МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Исследование криптографических шифров на основе подстановки (замены) символов

Студент: Евсеенко В. П.

ФИТ 3 курс 4 группа

Преподаватель: Савельева М. Г.

Минск 2025

Содержание

[Введение 3](#_Toc193146557)

[1 Практическое задание 4](#_Toc193146558)

[2 Гистограммы 10](#_Toc193146559)

[3 Оценка времени 11](#_Toc193146560)

[Вывод 12](#_Toc193146561)

# **Введение**

Цель лабораторной работы:

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

Задачи:

* закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости подстановочных шифров.
* ознакомиться с особенностями реализации и свойствами различных подстановочных шифров на основе готового программного средства (L\_LUX);
* разработать приложение для реализации указанных преподавателем методов подстановочного зашифрования/расшифрования;
* выполнить исследование криптостойкости шифров на основе статистических данных о частотах появления символов в исходном и зашифрованном сообщениях;
* оценить скорость зашифрования/расшифрования реализованных способов шифров;
* результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

# **1 Практическое задание**

Разработать авторское приложение в соответствии с целью лабораторной работы. Приложение должно реализовывать следующие операции:

* выполнять зашифрование/расшифрование текстовых документов (объемом не менее 5 тысяч знаков), созданных на основе алфавита языка в соответствии с нижеследующей таблицей вариантов задания; при этом следует использовать шифры подстановки из третьего столбца данной таблицы (варианты задания представлены в табл. 1.1).

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*, (1.1)

где *x*, *y* – индекс (порядковый номер, начиная с 0) символа в используемом алфавите; *k* – ключ. Для расшифрования сообщения *Сi* необходимо произвести рас четы, обратные выражению (1.1), т. е.

*х* ≡ *у* – *k* mod *N*, (1.2)

Соотношениям (1.1) и (1.2) соответствует классический шифр подстановки – шифр Цезаря.

Таблица 1.1 – Варианты задания

|  |  |  |
| --- | --- | --- |
| Вариант | Алфавит | Шифр |
| 1 | Белорусский | 1. На основе соотношений (1.1) и (1.2); *k* = 5 2 2. Виженера, ключевое слово – собственная фамилия |
| 2 | Русский | 1. На основе аффинной системы подстановок Цезаря; *a* = 7, *b* = 10 2. Виженера, ключевое слово – собственная фамилия |
| 3 | Английский | 1. Шифр Цезаря с ключевым словом, ключе вое слово – собственная фамилия 2. Таблица Трисемуса, ключевое слово – собственное имя |
| 4 | Немецкий | 1. На основе соотношений (1.1) и (1.2); *k* = 7 2. Таблица Трисемуса, ключевое слово – enigma |
| 5 | Польский | 1. На основе аффинной системы подстановок Цезаря; *a* = 5, *b* = 7 2. Шифр Порты |

Продолжение таблицы 1.1

|  |  |  |
| --- | --- | --- |
| 6 | Белорусский | 1. На основе соотношений (1.1) и (1.2); *k* = 21 2. Таблица Трисемуса, ключевое слово – собственное имя |
| 7 | Русский | 1. Шифр Порты 2. Шифр Цезаря с ключевым словом, ключе вое слово – собственная фамилия |
| 8 | Английский | 1. Шифр Цезаря с ключевым словом, ключе вое слово – собственная фамилия, *а* = 24 2. Таблица Трисемуса, ключевое слово – собственное имя |
| 9 | Немецкий | 1. На основе соотношений (1.1) и (1.2); *k* = 7 2. Таблица Трисемуса, ключевое слово – enigma |
| 10 | Польский | 1. На основе соотношений (1.1) и (1.2); *k* = 28 2. Шифр Порты |
| 11 | Белорусский | 1. Шифр Цезаря с ключевым словом, ключе вое слово – інфарматыка, *а* = 2 2. Таблица Трисемуса, ключевое слово – собственное имя |
| 12 | Русский | 1. Шифр Цезаря с ключевым словом, ключе вое слово – безопасность 2. Таблица Трисемуса, ключевое слово – безопасность |
| 13 | Английский | 1. На основе аффинной системы подстановок Цезаря; *a* = 6, *b* = 7 2. Таблица Трисемуса, ключевое слово – security |
| 14 | Немецкий | 1. Виженера, ключевое слово–собственная фамилия 2. Шифр Порты |
| 15 | Польский | 1. Виженера, ключевое слово bezpieczeństwo 2. На основе соотношений (1.1) и (1.2); *k* = 20 |

