Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Исследование криптографических шифров на основе подстановки (замены) символов**

Студент: Сивак М.Н.

ФИТ 3 курс 4 группа

Преподаватель: Блинова

Минск 2021

1. **Описание приложения**

Приложение написано на объектно-ориентированном языке программирования C# и предназначено для шифрования и дешифрования текстовых документов на основе английского языка, используя:

* шифр аффинной системы подстановок Цезаря – security;
* таблица Трисемуса с ключевым словом – security;

Программное средство генерирует данные для построения гистограмм частот появления символов для исходного и зашифрованного сообщений, а также оценивает время выполнения операций шифрования и дешифрования.

1. **Методика выполнения поставленных задач**

При шифровании исходного текста шифром аффинного преобразования Цезаря применяются операции сложения и умножения по модулю n над элементами множества (индексами букв алфавита).

Математическая модель шифра выглядит следующим образом: y ≡ ax + b (mod N), – шифрование, x ≡ a^(-1) (y+N–b) (mod N), – дешифрование, где a и b – целые числа, где a^(-1) – обратное к a число по модулю N, т. е. оно удовлетворяет уравнению аa^(-1) ≡ 1 mod N. Часть реализации шифрования приведена на рисунке 2.1.

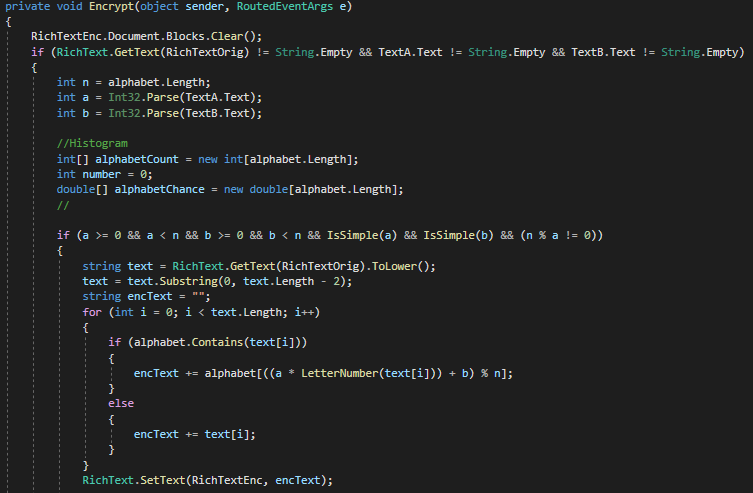


Рис. 2.1 – Реализация шифра Цезаря с ключевым словом

На рисунке приведен код, который работает с введёнными в окно ввода исходными данными, где message – это строка, хранящая считанные из файла

Дешифрование выполняется путем вычисления новой позиций в алфавите каждого символа после расшифровки по формуле (y-k) x ≡ a^(-1) (y+N–b) (mod N). Исходный текст, записанный в текстовое поле программы, ключи шифрования a и b, а также результат выполнения шифрования шифром аффинной подстановки Цезаря представлены на рисунке 2.2.

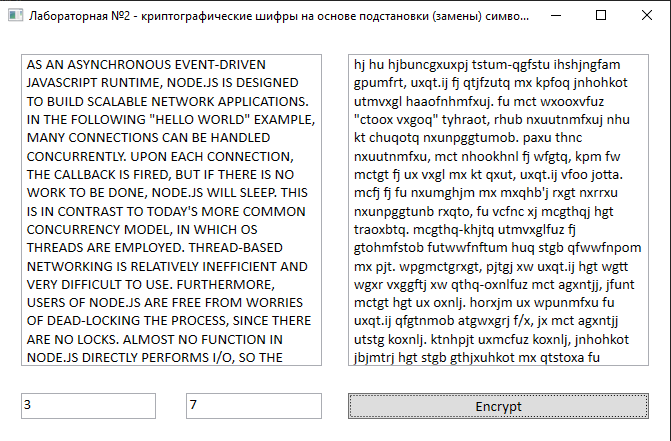


Рисунок 2.2 – Исходный текст и результат его шифрования

Как видно на рисунке 2.2, каждый символ исходного сообщения на основе выбранного алфавита заменяется на соответствующий символ, позиция которого вычислена по формуле y ≡ ax + b (mod N).

Пример дешифрования зашифрованного сообщения можно увидеть на рисунке 2.3.

