Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Исследование блочных шифров

Студент: Авсюкевич П. В.

ФИТ 3 курс 5 группа

Преподаватель: Савельева М. Г.

Минск 2024

Задание 1: разработать приложение, которое реализует 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. По алгоритму DES для первых 8 символов имени и фамилии.

Реализация алгоритма представлена на листинге1.1.

|  |
| --- |
| ICryptoTransform encryptor = tdes.CreateEncryptor();  stopwatch.Start();  byte[] encData = encryptor.TransformFinalBlock(data, 0, data.Length);  stopwatch.Stop();  TimeSpan encryptionTime = stopwatch.Elapsed;  Console.WriteLine("Зашифрованные данные(Base64): " + Convert.ToBase64String(encData));  Console.WriteLine("Время шифрования: " + encryptionTime + " мс\n"); |

Листинг 1.1 – Код зашифрования алгоритмом DES

Код для расшифрования предоставлен на листинге 1.2.

|  |
| --- |
| ICryptoTransform decryptor = tdes.CreateDecryptor();  stopwatch.Reset();  stopwatch.Start();  byte[] originalData = decryptor.TransformFinalBlock(encData, 0, encData.Length);  stopwatch.Stop();  TimeSpan decryptionTime = stopwatch.Elapsed;  Console.WriteLine("Используется другой ключ для расшифровки: " + "00FF00FF00FF010FF");  Console.WriteLine("Дешифрованные данные: " + ASCIIEncoding.ASCII.GetString(originalData));  Console.WriteLine("Время дешифрования: " + decryptionTime + " мс\n"); |

Листинг 1.2 – Код расшифрования

Для оценки лавинного эффекта был разработан код, представленный в листинге 1.3. Принцип оценки заключается в сравнении каждого бита исходных данных и зашифрованных, при отличии увеличивает счетчик. Для демонстрации лавинного эффекта изменим первую букву в ключе результат на рисунках 1.1, 1.2. Результат изменения все последовательности, следовательно и лавинного эффекта.

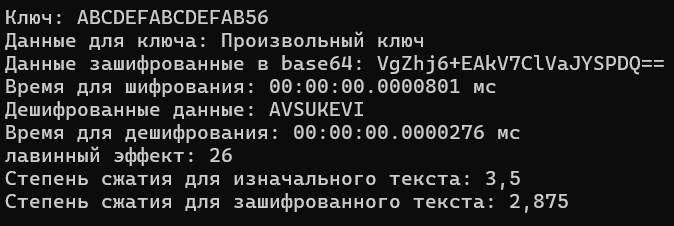


Рисунок 1.1 – Результат первым ключом

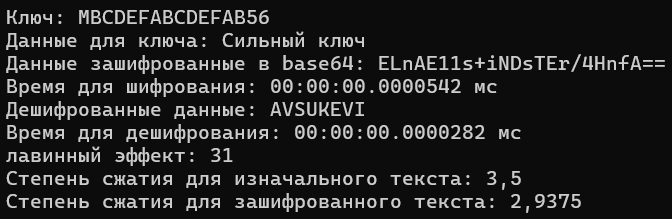


Рисунок 1.2 – Результат с изменённым первым символом ключа

|  |
| --- |
| static int AmountBits(byte b)  {  int count = 0;  for (int i = 0; i < 8; i++)  {  if ((b & (1 << i)) != 0)  {  count++;  }  }  return count;  } |

Листинг 1.3 – Функция для оценки лавинного эффекта

Результат выполнения программы предоставлен на рисунке 1.1.

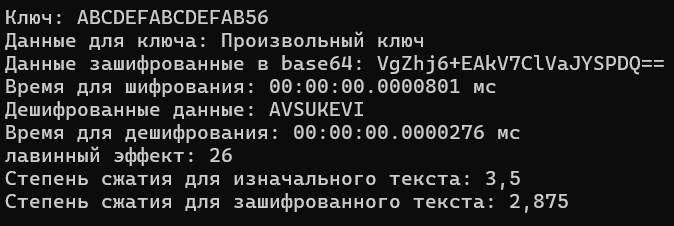


Рисунок 1.3 – Результат выполнения программы

Данный алгоритм позволяет выполнить зашифрование на основе слабых, полуслабых и сильных ключей . Для оценки разницы в лавинном эффекте приведем пример выполнения программы со всеми типами ключей, представлен на рисунке 1.4, 1.5, 1.6.