Выбранный вариант – 4. Далее выполнение практического задания будет производиться на основе немецкого алфавита. Данный шифр будет: на основе соотношений (1.1) и (1.2); *k* = 7, также Таблица Трисемуса, ключевое слово – enigma.

Код для шифрования и расшифрования текста по Цезарю на основе соотношений 1.1) и (1.2); где *k* = 7, приведен в листинге 1.1.

|  |
| --- |
| function caesarCipherEncrypt(text, k) {      let result = '';      text = text.toLowerCase();      for (let i = 0; i < text.length; i++) {          let char = text[i];          let index = germanAlphabet.indexOf(char);          if (index !== -1) {          let newIndex = (index + k) % germanAlphabet.length;              result += germanAlphabet[newIndex];          } else {              result += char;          }      }      return result;  }  function caesarCipherDecrypt(text, k) {      let result = '';      text = text.toLowerCase()      for (let i = 0; i < text.length; i++) {          let char = text[i];          let index = germanAlphabet.indexOf(char);          if (index !== -1) {          let newIndex = (index - k + germanAlphabet.length) % germanAlphabet.length;              result += germanAlphabet[newIndex];          } else {              result += char;          }      }      return result;  } |

Листинг 1.1 – Функции шифрования и расшифрования

Шифр Цезаря — это классический метод шифрования, который сдвигает каждую букву текста на фиксированное количество позиций в алфавите. В вашем случае для немецкого алфавита с *k* = 7 (сдвиг на 7 позиций), шифрование работает следующим образом: каждая буква текста заменяется на букву, находящуюся на 7 позиций дальше по алфавиту. Если символ выходит за пределы алфавита, сдвиг продолжается с начала, благодаря операции остатка от деления (mod).

Результат выполнения программы для шифрования и расшифрования текста приведен на рисунке 1.1.

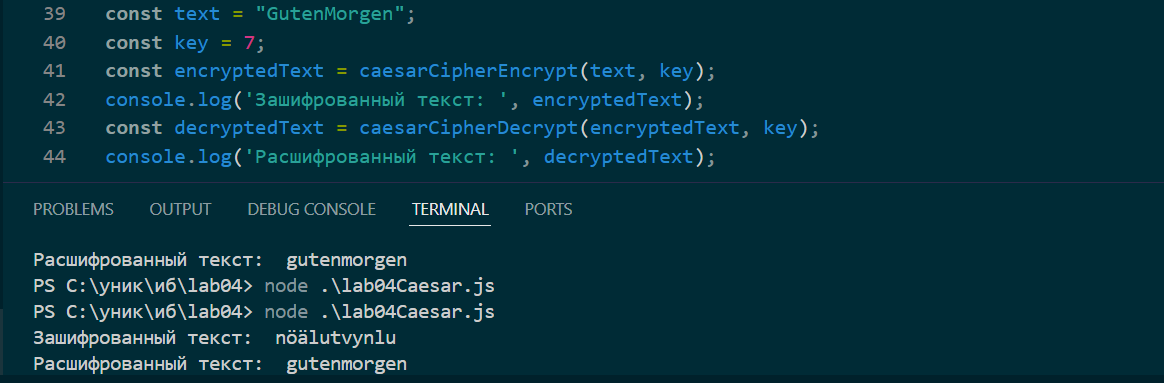


Рисунок 1.1 – Результат выполнения программы

Шифрование и расшифрование текста с помощью метода Трисемуса основано на использовании специальной таблицы, которая формируется из ключевого слова и оставшихся букв алфавита. Сначала строится таблица, в которую по порядку записываются буквы ключевого слова без повторений, а затем добавляются все остальные символы алфавита.

