Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

[Кафедра информационных](https://www.belstu.by/fakultety/fit/vm) систем и технологий

Специальность 1-40 01 01 Программное обеспечение информационных технологий

**Отчёт по лабораторной работе №5**

по дисциплине Информационная безопасность

Тема: Исследование криптографических шифров на основе перестановки символов

Студент: Станчик М.А.

ФИТ курс 3 группа 4

Преподаватель: Савельева М.Г.

Минск, 2025

**Лабораторная работа №5**

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации перестановочных шифров.

**Задачи:**

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости перестановочных шифров.

2. Ознакомиться с особенностями реализации и свойствами различных перестановочных шифров на основе готового программного средства (L\_LUX).

3. Разработать приложение для реализации указанных преподавателем методов перестановочного зашифрования/расшифрования.

4. Выполнить исследование криптостойкости шифров на основе статистических данных о частотах появления символов в исходном и зашифрованном сообщениях.

5. Оценить скорость зашифрования/расшифрования реализованных способов шифров.

6. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

# Теоретические сведения

Сущность перестановочного шифрования состоит в том, что исходный текст (*М*) и зашифрованный текст (*С*) основаны на использовании одного и того же алфавита, а тайной или ключевой информацией является алгоритм перестановки.

Шифры перестановки относятся к классу симметричных. Элементами текста могут быть отдельные символы (самый распространенный случай), пары, тройки букв и т. д.

Классическими примерами перестановочных шифров являются анаграммы.

В классической криптографии шифры перестановки делятся на два подкласса:

• шифры простой, или одинарной, перестановки – при зашифровании символы открытого текста *Мi* перемещаются с исходных позиций в новые (в шифртексте *Сi*) один раз;

• шифры сложной, или множественной, перестановки – при зашифровании символы открытого текста *Мi* перемещаются с исходных позиций в новые (в шифртексте *Сi*) несколько раз.

В общем случае для использования шифров одинарной перестановки используется таблица, состоящая из двух строк: в первой строке записываются буквы, во второй – цифры *J*. Строки состоят из n столбцов. Буквы составляют шифруемое сообщение. Цифры *J* = *j1*, *j2*, …, *jn*, где *j1* – номер позиции в зашифрованном сообщении первого символа открытого текста, *j2* – номер позиции в зашифрованном сообщении второго символа открытого текста и т. д. Таким образом, порядок следования цифр определяется используемым правилом (ключом) перестановки символов открытого текста для получения шифрограммы.

Основой современных шифров маршрутной перестановки является геометрическая фигура, обычно прямоугольник или прямоугольная матрица. В ячейки этой фигуры по определенному маршруту (слева направо, сверху вниз или каким-либо иным образом) записывается открытый текст. Для получения шифрограммы нужно записать символы этого сообщения в иной последовательности, т. е. по иному маршруту.

Шифр вертикальной перестановки является разновидностью шифра маршрутной перестановки. К особенностям вертикального шифра можно отнести следующие:

• количество столбцов в таблице фиксируется и определяется длиной ключа;

• маршрут вписывания: слева направо, сверху вниз;

• шифрограмма выписывается по столбцам в соответствии с их нумерацией (ключом).

Ключ может задаваться в виде текста (слова или словосочетания). Лексикографическое местоположение символов в ключевом выражении определяет порядок считывания столбцов.

Особенностью шифров множественной перестановки является минимум двукратная перестановка символов шифруемого сообщения. В простейшем случае это может задаваться перемешиванием не только столбцов, но и строк. Таким образом, этот случай соответствует использованию двух основных ключей: длина одного из них равна числу столбцов, другого – числу строк. К ключевой информацию мы можем относить также способы вписывания сообщения и считывания отдельных символов из текущего столбца матрицы.

Шифры гаммирования рассматриваются как самостоятельный класс. Такие шифры схожи с перестановочными тем, что в обоих случаях можно использовать табличное представление выполняемых операций на основе ключей. Вместе с тем шифры гаммирования имеют много общего с подстановочными шифрами, поскольку на самом деле при шифровании происходит подмена одних символов другими.

# Ход работы

Целью работы была разработка приложения, позволяющего выполнять шифрование/расшифрование текстовых документов, созданных на основе алфавита польського языка.

