Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Информационная безопасность**

Студент: Козека Е. М.

ФИТ 3 курс 4 группа

Преподаватель: Нистюк О. А.

Минск 2025

**Лабораторная работа №4. Исследование криптографических шифров на основе подстановки (замены) символов**

**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.

*Типы подстановочных шифров*:

• моноалфавитных (шифры однозначной замены или простые подстановочные);

• полиграммных;

• омофонических (однозвучные шифры или шифры многозначной замены);

• полиалфавитных.

В ***моноалфавитных шифрах*** операция замены производится раздельно над каждым одиночным символом сообщения Мi. Для наглядной демонстрации шифра простой замены достаточно выписать под заданным алфавитом тот же алфавит, но в другом порядке или, например, со смещением. Записанный таким образом алфавит называют алфавитом замены. Максимальное количество ключей для любого шифра этого вида не превышает N!, где N – количество символов в алфавите.

Соотношениям *y* ≡ *x + k* mod *N* (где *x*, *y* – индекс (порядковый номер, начиная с 0) символа в используемом алфавите; *k* – ключ) и *х* ≡ *у – k* mod *N* соответствует классический шифр подстановки – ***шифр Цезаря***.

Существуют различные модификации шифра Цезаря, в частности, *Атбаш* и *лозунговый шифр*.

Шифр ***Атбаш*** состоит в замене каждой буквы другой буквой, которая находится в алфавите на таком же расстоянии от конца алфавита, как оригинальная буква – от начала. Например, буква А заменяется на Я, буква Б – на Ю и т. д.

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

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

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*.

Расшифрование основано на использовании соотношения

*x* ≡ *a*–1 (*y + N – b*) mod *N*,

где *a*–1 – обратное к *a* число по модулю *N*; *аa*–1 ≡ 1 mod *N*.

В ***полиграммных шифрах*** одна подстановка соответствует сразу нескольким символам исходного текста. Первым известным шифром этого типа является *шифр Порты*. Шифр представляется в виде таблицы. Наверху горизонтально и слева вертикально записывается стандартный алфавит. В ячейках таблицы записываются числа в определенном порядке. Шифрование выполняется парами букв исходного сообщения. Первая буква пары указывает на строку, вторая – на столбец. В случае нечетного количества букв в сообщении *Мi* к нему добавляется вспомогательный символ, например «А».

***Омофонические шифры*** создавались с целью увеличить сложность частотного анализа шифртекстов путем маскировки реальных частот появления символов текста с помощью *омофонии*. Пример омофонического шифра – *книжный шифр*.

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

***Диск Альберти.*** В «Трактате о шифрах» Альберти приводит первое точное описание *многоалфавитного шифра* на основе *шифровального диска*. Он состоял из двух дисков – внешнего неподвижного и внутреннего подвижного, на которые были нанесены буквы алфавита. Процесс шифрования заключался в нахождении буквы открытого текста на внешнем диске и замене ее на букву с внутреннего диска, стоящую под ней. После этого внутренний диск сдвигался на одну позицию, и шифрование второй буквы производилось уже по-новому шифралфавиту. Ключом данного шифра являлся порядок расположения букв на дисках и начальное положение внутреннего диска относительно внешнего.

***Таблица Трисемуса***. В 1518 г. Аббат Иоганнес Трисемус, настоятель монастыря в Вюрцбурге, написал книгу «Полиграфия», в которой он описал ряда шифров, один из которых развивает идею *многоалфавитной подстановки*. 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.

В указанной книге Трисемус впервые систематически описал применение шифрующих таблиц, заполненных алфавитом в случайном порядке. Для получения такого шифра подстановки обычно использовались таблица для записи букв алфавита и *ключевое слово* (или фраза). Можно найти определенную аналогию с системой шифрования Цезаря с ключевым словом. В таблицу сначала вписывалось по стрелкам ключевое слово, причем повторяющиеся буквы также отбрасывались. Затем эта таблица дополнялась не вошедшими в нее буквами алфавита по порядку. Модулярная арифметика так же коммутативна, ассоциативна и дистрибутивна, как и обычная арифметика. В силу этих свойств сравнения можно почленно складывать, вычитать, умножать, возводить в степень.