После создания таблицы текст преобразуется в пары символов, при этом пробелы заменяются на специальный символ, например «x», чтобы избежать разрывов. Далее, в зависимости от расположения букв в таблице, происходит замена: если символы находятся в одной строке, они заменяются на соседние справа, если в одном столбце – на символы ниже, а если образуют прямоугольник, то заменяются на противоположные углы этого прямоугольника. Расшифрование происходит аналогично, но со сдвигом в обратную сторону.

В конце расшифрованного текста удаляются вспомогательные символы, такие как «x», если они были добавлены. Этот метод обеспечивает простую, но эффективную подстановку символов, усложняя анализ шифра без знания ключевого слова. Код для создания таблицы Трисемуса для выполнения задания, приведен в листинге 1.2.

|  |
| --- |
| function createTrisemusTable(key) {      let table = '';      let usedChars = new Set();      for (let i = 0; i < key.length; i++) {      let char = key[i].toLowerCase();   if (!usedChars.has(char) && germanAlphabet.includes(char)) { table += char;              usedChars.add(char);}}      for (let i = 0; i < germanAlphabet.length; i++) {          let char = germanAlphabet[i];          if (!usedChars.has(char)) {              table += char;              usedChars.add(char);}}      return table;} |

Листинг 1.2 – Функция для создания таблицы Трисемуса

Функция необходимая для шифрования по методу Трисемуса приведена в листинге 1.3.

|  |
| --- |
| function trisemusEncrypt(text, key) {      let table = createTrisemusTable(key);      let result = '';      text = text.toLowerCase().replace(/\s+/g, 'x');  for (let i = 0; i < text.length; i += 2) {          let pair = text.substring(i, i + 2);          if (pair.length < 2) {              pair += 'x';  }          let firstChar = pair[0];          let secondChar = pair[1];          let firstPos = table.indexOf(firstChar);          let secondPos = table.indexOf(secondChar);          let firstRow = Math.floor(firstPos / 5);          let firstCol = firstPos % 5;          let secondRow = Math.floor(secondPos / 5);          let secondCol = secondPos % 5;          if (firstRow === secondRow) {        result += table[firstRow \* 5 + (firstCol + 1) % 5];        result += table[secondRow \* 5 + (secondCol + 1) % 5];          } else if (firstCol === secondCol) {       result += table[((firstRow + 1) % 5) \* 5 + firstCol];        result += table[((secondRow + 1) % 5) \* 5 + secondCol];          } else {              result += table[firstRow \* 5 + secondCol];              result += table[secondRow \* 5 + firstCol];}}      return result;} |

Листинг 1.3 – Функция шифрования по методу Трисемуса

В листинге же 1.4 приведена функция расшифрования.

|  |
| --- |
| function trisemusDecrypt(text, key) {      let table = createTrisemusTable(key);      let result = '';      text = text.toLowerCase();      for (let i = 0; i < text.length; i += 2) {          let pair = text.substring(i, i + 2);          let firstChar = pair[0];          let secondChar = pair[1];          let firstPos = table.indexOf(firstChar);          let secondPos = table.indexOf(secondChar);          let firstRow = Math.floor(firstPos / 5);          let firstCol = firstPos % 5;          let secondRow = Math.floor(secondPos / 5); |
| let secondCol = secondPos % 5;          if (firstRow === secondRow) {    result += table[firstRow \* 5 + (firstCol - 1 + 5) % 5];    result += table[secondRow \* 5 + (secondCol - 1 + 5) % 5];          } else if (firstCol === secondCol) {    result += table[((firstRow - 1 + 5) % 5) \* 5 + firstCol];    result += table[((secondRow - 1 + 5) % 5) \* 5 + secondCol];          } else {              result += table[firstRow \* 5 + secondCol];              result += table[secondRow \* 5 + firstCol];          }}      if (result[result.length - 1] === 'x') {          result = result.slice(0, -1);  }      return result;} |

Листинг 1.4 – Функция расшифрования по методу Трисемуса

Результат выполнения шифрования и расшифрования по методу Трисемуса приведен на рисунке 1.2.