Для начала, был реализован алгоритм шифрования и расшифрования с помощью маршрутной перестановки. Маршрут: запись – по столбцам, считывание – по строкам таблицы.

Пример работы алгоритма:

1. Предположим, исходное сообщение – «Kocham programowanie».
2. Длина сообщения – 20 символов. Размер таблицы перестановки: 4×5.

Таблица (запись по столбцам):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K | a | r | a | a |
| o | m | o | m | n |
| c |  | g | o | i |
| h | p | r | w | e |

1. Далее для зашифрования сообщения таблица считывается по строкам. Зашифрованное сообщение: «Karaaomomnc goihprwe».
2. При расшифрования строится новая таблица, в которой столбцы и строки меняются местами. Т.е. если таблица была размеров 4×5, то новая таблица будет размеров 5×4. Исходное сообщение считывается по строкам.

Таблица (запись по столбцам):

|  |  |  |  |
| --- | --- | --- | --- |
| K | o | c | h |
| a | m |  | p |
| r | o | g | r |
| a | m | o | w |
| a | n | i | e |

Исходное сообщение: «Kocham programowanie».

Метод readFile принимает имя файла и его путь. Он выполняет чтение файла и запись его содержимого в строку. Код представлен в листинге 2.1.

public String readFile (String filePath) {

String content = "";

try {

content = new String(Files.readAllBytes(Paths.get(filePath)), "UTF-8");

}

catch (IOException e) {

e.printStackTrace();

}

return content;

}

Листинг 2.1 – Метод для записи содержимого текстового файла в строку

Метод шифрования encryptRouteSwap принимает как параметры количество строк в таблице для шифрования, количество столбцов в таблице для шифрования, исходный текст, который нужно зашифровать, потом организует текст в таблицу фиксированного размера, заполняет ее по столбцам и возвращает строку, представляющую зашифрованный текст. Код метода представлен в листинге 2.2.

public static String encryptRouteSwap(int rows, int cols, String text) throws IOException {

if (rows < 0) {

rows = rows \* (-1);

}

if (cols < 0) {

cols =cols \* (-1);

}

int totalChars = rows \* cols;

char[][] table = new char[rows][cols];

if (text.length() < totalChars) {

text = String.format("%-" + totalChars + "s", text).replace(' ', 'X');

}

if (text.length() > totalChars) {

System.out.println("Кол-во символов в тексте больше, чем может вместиться в таблицу, поэтому текст будет обрезан");

text = text.substring(0, totalChars);

}

int index = 0;

for (int col = 0; col < cols; col++) {

for (int row = 0; row < rows; row++) {

table[row][col] = text.charAt(index++);

}

}

StringBuilder result = new StringBuilder();

for (int row = 0; row < rows; row++) {

for (int col = 0; col < cols; col++) {

result.append(table[row][col]);

}

}

return result.toString();

}

Листинг 2.2 – Код метода шифрования с помощью маршрутной перестановки

Метод расшифрования decryptRouteSwap принимает как параметры количество строк в таблице, использованной для шифрования, количество столбцов в таблице, использованной для шифрования, зашифрованный текст, который нужно расшифровать. Метод формирует таблицу из зашифрованного текста, заполняя её по строкам, и затем считывает текст по столбцам, возвращая расшифрованный результат. Метод также обрабатывает случаи, когда зашифрованный текст не соответствует ожидаемой длине. Код метода представлен в листинге 2.3.

public String decryptRouteSwap(int rows, int cols, String encryptedText) throws IOException {

if (rows < 0) {

rows = rows \* (-1);

}

if (cols < 0) {

cols = cols \* (-1);

}

int totalChars = rows \* cols;

char[][] table = new char[rows][cols];

if (encryptedText.length() != totalChars) {

throw new IOException("Некорректная длина зашифрованного текста. Ожидается: " + totalChars);

}

int index = 0;

for (int row = 0; row < rows; row++) {

for (int col = 0; col < cols; col++) {

table[row][col] = encryptedText.charAt(index++);

}

}

StringBuilder result = new StringBuilder();

for (int col = 0; col < cols; col++) {

for (int row = 0; row < rows; row++) {

result.append(table[row][col]);

}

}

String decrypted = result.toString();

if (decrypted.endsWith("X")) {

decrypted = decrypted.replaceAll("X+$", "");

}

return decrypted;

}

Листинг 2.3 – Код метода расшифрования шифра маршрутной перестановки

Результат работы методов приведен на рисунке 2.1.

