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 №4**

по дисциплине Информационная безопасность

Тема: Исследование криптографических шифров на основе подстановки (замены) символов

Студент: Станчик М.А.

ФИТ курс 3 группа 4

Преподаватель: Савельева М.Г.

Минск, 2025

**Лабораторная работа №4**

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

**Задачи:**

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

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

3. Разработать приложение для реализации указанных преподавателем методов подстановочного 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 подстановочного шифрования состоит в том, что исходный текст (из множества *М*) и зашифрованный текст (из множества *С*) основаны на использовании одного и того же или разных алфавитов, а тайной или ключевой информацией является алгоритм подстановки.

В моноалфавитных шифрах операция замены производится раздельно над каждым одиночным символом сообщения *Мi*.

Максимальное количество ключей для любого шифра этого вида не превышает *N*!, где *N* – количество символов в алфавите.

Для математического описания криптографического преобразования предполагаем, что зашифрованная буква *ay* (*ay* ∈ *Сi)*, соответствующая символу *aх* (*aх* ∈ *Мi*), находится на позиции

*y* ≡ *x* + *k* mod *N*,

где *x*, *y* – индекс (порядковый номер, начиная с 0) символа в используемом алфавите; *k* – ключ.

Для расшифрования сообщения *Сi* необходимо произвести расчеты, обратные выражению, т. е.

*х* ≡ *у* – *k* mod *N*.

Одним из существенных недостатков моноалфавитных шифров является их низкая криптостойкость. Зачастую метод криптоанализа базируется на частоте встречаемости букв исходного текста.

Система шифрования Цезаря с ключевым словом (лозунгом) также является одноалфавитной системой подстановки. Особенностью этой системы является использование ключевого слова (лозунга) для смещения и изменения порядка символов в алфавите подстановки (желательно, чтобы все буквы ключевого слова были различными). Ключевое слово пишется в начале алфавита подстановки.

Применяя одновременно операции сложения и умножения по модулю n над элементами множества (индексами букв алфавита), можно получить систему подстановок, которую называют аффинной системой подстановок Цезаря. Определим процедуру зашифрования в такой системе:

*y* ≡ *ax* + *b* mod *N*,

где *a* и *b* – целые числа. При этом взаимно однозначные соответствия между открытым текстом и шифртекстом будут иметь место только при выполнении следующих условий: 0 ≤ *a*, *b* < *N*, наибольший общий делитель (НОД) чисел *a*, *N* равен 1, т. е. эти числа являются взаимно простыми.

Полиалфавитные (или многоалфавитные) шифры состоят из нескольких шифров однозначной замены. Выбор варианта алфавита для зашифрования одного символа зависит от особенностей метода шифрования.

Таблица Трисемуса. Зашифрование осуществляется так: заготавливается таблица подстановки (так называемая «таблица Трисемуса» – таблица со стороной, равной *N*, где *N* – мощность алфавита), где первая строка – это алфавит, вторая – алфавит, сдвинутый на один символ, и т. д. При зашифровании первая буква открытого текста заменяется на букву, стоящую в первой строке, вторая – на букву, стоящую во второй строке, и т. д. После использования последней строки вновь возвращаются к первой. При шифровании с ключевым словом буква открытого текста заменяется буквой, расположенной ниже нее в том же столбце. Если буква текста оказывается в нижней строке таблицы, тогда для шифртекста берут самую верхнюю букву из того же столбца.

Шифр Виженера. В этом шифре мы имеем дело с последовательностью сдвигов, циклически повторяющейся. Основная идея заключается в следующем. Создается таблица (таблица Виженера) размером *N*×*N* (*N* – число знаков в используемом алфавите). Эти знаки могут включать не только буквы, но и, например, пробел или иные знаки. В первой строке таблицы записывается весь используемый алфавит. Каждая последующая строка получается из предыдущего циклического сдвига последней на 1 символ влево.

Криптоанализ – это раздел криптологии, занимающийся методами взлома шифров или методами организации криптографических атак на шифры.