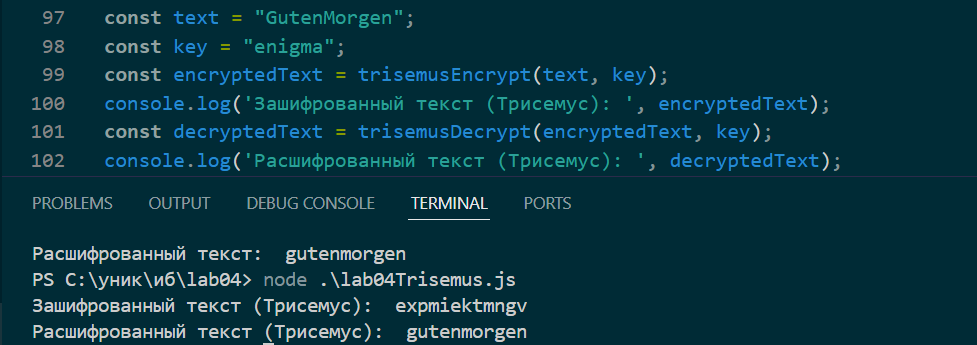


Рисунок 1.2 – Результат выполнения функций по методу Трисемуса

# **2 Гистограммы**

Сформировать гистограммы частот появления символов для исходного и зашифрованного сообщений.

Для формирования гистограмм в приложении L\_LUX, было выполнено необходимое шифрование и расшифрования сообщения. Результат формирования гистограмм приведен на рисунке 2.1

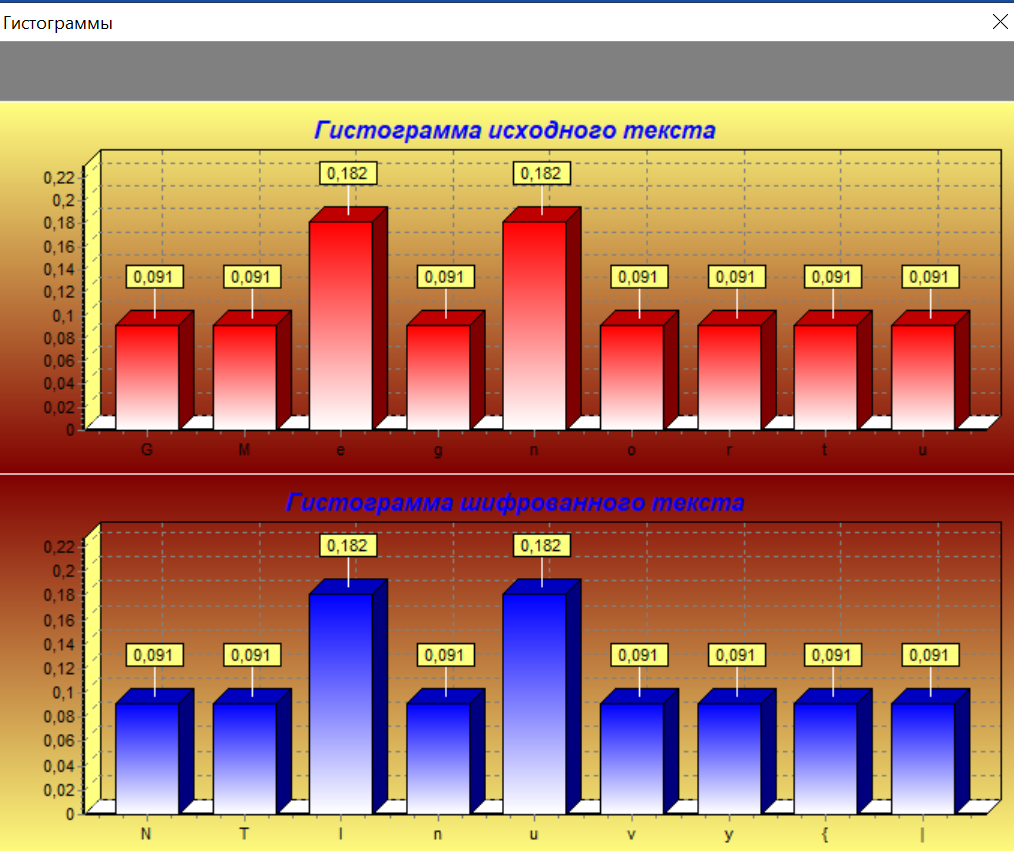


Рисунок 2.1 – Гистограммы частот появления символов по Цезарю

Результат формирования гистограмм по методу Трисемуса приведен на рисунке 2.2.