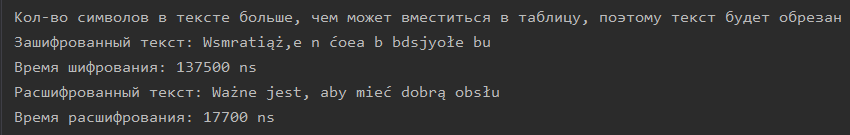


Рисунок 2.1 – Результат шифрования и дешифрования текста с помощью маршрутной перестановки

Метод calculateProbabilities принимает строку, которая содержит символы. Предназначен для вычисления вероятностей появления каждого уникального символа в заданной строке. Метод возвращает отображение, где ключами являются уникальные символы из строки, а значениями - соответствующие вероятности этих символов. Вероятность символа вычисляется как отношение количества его вхождений к общему количеству символов в строке. Код метода представлен в листинге 2.4.

public Map<Character, Double> calculateProbabilities(String text) {

Map<Character, Integer> frequencyMap = new HashMap<>();

int totalCount = 0;

for (char c : text.toCharArray()) {

frequencyMap.put(c, frequencyMap.getOrDefault(c, 0) + 1);

totalCount++;

}

Map<Character, Double> probabilities = new HashMap<>();

for (Map.Entry<Character, Integer> entry : frequencyMap.entrySet()) {

char character = entry.getKey();

int frequency = entry.getValue();

double probability = (double) frequency / totalCount;

probabilities.put(character, probability);

}

return probabilities;

}

Листинг 2.4 – Метод для вычисления вероятностей появления каждого уникального символа в строке

Метод calculateEntropy нужен для вычисления энтропии текста. Метод принимает отображение, где ключами являются уникальные символы из строки, а значениями - соответствующие вероятности этих символов. Этотпараметр должен быть получен с помощью метода calculateProbabilities. Код метода представлен в листинге 2.5.

public double calculateEntropy(Map<Character, Double> charProbabilities) {

double entropy = 0.0;

for (double charProbability : charProbabilities.values()) {

if (charProbability > 0) {

entropy -= charProbability \* (Math.log(charProbability) / Math.log(2));

}

}

return entropy;

}

Листинг 2.5 – Метод для вычисления вероятностей появления каждого уникального символа в строке

Метод saveFrequenciesToExcel предназначен для сохранения частот символов в Excel файл. Принимает отображение, где ключами являются уникальные символы, а значениями - соответствующие частоты этих символов, и имя файла (включая путь), в который будет сохранён Excel файл с частотами символов. Код метода представлен в листинге 2.6.

public void saveFrequenciesToExcel(Map<Character, Double> frequencies, String filename) {

Workbook workbook = new XSSFWorkbook();

Sheet sheet = workbook.createSheet("Frequencies");

Row headerRow = sheet.createRow(0);

headerRow.createCell(0).setCellValue("Символ");

headerRow.createCell(1).setCellValue("Частота");

int rowNum = 1;

for (Map.Entry<Character, Double> entry : frequencies.entrySet()) {

Row row = sheet.createRow(rowNum++);

row.createCell(0).setCellValue(entry.getKey().toString());

row.createCell(1).setCellValue(entry.getValue());

}

try (FileOutputStream fileOut = new FileOutputStream(filename)) {

workbook.write(fileOut);

} catch (Exception e) {

e.printStackTrace();

}

System.out.println("Частота символов сохранена в " + filename);

}

Листинг 2.6 – Метод для сохранения частот символов в Excel файл

Метод createDataset предназначен для создания набора данных, который будет использоваться для построения гистограммы с помощью библиотеки JFreeChart. Принимает отображение, которое вычисляется с помощью метода calculateProbabilities. Метод возвращает заполненный объект DefaultCategoryDataset. Код метода представлен в листинге 2.7.

private CategoryDataset createDataset(Map<Character, Double> frequencies) {

DefaultCategoryDataset dataset = new DefaultCategoryDataset();

for (Map.Entry<Character, Double> entry : frequencies.entrySet()) {

dataset.addValue(entry.getValue(), "Частота", entry.getKey().toString());

}

return dataset;

