Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

ОТЧЕТ

к лабораторной работе № 1

на тему

**ШИФР ЦЕЗАРЯ. ШИФР ВИЖЕНЕРА.**

Выполнил:

студент гр. 153503

Бобко И.В.

Проверил:

Лещенко Е.А.

Минск 2024

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc157598705)

[2 Теоретические сведения 4](#_Toc157598706)

[3 Полученные результаты 6](#_Toc157598712)

[Выводы 8](#_Toc157598713)

[Приложение А (обязательное) Листинг кода 9](#_Toc157598714)

[Приложение Б (обязательное) Блок-схема алгоритма 12](#_Toc157598715)

# **1 ЦЕЛЬ РАБОТЫ**

Целью выполнения данной лабораторной работы является реализация программных средств шифрования и дешифрования текстовых файлов используя такие методы шифрования как Шифр Цезаря, (шифр сдвига, код Цезаря) и шифр Виженера.

**2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ** 

**Шифр Цезаря**

Шифр Цезаря, также известный, как шифр сдвига, код Цезаря или сдвиг Цезаря – один из самых простых и наиболее широко известных методов шифрования.

Шифр Цезаря – это вид шифра подстановки, в котором каждый символ в открытом тексте заменяется символом, находящимся на некотором постоянном числе позиций левее или правее него в алфавите. Например, в шифре со сдвигом 4 А была бы заменена на Г, Б станет Д, и так далее.

Шифр назван в честь римского императора Гая Юлия Цезаря, использовавшего его для секретной переписки со своими генералами.

Шаг шифрования, выполняемый шифром Цезаря, часто включается как часть более сложных схем, таких как шифр Виженера, и все ещё имеет современное приложение в системе *ROT13*. Как и все моноалфавитные шифры, шифр Цезаря легко взламывается и не имеет практически никакого применения на практике.



Рисунок 1 – Шифр Цезаря

***Математическая модель***

Если сопоставить каждому символу алфавита его порядковый номер (нумеруя с 0), то шифрование и дешифрование можно выразить формулами модульной арифметики:

y=(x+k)\ \mod\ n

x=(y-k+n)\ \mod\ n,

где ~x – символ открытого текста, ~y – символ шифрованного текста, ~n – мощность алфавита, а ~k – ключ.

**Шифр Виженера**

Шифр Виженера состоит из последовательности нескольких шифров Цезаря с различными значениями сдвига. Для зашифровывания может использоваться таблица алфавитов, называемая *tabula recta* или квадрат (таблица) Виженера. Применительно к латинскому алфавиту таблица Виженера составляется из строк по 26 символов, причём каждая следующая строка сдвигается на несколько позиций. Таким образом, в таблице получается 26 различных шифров Цезаря. На каждом этапе шифрования используются различные алфавиты, выбираемые в зависимости от символа ключевого слова.

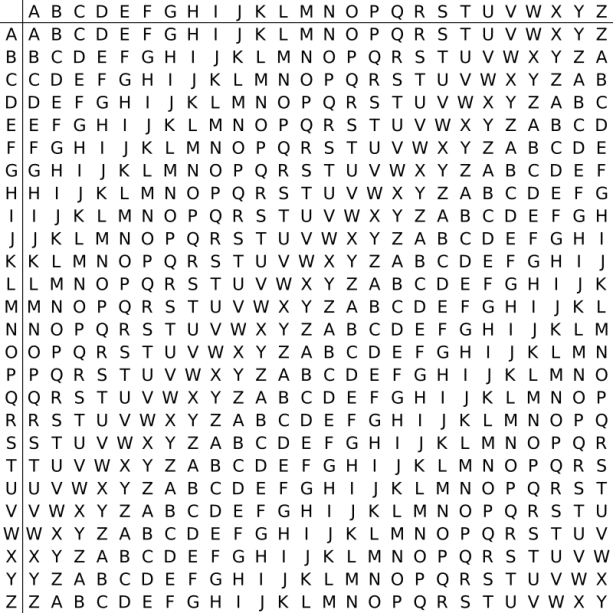


Рисунок 2 – Квадрат Виженера

Расшифровывание производится следующим образом: находим в таблице Виженера строку, соответствующую первому символу ключевого слова; в данной строке находим первый символ зашифрованного текста. Столбец, в котором находится данный символ, соответствует первому символу исходного текста. Следующие символы зашифрованного текста расшифровываются подобным образом.