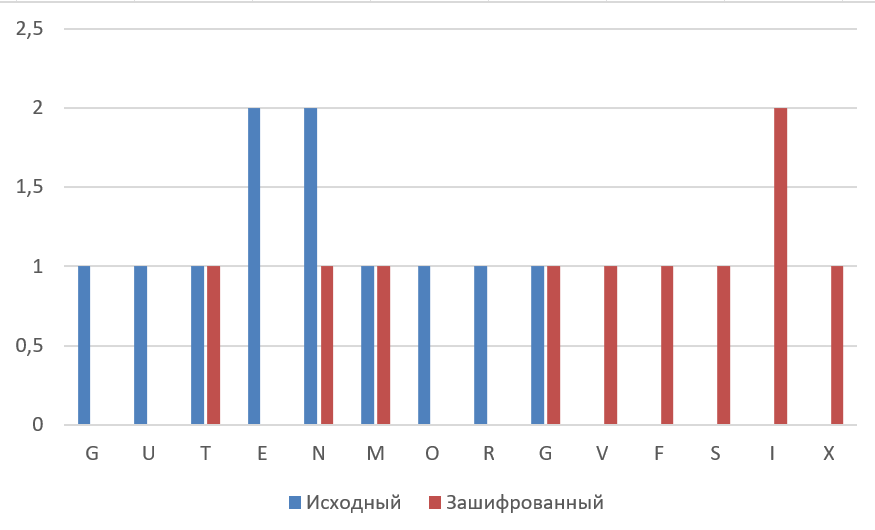


Рисунок 2.2 – Гистограмма частот появления символов по Трисемусу

# **3 Оценка времени**

Оценить время выполнения операций зашифрования/расшифрования (напоминание: во многих языках программирования есть встроенные методы для замеров времени; при отсутствии такового в используемом языке можно воспользоваться разностью двух дат (например, в миллисекундах: время после выполнения программы – время до начала выполнения преобразования)).

Для оценки времени выполнения операций шифрования и расшифрования в JavaScript мы использовали встроенный метод performance.now(), который позволяет измерять время с высокой точностью до микросекунд. Перед началом выполнения операции фиксируется текущее время, после завершения снова вызывается performance.now(), чтобы получить конечное значение. Разница между конечным и начальным временем показывает, сколько миллисекунд заняла операция. Этот метод обеспечивает точную оценку производительности алгоритма и помогает выявить возможные узкие места при обработке больших объемов данных.

Оценка времени выполнения операций шифра Цезаря представлен на рисунке 3.1, на рисунке 3.2 оценка времени для метода Трисемуса.