}

Листинг 2.7 – Метод для создания набора данных

Метод saveHistogramToExcel предназначен для создания и сохранения гистограммы частот символов в Excel файл. Метод принимает отображение и путь к файлу Excel, ничего не возвращает. Код метода представлен в листинге 2.8.

public void saveHistogramToExcel(Map<Character, Double> frequencies, String excelFilePath) {

CategoryDataset dataset = createDataset(frequencies);

JFreeChart histogram = ChartFactory.createBarChart(

"Частоты символов",

"Символ",

"Частота",

dataset

);

String imagePath = "histogram.png";

try {

ChartUtils.saveChartAsPNG(new File(imagePath), histogram, 800, 600);

System.out.println("Гистограмма сохранена в " + imagePath);

} catch (IOException e) {

e.printStackTrace();

}

insertImageToExcel(imagePath, excelFilePath);

}

Листинг 2.8 – Метод для создания и сохранения гистограммы частот символов в Excel файл

Далее, нужно было сформировать гистограммы частот появления символов для исходного и зашифрованного сообщений. Гистограмма частот появления символов исходного сообщения приведена на рисунке 2.2.

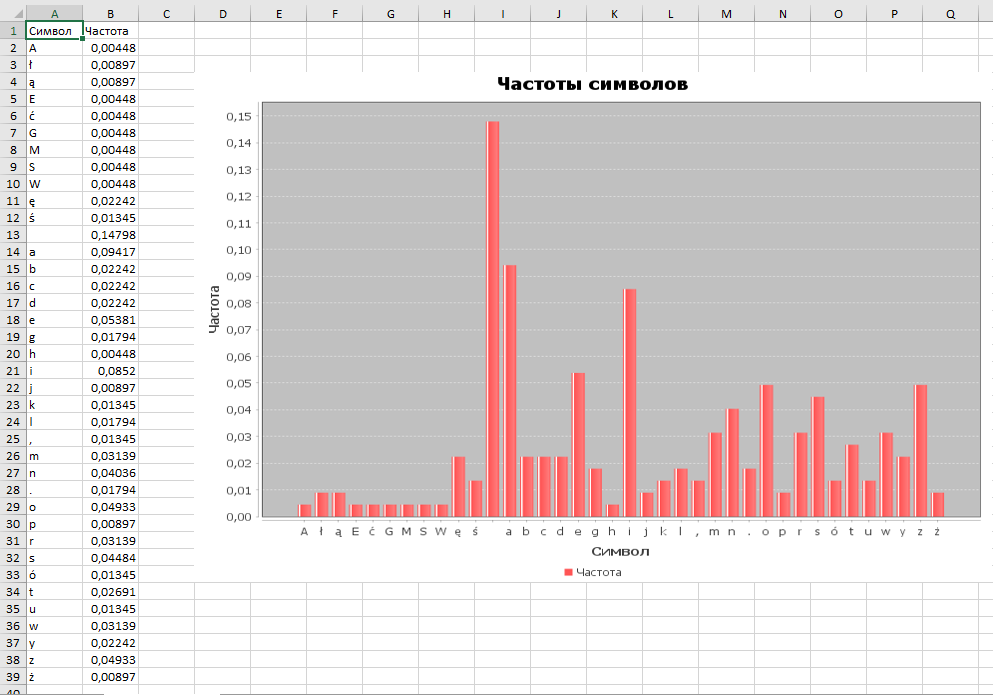


Рисунок 2.2 – Гистограмма частот появления символов исходного сообщения

Гистограмма частот появления символов зашифрованного сообщения приведена на рисунке 2.3.