Основные криптоатаки:

* Атака с известным шифртекстом;
* Атака с выбором шифртекста;
* Адаптивная атака с выбором шифртекста;
* Атака с известным открытым текстом;
* Атака с выбором открытого текста;
* Адаптивная атака с выбором открытого текста;
* Атака на основе связанных ключей;
* Атака с выбором ключа.

# Ход работы

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

Для начала, был реализован алгоритм шифрования и расшифрования с помощью шифра Цезаря. Метод шифрования caesarEncrypt принимает на вход исходный текст, ключ, алфавит подстановки и его мощность. Код метода представлен на листинге 2.1.

public static String caesarEncrypt(String message, int secretKey, String alphabet, int alphabetPower) {

StringBuilder encryptedMessage = new StringBuilder();

for (char character : message.toCharArray()) {

int index = alphabet.indexOf(character);

if (index != -1) {

int encryptedIndex = (index + secretKey) % alphabetPower;

encryptedMessage.append(alphabet.charAt(encryptedIndex));

}

else {

System.out.println("Символ " + character + " не найден в алфавите, добавили без изменений");

encryptedMessage.append(character);

}

}

return encryptedMessage.toString();}

Листинг 2.1 – Код метода шифрования с помощью шифра Цезаря

Метод расшифрования caesarDecrypt принимает на вход зашифрованный текст, ключ, алфавит подстановки и его мощность. Код метода представлен на листинге 2.2.

public static String caesarDecrypt(String encryptedMessage, int key, String alphabet, int alphabetPower) {

StringBuilder decryptedMessage = new StringBuilder();

for (char character : encryptedMessage.toCharArray()) {

int index = alphabet.indexOf(character);

if (index != -1) {

int decryptedIndex = (index - key + alphabetPower) % alphabetPower;

decryptedMessage.append(alphabet.charAt(decryptedIndex));

}

else {

System.out.println("Символ " + character + " не найден в алфавите, добавили без изменений");

decryptedMessage.append(character);

}

}

return decryptedMessage.toString();

}

Листинг 2.2 – Код метода расшифрования шифра Цезаря

Результат работы методов приведен на рисунке 2.1.

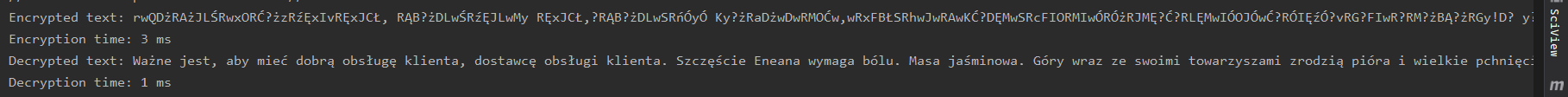


Рисунок 2.1 – Результат шифрования и дешифрования текста с помощью шифра Цезаря с ключевым словом «інфарматыка»

Метод readFile принимает имя файла и его путь. Он выполняет чтение файла и запись его содержимого в строку. Код представлен в листинге 2.4.

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 2.4 – Метод для записи содержимого текстового файла в строку

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 2.5.

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 2.5 – 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 2.6.

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 2.6 – 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 2.7.

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 2.7 – 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 2.8.

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 2.8 – 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 2.9.

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

Гистограмма частот появления символов исходного сообщения приведена на рисунке 2.2.