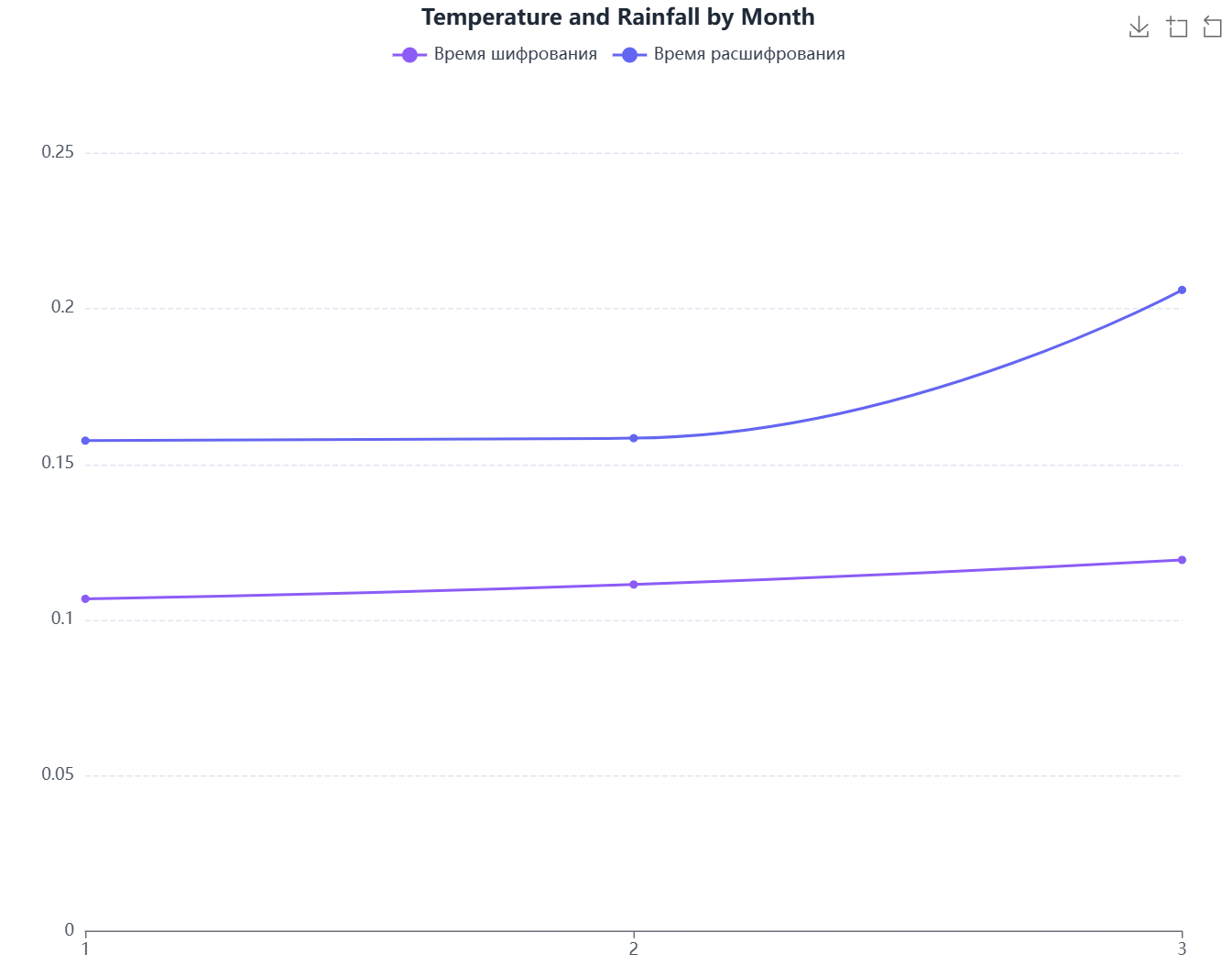
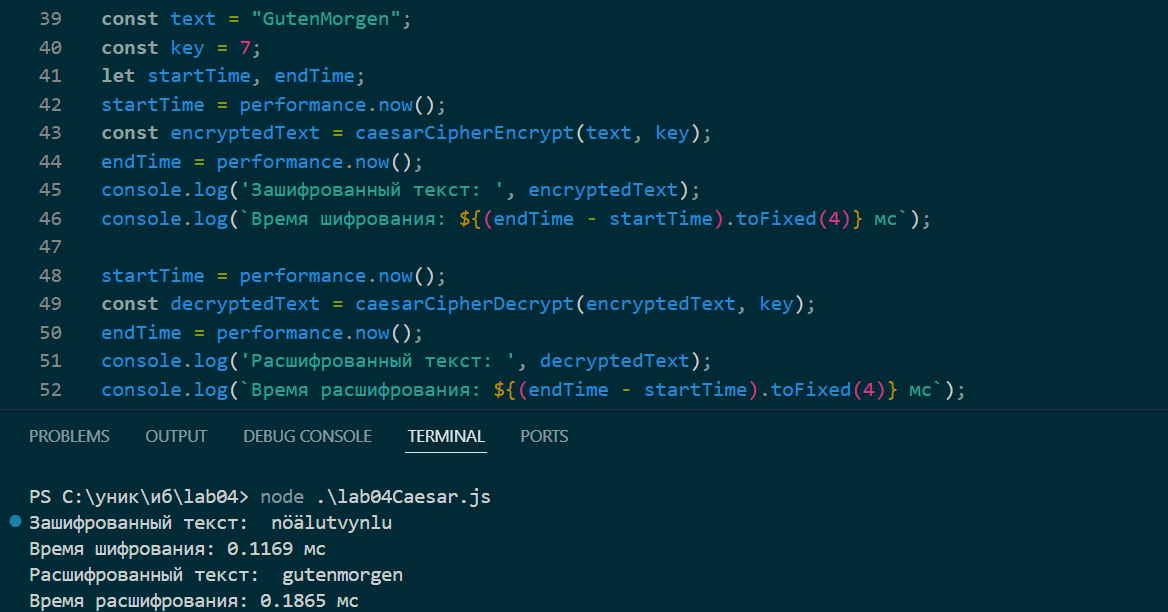