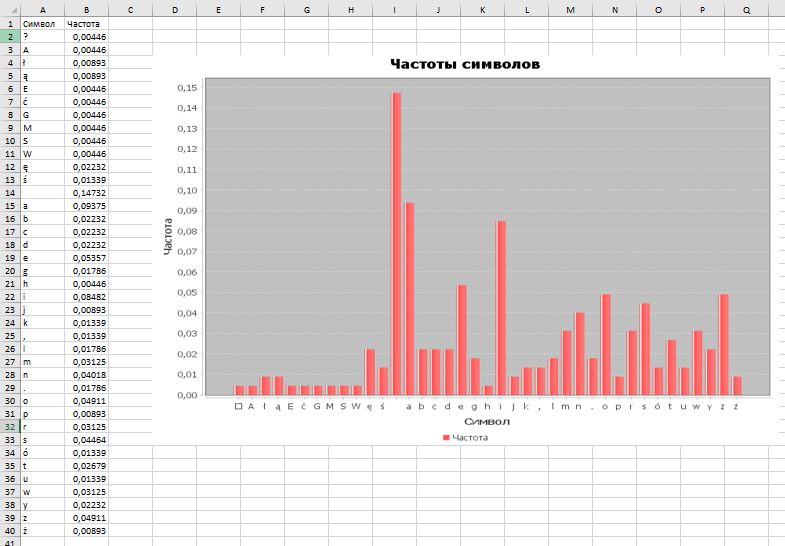


Рисунок 2.3 – Гистограмма частот появления символов зашифрованного сообщения

Для зашифрования необходима следующая информация: два ключевых слова, а также количество строк и столбцов в таблице, в которую будет вписано исходное сообщение, которое также должно быть больше или равно длине исходного сообщения. В случае, когда ключевые слова слишком короткие, они циклически повторяются. Далее записываются индексы строк и столбцов в соответствии с порядком символов в алфавите, после выполняется перестановка строк и далее столбцов.

Код метода, который выполняет зашифрование шифром множественной перестановки представлен в листинге 2.9.

public String encodeMultiple(String text, String columnKey, String rowKey) {

int columnLength = columnKey.length();

int rowLength = rowKey.length();

int index = 0;

int size = (int) Math.ceil((double) text.length() / (rowLength \* columnLength));

char[][] matrix = new char[rowLength][columnLength];

StringBuilder encryptedText = new StringBuilder();

for (int k = 0; k < size; k++) {

for (int i = 0; i < rowLength; i++) {

for (int j = 0; j < columnLength; j++) {

if (index < text.length()) {

matrix[i][j] = text.charAt(index++);

}

else {

matrix[i][j] = '\0';

}

}}

char[][] buff1 = new char[rowLength][columnLength];

int pr1 = 0;

HashSet<Integer> processedIndexes1 = new HashSet<>();

for (char column : columnKey.toCharArray()) {

for (int columnIndex = 0; columnIndex < columnKey.length(); columnIndex++) {

if (column == columnKey.charAt(columnIndex) && !processedIndexes1.contains(columnIndex)) {

processedIndexes1.add(columnIndex);

for (int i = 0; i < rowLength; i++) {

buff1[i][pr1] = matrix[i][columnIndex];

}

pr1++;}}}

char[][] buff2 = new char[rowLength][columnLength];

int pr2 = 0;

HashSet<Integer> processedIndexes2 = new HashSet<>();

for (char row : rowKey.toCharArray()) {

for (int rowIndex = 0; rowIndex < rowKey.length(); rowIndex++) {

if (row == rowKey.charAt(rowIndex) && !processedIndexes2.contains(rowIndex)) {

processedIndexes2.add(rowIndex);

for (int i = 0; i < columnLength; i++) {

buff2[pr2][i] = buff1[rowIndex][i];

}

pr2++;

}

}

}

for (int i = 0; i < columnLength; i++) {

for (int j = 0; j < rowLength; j++) {

encryptedText.append(buff2[j][i]);

}

}}

return encryptedText.toString();}

Листинг 2.9 – Метод для зашифрования шифром множественной подстановки

Для расшифрования текста, зашифрованного шифром множественной перестановки, необходимо выполнить те же действия в обратном порядке. Метод, реализующий расшифрование текста, представлен в листинге 2.10.

public String decodeMultiple(String text, String columnKey, String rowKey) {

int columnLength = columnKey.length();

int rowLength = rowKey.length();

int index = 0;

int size = (int) Math.ceil((double) text.length() / (rowLength \* columnLength));

char[][] matrix = new char[rowLength][columnLength];

StringBuilder decodeText = new StringBuilder();

for (int k = 0; k < size; k++) {

for (int i = 0; i < rowLength; i++) {

for (int j = 0; j < columnLength; j++) {

if (index < text.length()) {

matrix[i][j] = text.charAt(index++);

} else {

matrix[i][j] = '\0';

}

}

}

char[][] buff1 = new char[rowLength][columnLength];

int pr1 = 0;

