**, реализующие алгоритм 3DES-EDE2

Сначала исходная строка разбивается на блоки фиксированной длины с дополнением последнего блока. Поскольку мы указали режим шифрования ECB, входной поток будет разбит на блкои длиной 8 байт (64 бит) по умолчанию. Режим дополнения Zeros указывает, что последний блок будет дополняться нулями. Также выполняется преобразование ключевой информации – переданный в параметрах ключ хешируется с помощью алгоритма MD5.

Данное программное средство должно реализовать алгоритм 3DES-EDE2: операции шифрование-расшифрование-шифрование, на первом и третьем шаге используется одинаковый ключ. Реализация данного алгоритма представлена на рисунке 2.2.

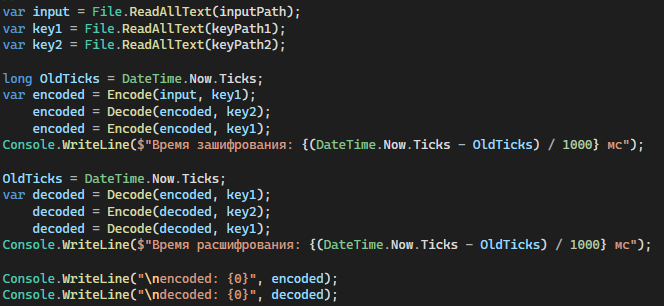


Рисунок 2.2 – Реализация алгоритма 3DES-EDE2

Для зашифрования и расшифования используются ключи key1, key2, считанные из файловой системы. Расшифрование происходит в обратном порядке к процедуре зашифрования, ключи также используются в обратном порядке.

Для зашифрования исходного текста использована простая маршрутная перестановка. Сначала исходная строка разбивается на подстроки длиной k (ключ), после чего последовательно считываются первый символ каждой подстроки, затем второй и т.д. по возрастанию. Реализация данного алгоритма представлена на рисунке 2.1.

# **3 Результаты работы приложения**

Для выполнения расчетов достаточно необходимо запустить приложение. Рисунки 3.1 – 3.2 показывают необходимые расчеты и вызовы методов, требуемые в данной лабораторной работе.

## **3.1 Шифрование по алгоритму**

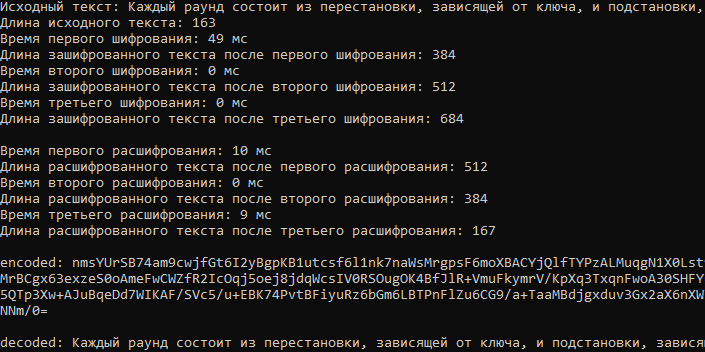


Рисунок 3.1 – Результат работы методов **Encode/Decode** с входным текстом (in.txt)

Была проведена оценка скорости выполнения операций зашифрования – 49 сек, и расшифрования – 10 сек. Из этого можно сделать вывод, расшифрование происходит в разы быстрее, так как использует уже кэшированные при зашифровании данные. В целом, у алгоритма достаточно высокая скорость выполнения, которая обусловлена малой длиной ключей key1, key2.

На каждом из трех этапов шифрования, при реализации алгоритма 3DES-EDE2, было подсчитано количество символов в шифротексте. Результат вычислений показан на рисунке 3.1. Проанализировав его, нетрудно заметить, что на каждом шаге количество символов вырастает почти в 2 раза, что обуславливает постоянно возрастающий «лавинный эффект» и рост размера файла с шифротекстом.