Рисунок 3.1 – Оценка времени операций шифра Цезаря

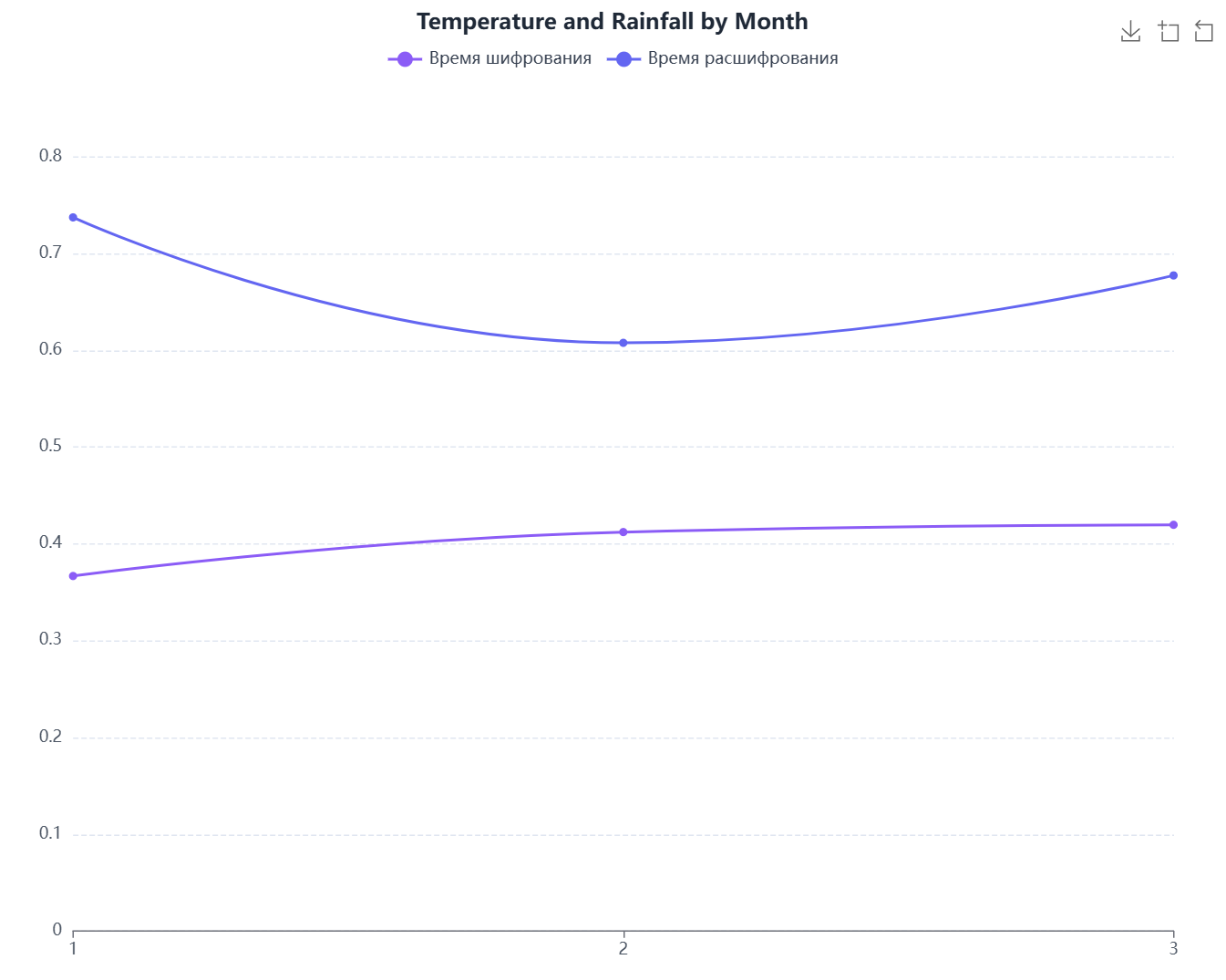
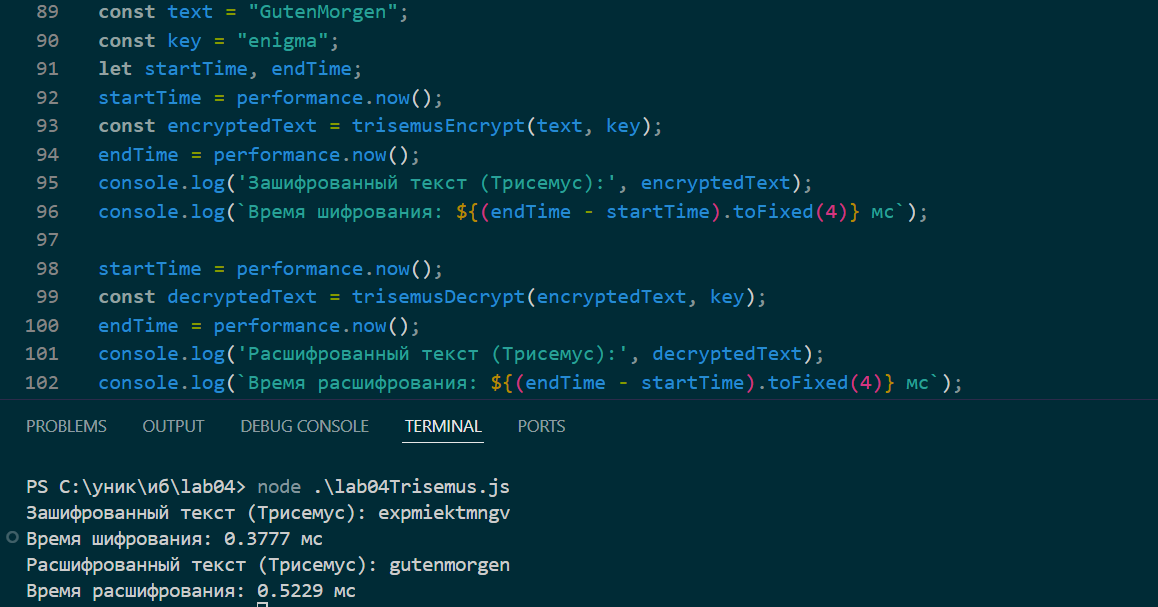


Рисунок 3.2 – Оценка времени операций по методу Трисемуса

# **Вывод**

В ходе лабораторной работы были изучены и реализованы подстановочные шифры, включая шифр Цезаря и шифр Трисемуса. Разработанное приложение успешно выполняет зашифрование и расшифрование текстовых сообщений, а также позволяет анализировать криптостойкость алгоритмов на основе частотного распределения символов. Проведены эксперименты по измерению скорости работы алгоритмов, что позволило оценить их эффективность. Полученные результаты подтверждают, что подстановочные шифры уязвимы к анализу частот, однако могут использоваться для обеспечения базового уровня защиты информации.