HashSet<Integer> processedIndexes1 = new HashSet<>();

for (char row : rowKey.toCharArray()) {

for (int rowIndex = 0; rowIndex < rowKey.length(); rowIndex++) {

if (row == rowKey.charAt(rowIndex) && !processedIndexes1.contains(rowIndex)) {

processedIndexes1.add(rowIndex);

for (int i = 0; i < columnLength; i++) {

buff1[rowIndex][i] = matrix[pr1][i];

}

pr1++;

}

}

}

char[][] buff2 = new char[rowLength][columnLength];

int pr2 = 0;

HashSet<Integer> processedIndexes2 = new HashSet<>();

for (char column : columnKey.toCharArray()) {

for (int columnIndex = 0; columnIndex < columnKey.length(); columnIndex++) {

if (column == columnKey.charAt(columnIndex) && !processedIndexes2.contains(columnIndex)) {

processedIndexes2.add(columnIndex);

for (int i = 0; i < rowLength; i++) {

buff2[i][columnIndex] = buff1[i][pr2];

}

pr2++;

}

}

}

for (int i = 0; i < columnLength; i++) {

for (int j = 0; j < rowLength; j++) {

decodeText.append(buff2[j][i]);

}

}

}

return decodeText.toString().replaceAll("\0", "");

}

Листинг 2.10 – Метод расшифрования с помощью шифра множественной перестановки

Результат работы методов приведен на рисунке 2.4.

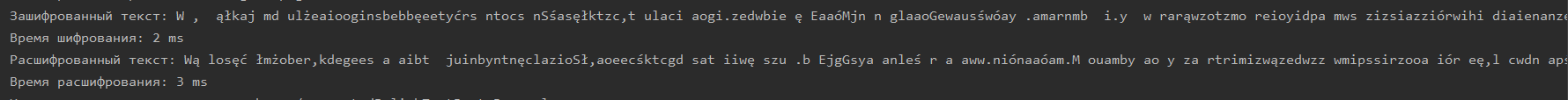


Рисунок 2.4 – Результат шифрования и дешифрования текста методом множественной перестановки

Гистограмма частот появления символов исходного сообщения приведена на рисунке 2.5.

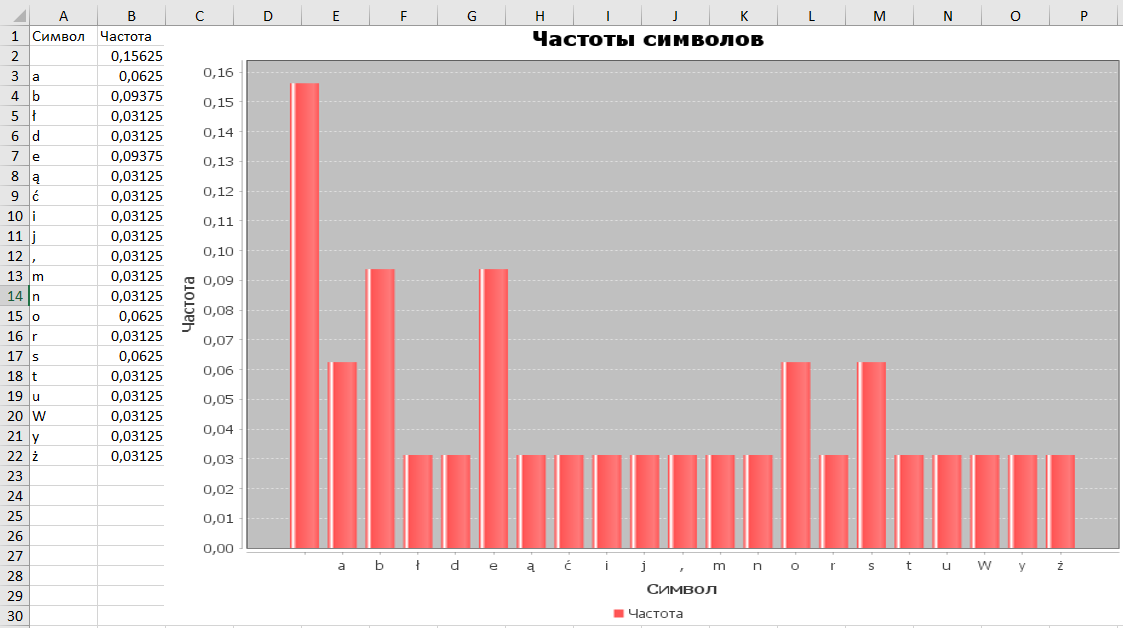


Рисунок 2.5 – Гистограмма частот появления символов исходного сообщения

Гистограмма частот появления символов зашифрованного сообщения приведена на рисунке 2.6.