Таким образом, ключом в таблицах Трисемуса является ключевое слово и размер таблицы. При шифровании 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. Указанный размер таблицы для алфавита русского языка может соответствовать 4×8 либо 8×4.

***Шифр Виженера***. В 1586 г. французский дипломат Блез Виженер представил перед комиссией Генриха III описание простого, но довольно стойкого шифра, в основе которого лежит таблица Трисемуса. 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. Таким образом, при мощности алфавита (английского языка), равной 26, необходимо выполнить последовательно 25 сдвигов для формирования всей таблицы.

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

При ***атаке с известным шифртекстом*** предполагается, что противник знает алгоритм шифрования, у него имеется набор перехваченных шифрограмм, но он не знает секретный ключ.

Разновидности такой атаки:

• *полный перебор ключей* (лобовая атака);

• *атака по словарю*, перебор ключей по словарю; применяется часто для взлома паролей;

• *частотный криптоанализ* – метод взлома шифра, основывающийся на предположении о существовании зависимости между частотой появления символов алфавита в открытых сообщениях и соответствующих шифрозамен в шифрограммах.

**Ход работы**

**Задание 1.** Разработать авторское приложение в соответствии с целью лабораторной работы. Приложение должно выполнять зашифрование и 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.

Для начала, был реализован алгоритм шифрования и расшифрования с помощью шифра Цезаря с ключевым словом. Функция шифрования caesarEncode принимает на вход исходный текст и алфавит подстановки.

const caesarEncode = (text, cipherAlphabet) => {

    let resultCipher = "";

    text = text.toLowerCase();

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

        const currentChar = text[i];

        if (ENGLISH\_ALPHABET.includes(currentChar)) {

            let originalIndex = ENGLISH\_ALPHABET.indexOf(currentChar);

            resultCipher += cipherAlphabet[originalIndex];

        }

        else {

            resultCipher += currentChar;

        }

    }

    return resultCipher;

};

Листинг 4.1 – Функция caesarEncode

Функция расшифрования caesarDecode принимает на вход зашифрованный текст и алфавит подстановки.

const caesarDecode = (text, cipherAlphabet) => {

    let originalText = "";

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

        const currentChar = text[i];

        if (cipherAlphabet.includes(currentChar)) {

            let originalIndex = cipherAlphabet.indexOf(currentChar);

            originalText += ENGLISH\_ALPHABET[originalIndex];

        }

else {

            originalText += currentChar;

        }

    }

    return originalText;

};

Листинг 4.2 – Функция caesarDecode

Функция formCaesarAlphabet принимает на вход ключевое слово, алфавит подстановки и позицию для подстановки ключа.

const formCaesarAlphabet = (keyword, alphabet, shift) => {

let uniqueKeyword = [...new Set(keyword)].join('');

let filteredAlphabet = alphabet.split('').filter(letter =>

!uniqueKeyword.includes(letter)).join('');

let insertPart = uniqueKeyword.slice(0, alphabet.length - shift);

let overflowPart = uniqueKeyword.slice(alphabet.length - shift);

let resultAlphabet =

overflowPart +

filteredAlphabet.slice(0, shift) +

insertPart +

filteredAlphabet.slice(shift);

return resultAlphabet;

};

Листинг 4.3 – Функция formCaesarAlphabet

Результат работы функций приведен на рисунке 4.1.