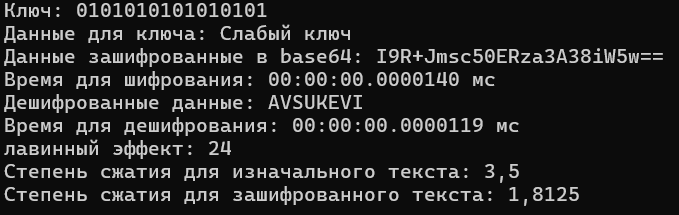


Рисунок 1.4 – Результат выполнения программы со слабым ключом

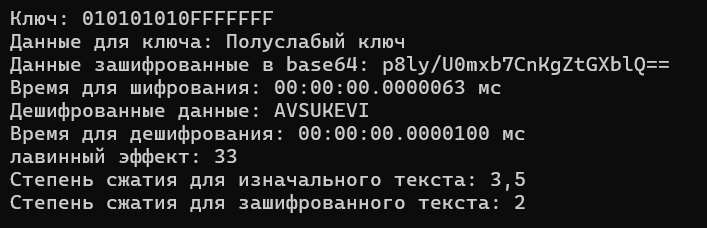


Рисунок 1.5 – Результат выполнения программы с полуслабым ключом

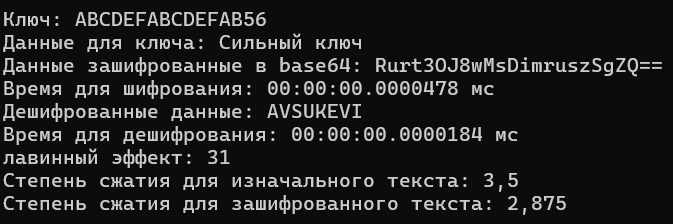


Рисунок 1.6 – Результат выполнения с сильным ключом

Для оценки времени выполнения операций зашифрования и расшифрования построим графики, которые представлены на рисунке

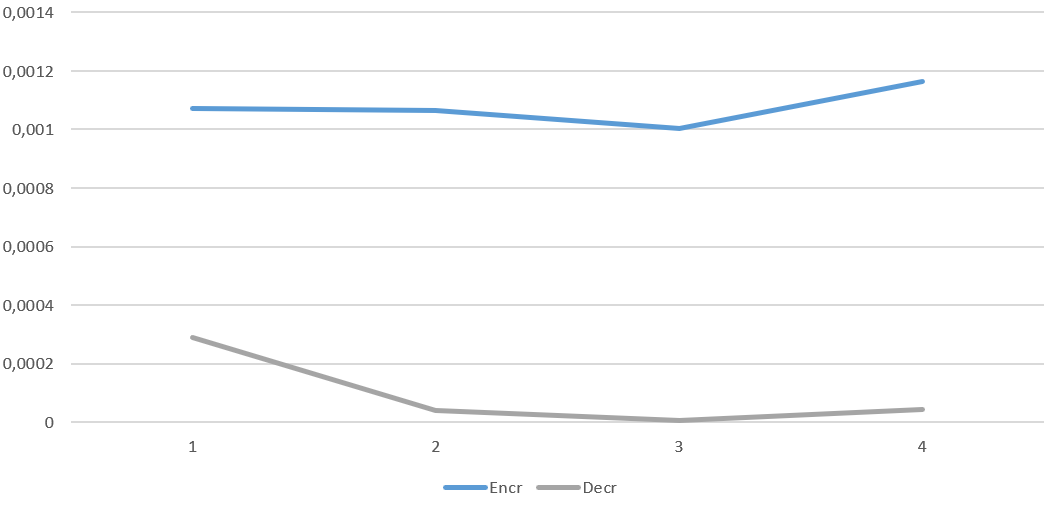


Рисунок 1.7 – Время выполнения расшифрования и зашифрования

Из-за того, что в DES создается псевдослучайная последовательность битов она не имеет определенной структуры, следовательно это делает ее несжимаемой. Размер исходного: 1 байт. Результат сжатия для надежного ключа: 0.8 байта, для слабого ключа: 0.9, для полуслабого ключа: 0.82, для исходного: 0.34 байт.

В ходе выполнения этой лабораторной работы мы не только изучили основы блочного шифрования DES, но и приобрели практические навыки по его реализации в приложениях. Мы освоили ключевые концепции алгоритма DES, такие как перестановки, замены и циклические сдвиги, и применили их при разработке наших собственных реализаций.