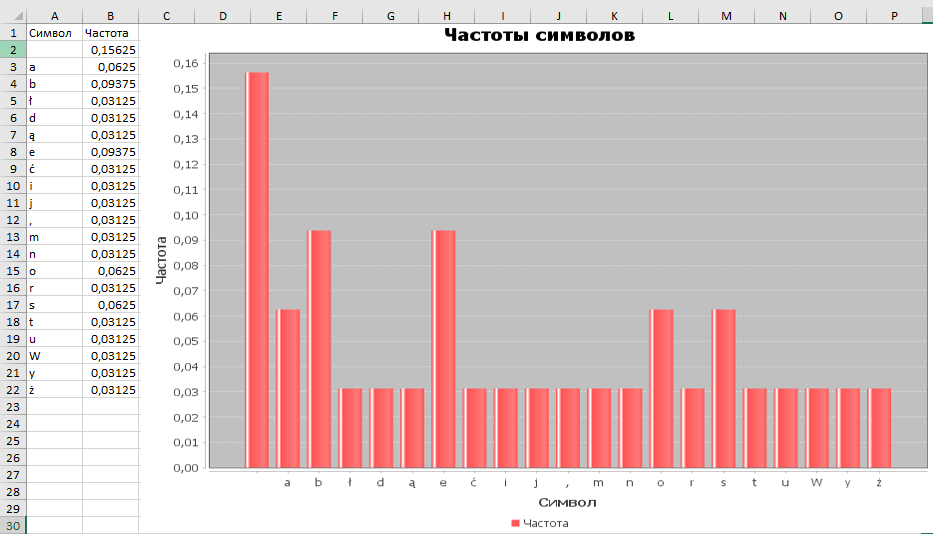


Рисунок 2.6 – Гистограмма частот появления символов зашифрованного сообщения

Для оценки времени, затраченного на выполнение операций зашифрования и расшифрования, используются методы System.nanoTime() и System.currentTimeMillis(). Вывод методов оценки времени представлен на рисунке 2.7.

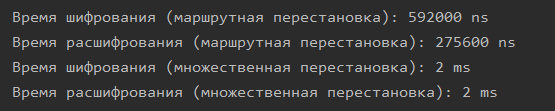


Рисунок 2.7 – Результат оценки времени выполнения

Как видно по результатам, зашифрование шифром множественной перестановки происходит медленнее, чем зашифрование маршрутным перестановочным шифром.

Графики выполнения программы трижды приведен на рисунке 4.2.



Рисунок 2.8 – Графики выполнения программы

Из графиков можно сделать вывод, что время выполнения может отличаться на несколько миллисекунд или долей миллисекунд.

Также было определено, зашифрование текста с помощью множественной перестановки выполняется примерно за 2 миллисекунды, а расшифрование за 1-2 миллисекунды, что немного уступает по сравнению с шифрованием и расшифрованием с помощью маршрутной перестановки.

**Вывод:** в ходе выполнения лабораторной работы были изучены основные принципы два перестановочных шифра – маршрутная перестановка и множественная перестановка, методы их зашифрования и расшифрования. С точки зрения криптоанализа, шифры не являются криптостойкими, так как уязвимы к частотному анализу. Также было разработано приложение для реализации методов маршрутной и множественной перестановок. Было выполнено исследование криптостойкости шифров на основе статистических данных о частотах появления символов в исходном и зашифрованном сообщениях. Была проведена оценка скорости зашифрования и расшифрования реализованных способов шифров. Это позволило сравнить эффективность различных алгоритмов. При сравнении времени выполнения операций зашифрования и расшифрования для обоих шифров выясняется, что зашифрование шифром множественной подстановки быстрее зашифрования маршрутной перестановкой примерно на 30%. Учитывая, что криптостойкость подстановочных и перестановочных шифров примерно одинаковая, ввиду большего быстродействия целесообразно использовать перестановочные шифры.