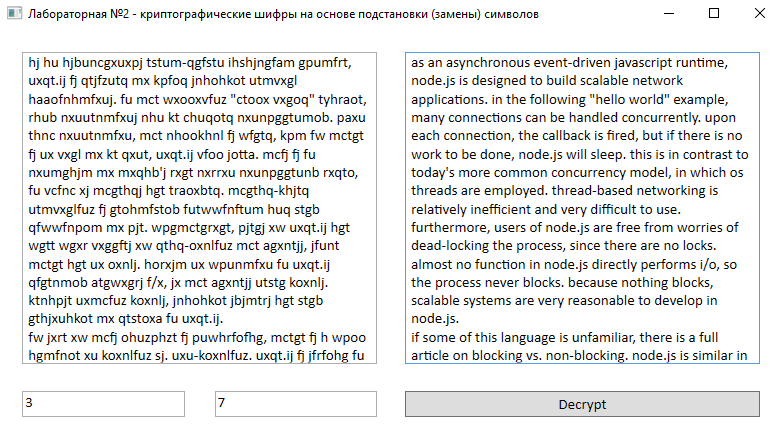


Рисунок 2.3 – Результат дешифрования зашифрованного текста

Также приложение генерирует гистограммы частот исходного и расшифрованного текстов.

Изображения частот исходного и расшифрованного текстов можно увидеть на рисунках 2.4 и 2.5 соответственно.

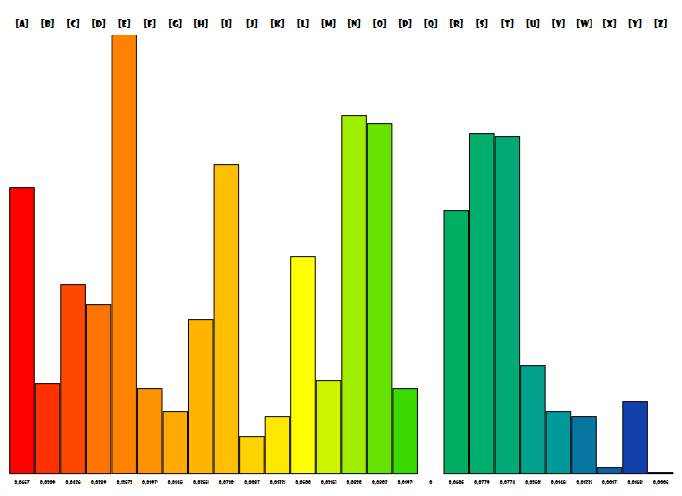


Рисунок 2.4 – Гистограмма частот исходного сообщения

Мы видим на гистограмме частот исходного текста, что самый часто встречаемый символ в исходном сообщении – это символ «E», остальные из самых встречаемых: «N», «O», «S», «T», «I».

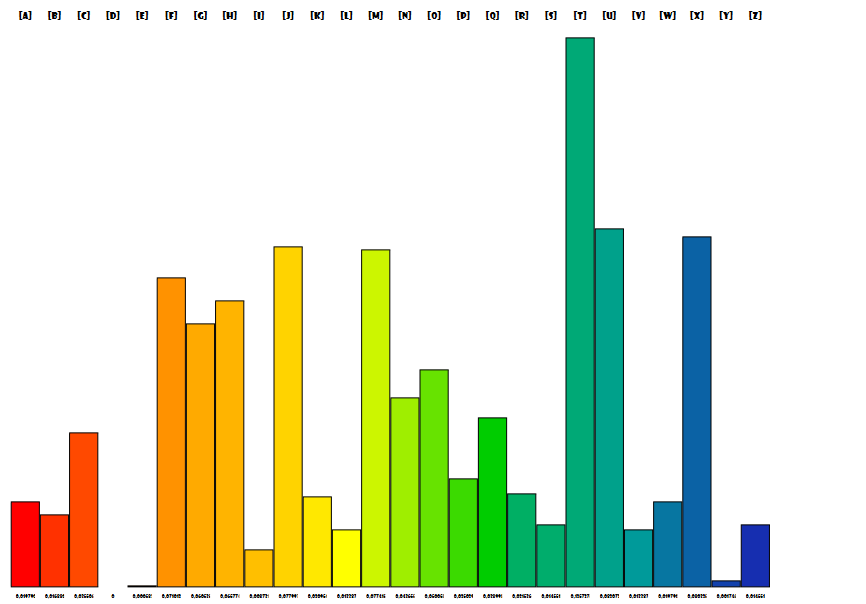


Рисунок 2.5 – Гистограмма частот зашифрованного сообщения

На данной гистограмме мы видим, что самый частый символ в зашифрованном сообщении – это «T», остальные из самых встречаемых: «U», «X», «J», «M», «F».