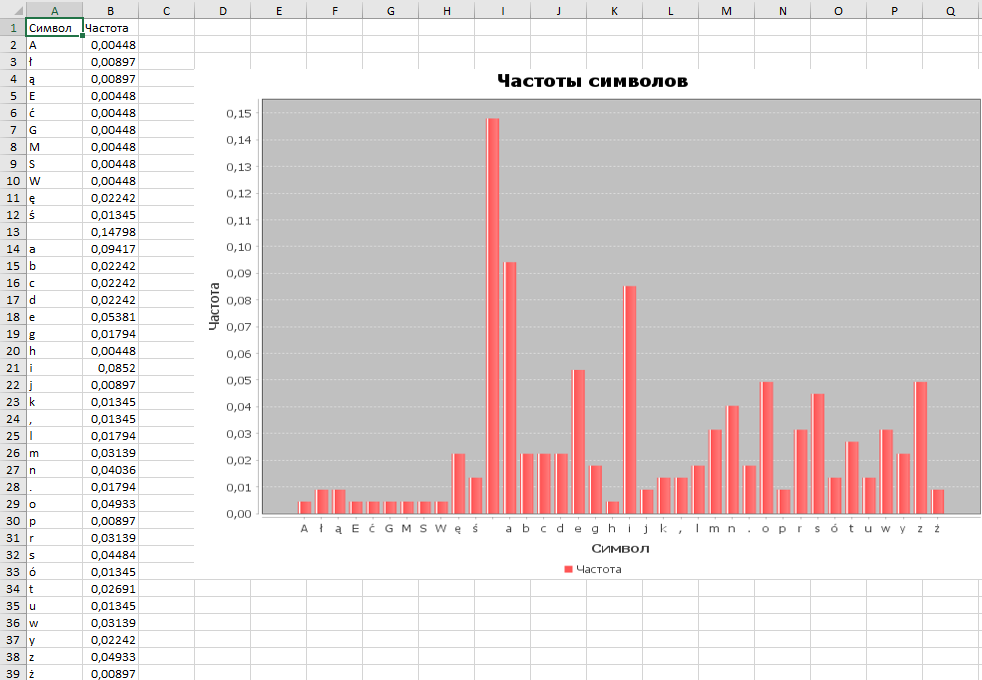


Рисунок 2.2 – Гистограмма частот появления символов исходного сообщения

Гистограмма частот появления символов зашифрованного сообщения приведена на рисунке 2.3.

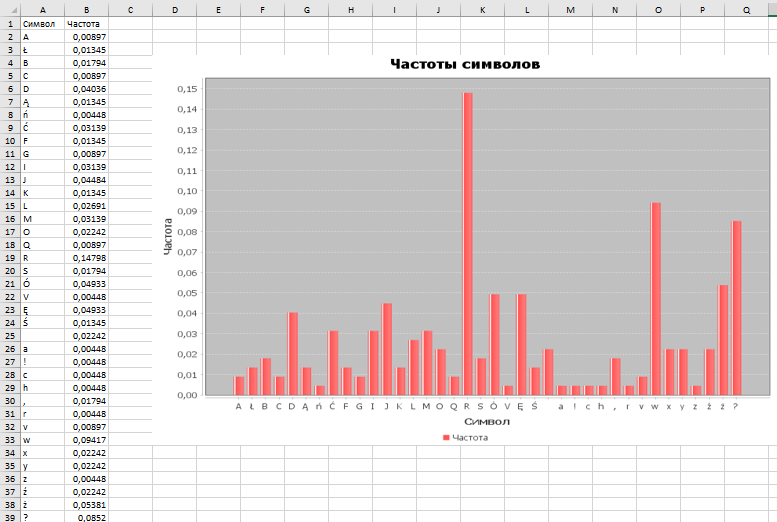


Рисунок 2.3 – Гистограмма частот появления символов зашифрованного сообщения

Сопоставив гистограммы частот появления символов исходного и зашифрованного сообщений можно определить алфавит подстановки.

Также было определено, что как зашифрование, так и расшифрование текста с помощью шифра Цезаря выполняется примерно за 1-3 миллисекунды, расшифрование проводится на 1-2 миллисекунды дольше.

Затем были разработаны несколько функций, выполняющих шифрование и дешифрование текста с использованием шифра Порты.