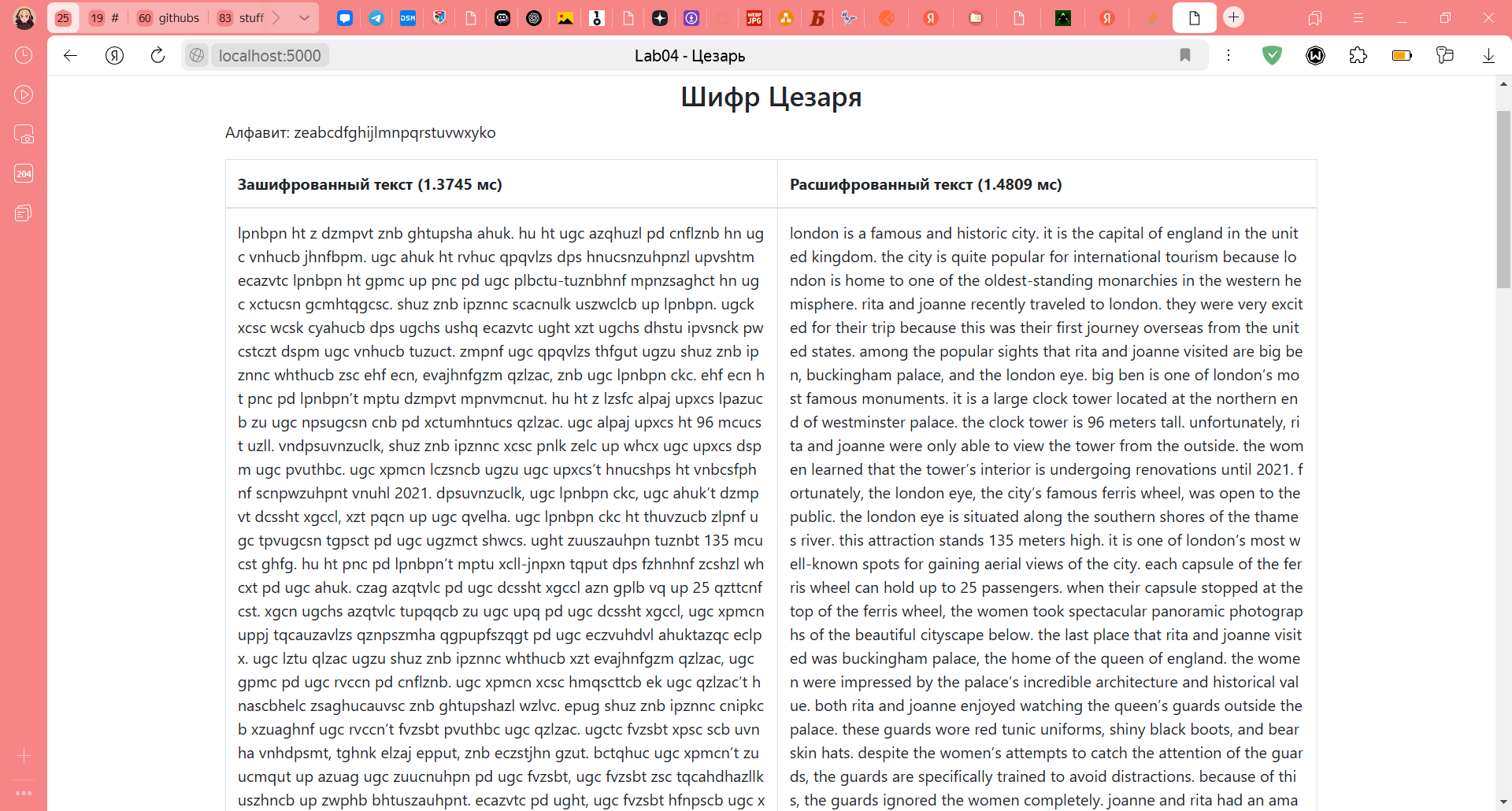


Рисунок 4.1 – Результат шифрования и дешифрования текста с помощью шифра Цезаря с ключевым словом «kozeka» и a = 24

Затем были разработаны несколько функций, выполняющих шифрование и дешифрование текста с использованием таблицы Трисемуса, где ключевое слово – собственное имя.