## **3.2 Сравнение размеров исходного и шифрованного текстов**

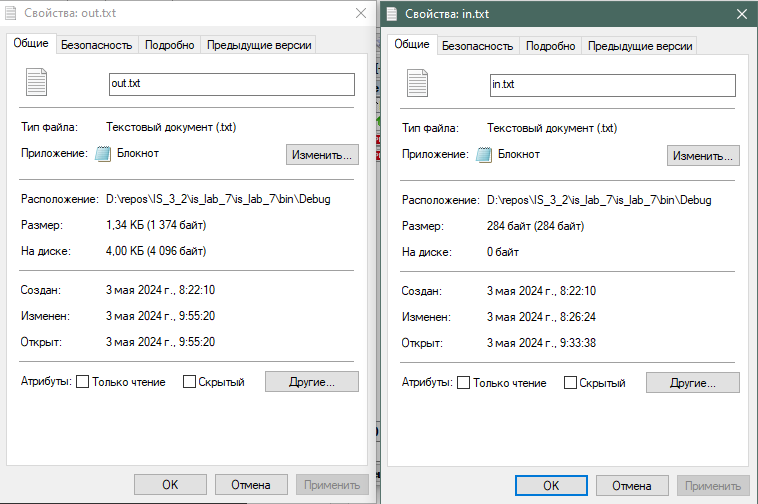


Рисунок 3.2 – Сравнение размеров исходного и шифрованного текста

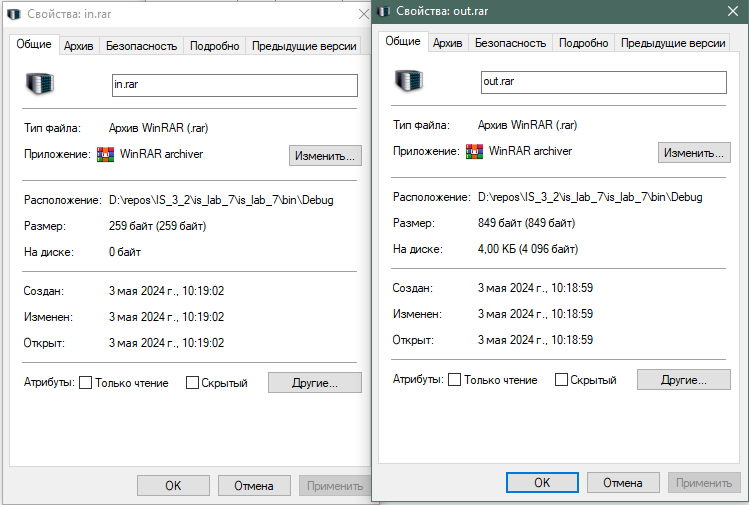


Рисунок 3.3 – Сравнение размеров сжатых открытого и шифрованного текста

Также, была оценена степень сжатия открытого текста (284 байт) и соответствующего зашифрованного текста (1,34 Кбайт). Такую ощутимую разницу можно объяснить тем, что с каждым вызовом зашифрования возрастает «лавинный эффект» – растет зависимость всех битов результата от битов исходных данных и ключа, а также растет количество символов в зашифрованном тексте по отношению к количеству символов исходного текста.

## **3.3 Проверка «лавинного эффекта» в 3DES**

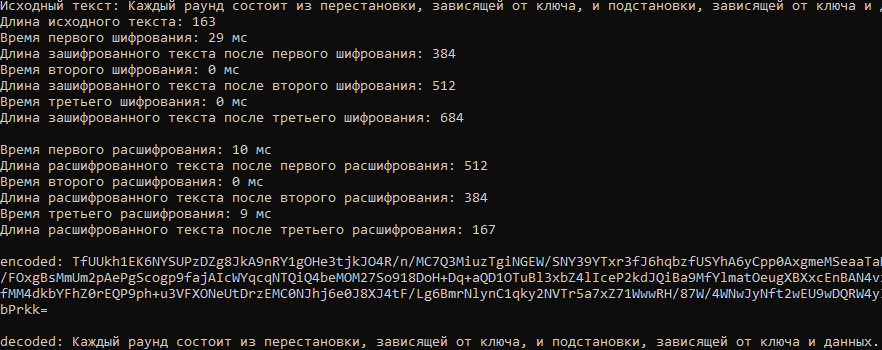


Рисунок 3.4 – Результат работы методов **Encode/Decode** с входным текстом (in.txt) и изменённым ключом