Сначала была разработан метод portCipherEncrypt, который реализует шифрование с помощью портативного шифра (или шифра замены), который использует матрицу символов (алфавит) для преобразования букв текста в числовые индексы. Метод возвращает зашифрованный текст в виде строки, убирая лишние пробелы в начале и конце. Код метода представлен в листинге 2.10.

public static String portCipherEncrypt(String text, String alphabet, int alphabetLength) {

if (text.length() % 2 != 0) {

text += "А";

}

StringBuilder encryptedText = new StringBuilder();

for (int i = 0; i < text.length(); i += 2) {

char firstChar = text.charAt(i);

char secondChar = text.charAt(i + 1);

int row = alphabet.indexOf(firstChar);

int col = alphabet.indexOf(secondChar);

if (row != -1 && col != -1) {

int encryptedIndex = row \* alphabetLength + col + 1;

encryptedText.append(String.format("%03d ", encryptedIndex));

}

}

return encryptedText.toString().trim();

}

Листинг 2.10 – Код метода portCipherEncrypt

Результат работы метода приведен на рисунке 2.4.



Рисунок 2.4 – Зашифрованый текст в числовом формате

Метод portCipherDecrypt предназначен для расшифровки текста, зашифрованного с использованием портативного шифра (или шифра замены), аналогичного методу portCipherEncrypt. Метод принимает как параметры строку, содержащую зашифрованный текст, представленный в виде числовых индексов, разделенных пробелами, строку, представляющую алфавит (набор символов), используемую для расшифровки, длину алфавита, которая используется для вычислений. Метод возвращает расшифрованный текст в виде строки. Код метода представлен в листинге 2.11.

public static String portCipherDecrypt(String encryptedText, String alphabet, int alphabetLength) {

StringBuilder decryptedText = new StringBuilder();

String[] indices = encryptedText.split(" ");

for (String indexStr : indices) {

int index = Integer.parseInt(indexStr) - 1;

int row = index / alphabetLength;

int col = index % alphabetLength; decryptedText.append(alphabet.charAt(row)).append(alphabet.charAt(col));

}

return decryptedText.toString();}

Листинг 2.11 – Код метода дешифрования с помощью метода portCipherDecrypt

Результат работы метода приведен на рисунке 2.5.



Рисунок 2.5 – Результат дешифрования текста с помощью шифрования Порта

Гистограмма частот появления символов исходного текста приведена на рисунке 2.6.

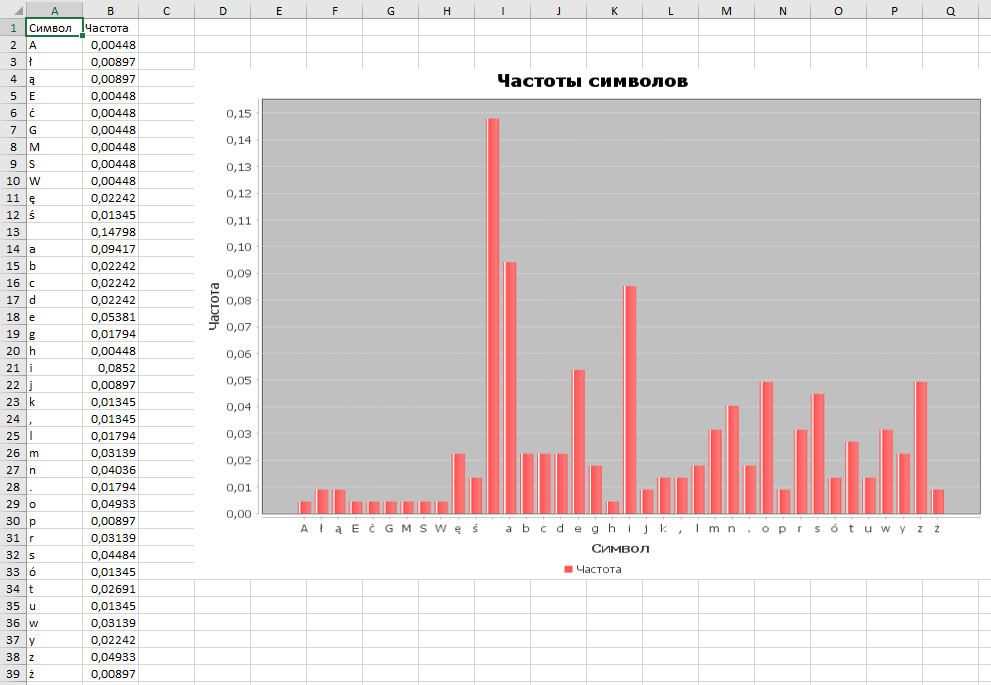


Рисунок 2.6 – Гистограмма частот появления символов исходного текста

Гистограмма частот появления символов зашифрованного текста приведена на рисунке 2.7.