Сначала была разработана функция formTrithemiusTable, которая принимает размер таблицы и с помощью ключевого слова формирует результирующую таблицу Трисемуса.

const formTrithemiusTable = (k, m) => {

let resultTable = [];

let keyword = TRITHEMIUS\_KEYWORD.split('');

let alphabet = ENGLISH\_ALPHABET.split('');

for (let i = 0; i < k; i++) {

let row = [];

for (let j = 0; j < m; j++) {

if (keyword.length !== 0) {

row.push(keyword.shift());

}

else {

let symbol = alphabet.shift();

while (TRITHEMIUS\_KEYWORD.includes(symbol)) {

symbol = alphabet.shift();

}

row.push(symbol);

}

}

resultTable.push(row);

}

return resultTable;

};

Листинг 4.4 – Функция formTrithemiusTable

Функция шифрования trithemiusEncode принимает на вход исходный текст и таблицу Трисемуса.

const trithemiusEncode = (text, trithemiusTable) => {

    let result = "";

    text = text.toLowerCase();

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

        if (ENGLISH\_ALPHABET.includes(text[i])) {

            let currentIndex = getIndexOfK(trithemiusTable, text[i]);

            if (currentIndex[0] === -1) {

                result += text[i];

                continue;

            }

            let newRow = (currentIndex[0] + 1) %

trithemiusTable.length;

            result += trithemiusTable[newRow][currentIndex[1]];

        }

else {

            result += text[i];

        }

    }

    return result;

};

Листинг 4.5 – Функция trithemiusEncode

Функция расшифрования trithemiusDecode принимает на вход зашифрованный текст и таблицу Трисемуса.

const trithemiusDecode = (text, trithemiusTable) => {

    let originalText = "";

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

        let currentIndex = getIndexOfK(trithemiusTable, text[i]);

        if (currentIndex[0] === -1) {

            originalText += text[i];

            continue;

        }

        let newRow = currentIndex[0] - 1;

        if (newRow < 0) {

            newRow = trithemiusTable.length - 1;

        }

        if (trithemiusTable[newRow][currentIndex[1]]) {

            originalText += trithemiusTable[newRow][currentIndex[1]];

        }

        else {

            console.error(`Ошибка: индекс [${newRow}, ${currentIndex[1]}] выходит за границы!`);

            originalText += text[i];

        }

    }

    return originalText;

};

Листинг 4.6 – Функция trithemiusDecode

Результат работы функций приведен на рисунке 4.2.