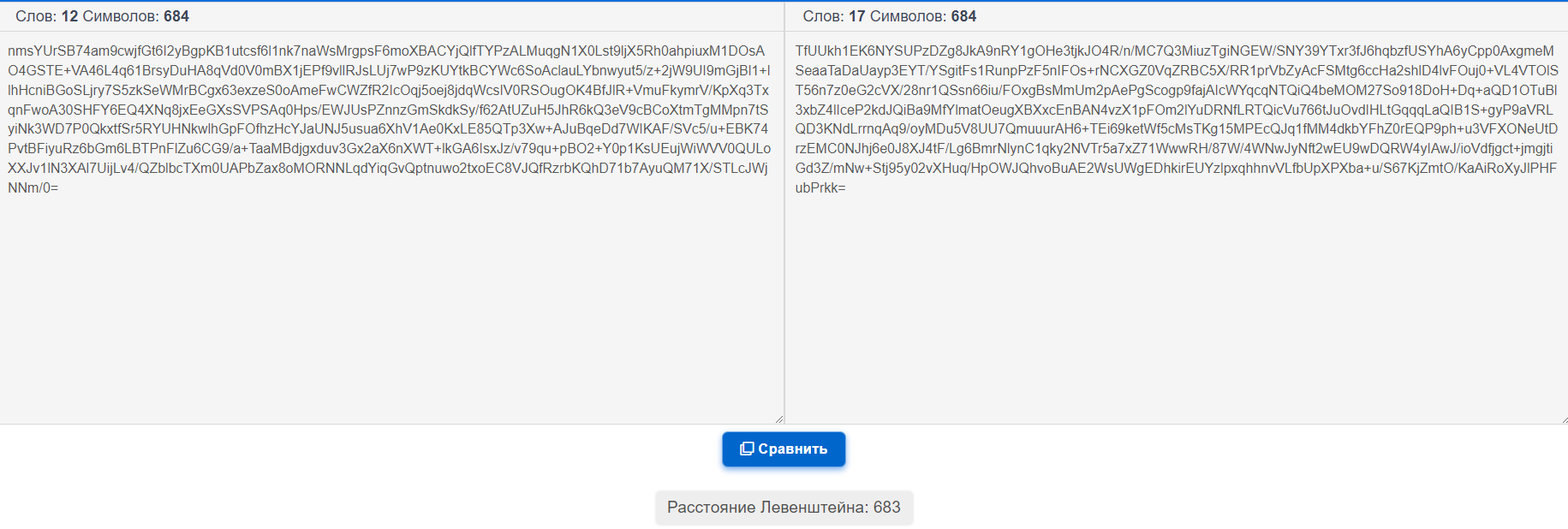


Рисунок 3.5 – Сравнение шифрованных текстов при проверке «лавинного эффекта»

«Лавинный эффекта» — это свойство алгоритма шифрования, при котором даже незначительное изменение в исходных данных приводит к значительному изменению в зашифрованном тексте. При проверке его работы мы изменяем key1.txt: «чёрт» => «чёрк», — и видим значительное изменение в шифрованном тексте. Расстояние Левенштейна 683/684, что указывает на критическое различие в результате.

## **3.4 Проверка расшифровки с изменённым ключом**

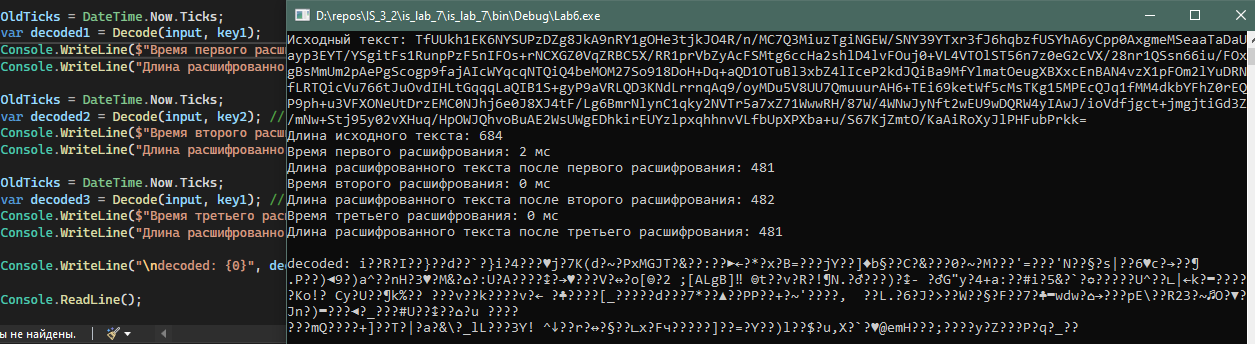


Рисунок 3.6 – Вывод расшифровки при смене ключа

При проверке его работы мы возвращаем исходное значение key1.txt: «чёрк» => «чёрт», — и используем результат работы при «чёрк», как входные данные. Из-за изменения мы получаем некорректную дешифрацию.

# **4 Вывод**

В ходе лабораторной работы были приобретены практические навыки разработки и использования приложений для реализации блочных шифров. Также, был выполнен анализ криптостойкости блочных шифров, оценена скорость зашифрования/расшифрования и сделаны соответствующие выводы.