Рассмотрим пример шифрования исходного текста с помощью так называемой таблицы Трисемуса с применением ключевого слова. Для начала нужно избавиться от повторяющихся символов в ключевом слове. Далее его нужно вписать в таблицу, размером m×n, где m – строки таблицы, а n – столбцы, сформированную на основании исходного алфавита. Потом следует последовательно, начиная с последней буквы ключевого слова, дописать остальные символы исходного алфавита, исключая буквы, имеющиеся в ключевом слове. Таким образом формируется новый алфавит на основе исходного и ключевого слова. Далее при шифровании происходит замена символа входного текста на стоящий в сформированной таблице на одну позицию ниже его. Если входной символ является самым нижним в таблице, то он заменяется на символ, стоящий в этом же столбце в самом верху.

Реализация непосредственно участка кода, выполняющего шифрования с помощью таблицы Трисемуса с ключевым словом отображена на рисунке 2.4.

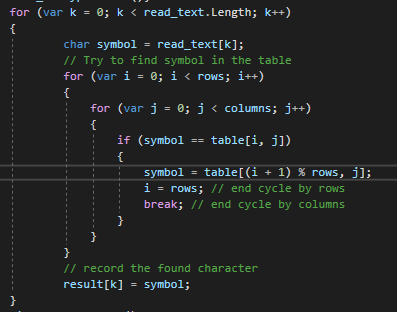


Рис. 2.4 – Реализация шифра таблицей Трисемуса с ключевым словом

Дешифрование производится в обратном порядке: символы зашифрованного сообщения заменяются на символы, стоящие выше их на одну позицию столбца, в котором они находятся.

Также приложение оценивает скорость выполнения шифрования/дешифрования с помощью встроенного класса языка C# – System.Diagnostics.Stopwatch.

Результаты вычисления скорости выполнения шифрования и дешифрования с помощью аффинного шифра Цезаря представлены на рисунке 2.5.



Рис. 2.5 – Скорость выполнения шифрования и дешифрования Цезаря

Результаты вычисления скорости выполнения шифрования и дешифрования с помощью таблицы Трисемуса представлены на рисунке 2.6.



Рисунок 2.6 – Скорость выполнения шифрования и дешифрование таблицей Трисемуса

Мы можем увидеть, что дешифрование зашифрованного текста занимает немного больше времени, чем шифрование, так как алгоритм при дешифровании является более сложным, следовательно машине требуется немного больше времени для выполнения данной операции.

На рисунке 2.7 можно увидеть результат выполнения шифрования таблицей Трисемуса с ключевым словом.



Рисунок 2.7 – Шифрование таблицей Трисемуса с ключевым словом

Мы видим, что все расшифрованные символы соответствуют каждому символу исходного сообщения. Значит, алгоритм дешифрования зашифрованного текста работает правильно.

Также приложение генерирует гистограммы частот появления символов в исходном и зашифрованном текстах, которые представлены на рисунке 2. И рисунке 2.9 соответственно.

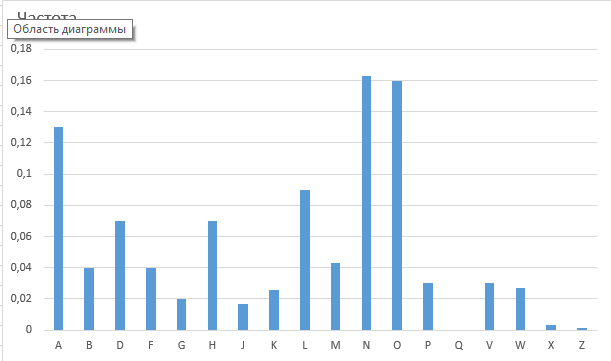


Рис. 2.8 – Гистограмма частот исходных символов

Данные гистограммы построены на основании полученных данных в ходе частотного анализа исходного текста и зашифрованного текста.

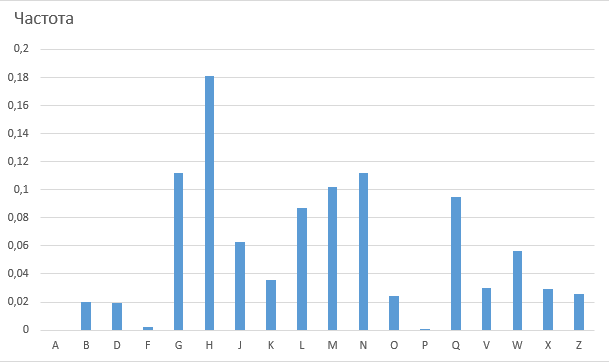


Рис. 2.9 – Гистограмма частота шифросимволов

**Вывод**

В ходе лабораторной работы были приобретены навыки написания алгоритмов для подстановочных шифров. Были реализованы алгоритмы аффинной подстановки Цезаря и таблицы Трисемуса с применением ключевого слова «security» для шифрования и дешифрования исходного текста. Также был проведён частотный анализ текста при шифровании и дешифровании и построены гистограммы, показывающие частоту появления каждого символа в исходном тексте.