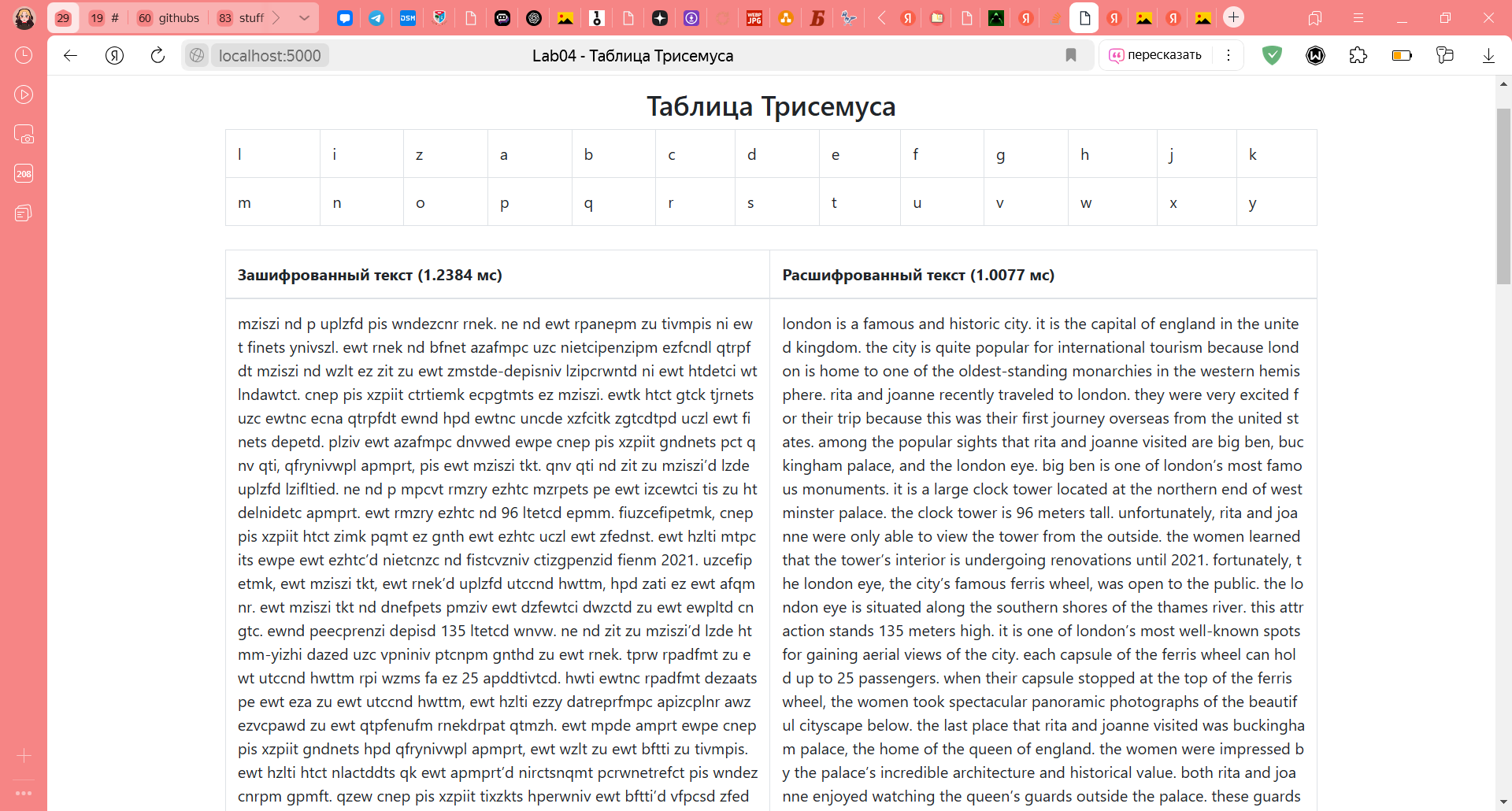


Рисунок 4.2 – Результат шифрования и дешифрования текста с помощью таблицы Трисемуса с ключевым словом «liza»

**Задание 2.** Сформировать гистограммы частот появления символов для исходного и зашифрованного сообщений. Оценить время выполнения операций зашифрования/расшифрования.

***Шифр Цезаря:***

Гистограмма частот появления символов исходного сообщения приведена на рисунке 4.3.

Рисунок 4.3 – Гистограмма частот появления символов исходного сообщения

Гистограмма частот появления символов зашифрованного сообщения приведена на рисунке 4.4.

Рисунок 4.4 – 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 1.5-2 миллисекунды.

***Таблица Трисемуса:***

Гистограмма частот появления символов исходного сообщения приведена на рисунке 4.5.

Рисунок 4.5 – Гистограмма частот появления символов исходного сообщения

Гистограмма частот появления символов зашифрованного сообщения приведена на рисунке 4.6.

Рисунок 4.6 – 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 таблицы Трисемуса выполняется примерно за 1-5 миллисекунд, что уступает по сравнению с шифрованием и расшифрованием с помощью шифра Цезаря.

**Вывод:** в ходе выполнения лабораторной работы были изучены основных принципов работы подстановочных шифров, методов их зашифрования и расшифрования. Также было разработано приложение для реализации указанных методов подстановочного зашифрования и расшифрования. Было выполнено исследование криптостойкости шифров на основе статистических данных о частотах появления символов в исходном и зашифрованном сообщениях. Было подтверждено, что шифры подстановки имеют низкую криптостойкость. Была проведена оценка скорости зашифрования и расшифрования реализованных способов шифров. Это позволило сравнить эффективность различных алгоритмов.