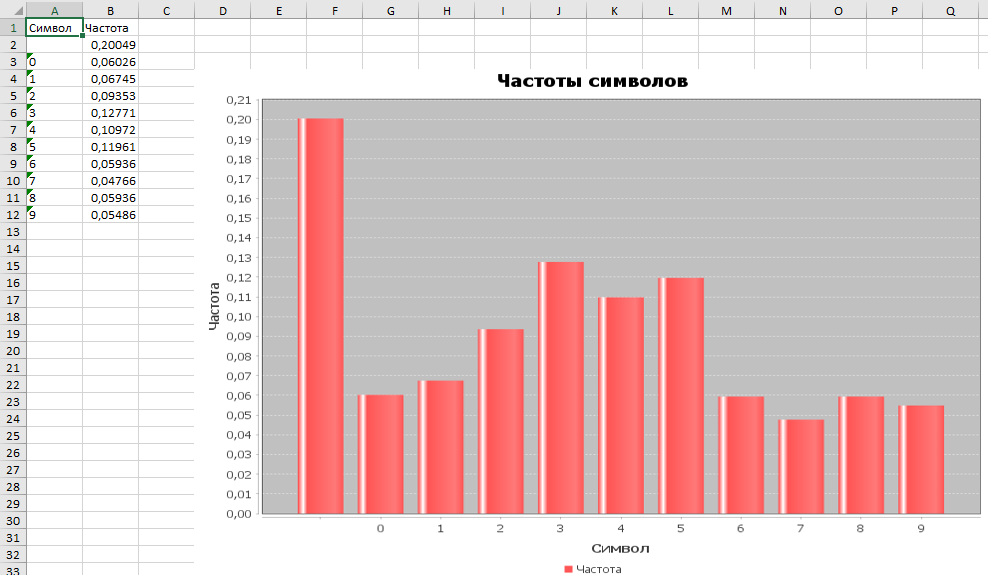


Рисунок 2.7 – Гистограмма частот появления символов зашифрованного текста

Для наглядной оценки времени, которое было затрачено на выполнение операций зашифрования и расшифрования, были сделаны графики. Программа запускалась трижды. Результат приведен на рисунке 2.8.



Рисунок 2.8 – Графики выполнения программы

На основе представленных данных о времени выполнения операций можно сделать вывод, что обе операции шифрования и расшифрования шифра Цезаря выполняются крайне быстро, так как алгоритм основан на простом сдвиге символов. Шифрование Порты значительно медленнее (в ~19 раз), чем у Цезаря, из-за необходимости поиска соответствий между двумя алфавитами, расшифрование же сопоставимо по скорости с шифром Цезаря.

Сопоставив гистограммы частот появления символов исходного и зашифрованного сообщений можно определить алфавит подстановки.

Также было определено, зашифрование текста с помощью шифрования Порта выполняется примерно за 35 миллисекунд, а расшифрование за 3, что значительно уступает по сравнению с шифрованием и расшифрованием с помощью шифра Цезаря.

**Вывод:** в ходе выполнения лабораторной работы были изучены основные принципы работы подстановочных шифров, методы их зашифрования и расшифрования. Было разработано приложение, позволяющее шифровать и расшифровывать тексты методами Цезаря и Порты, а также анализировать криптостойкость через частотный анализ символов. Было выполнено исследование криптостойкости шифров на основе статистических данных о частотах появления символов в исходном и зашифрованном сообщениях. Шифр Порты демонстрирует большую криптостойкость, чем шифр Цезаря, так как использование двух алфавитов усложняет частотный анализ, но требует больше времени для шифрования и расшифрования из-за необходимости поиска соответствий между алфавитами. При этом его расшифрование также медленнее из-за необходимости поиска соответствий между алфавитами.