Операция расшифровывания в компьютере соответствует сложению кодов *ASCII* символов сообщения и ключа по некоторому модулю. Кажется, что если таблица будет более сложной, чем циклическое смещение строк, то шифр станет надежнее. Это действительно так, если ее менять чаще, например, от слова к слову. Но составление таких таблиц, представляющих собой латинские квадраты, где любая буква встречается в строке или столбце один раз, трудоемко и его стоит делать лишь на ЭВМ.

# **3 ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ**

В ходе выполнения данной лабораторной работы было разработано программное средство, читающее данные из файла и шифрующие (дешифрующие) их с помощью с помощью Шифра Цезаря и шифра Виженера.

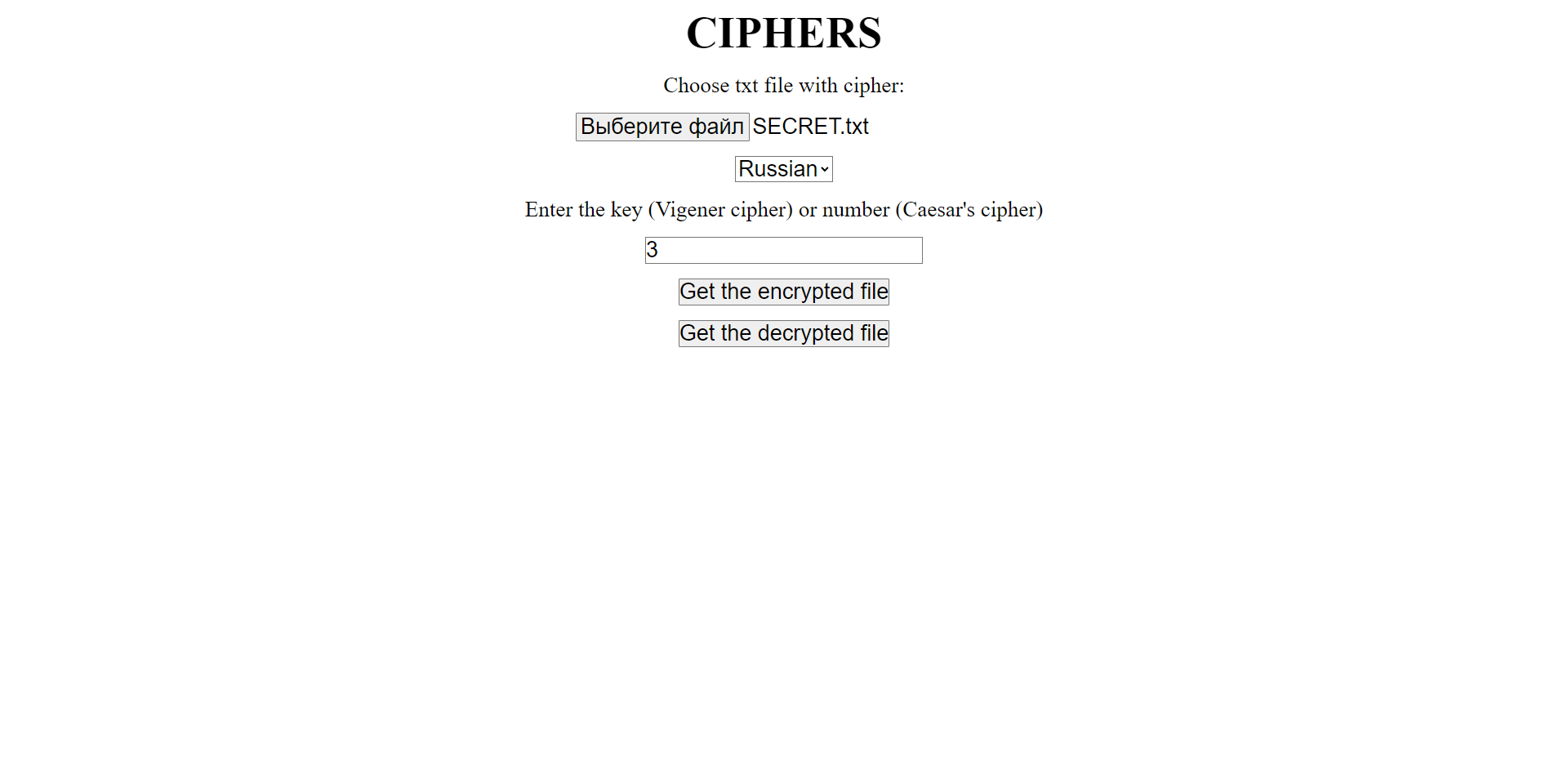


Рисунок 1 – Интерфейс программы

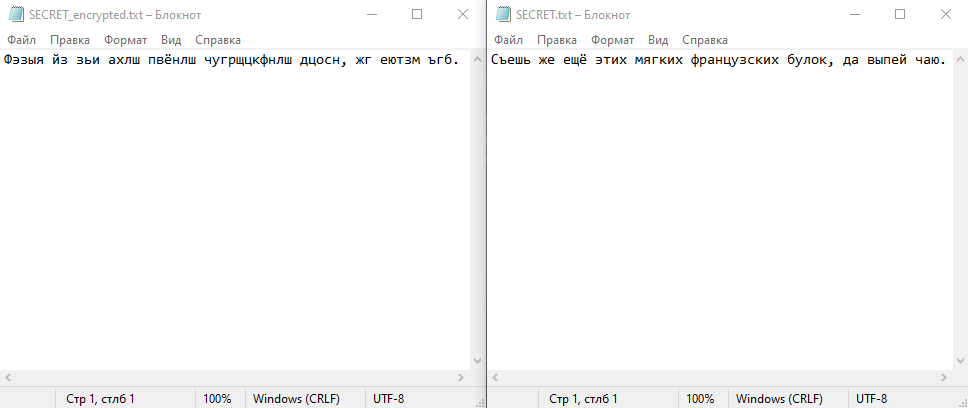


Рисунок 2 – Результат шифрования шифром Цезаря с ключом 3

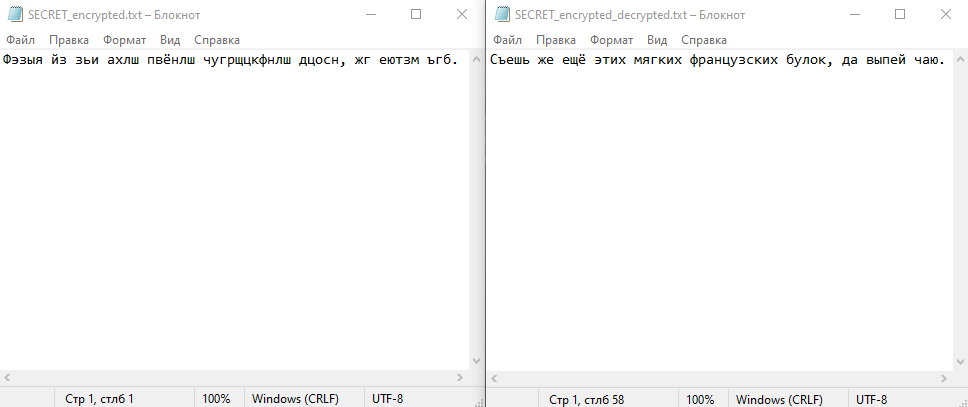


Рисунок 3 – Результат дешифрования шифром Цезаря с ключом 3

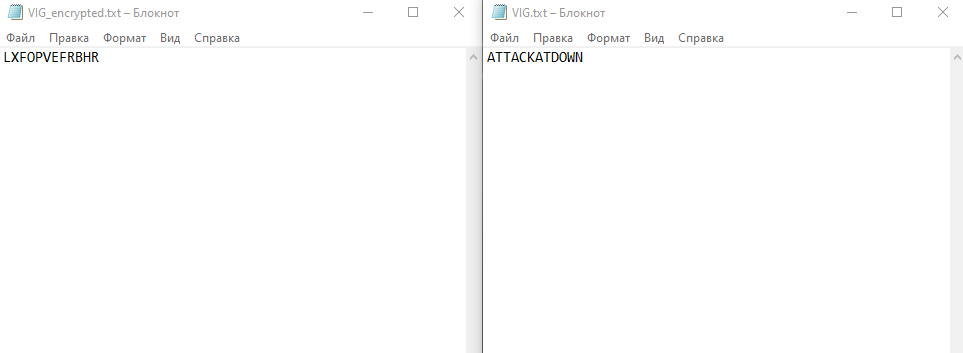


Рисунок 4 – Результат шифрования шифром Виженера с ключом LEMON

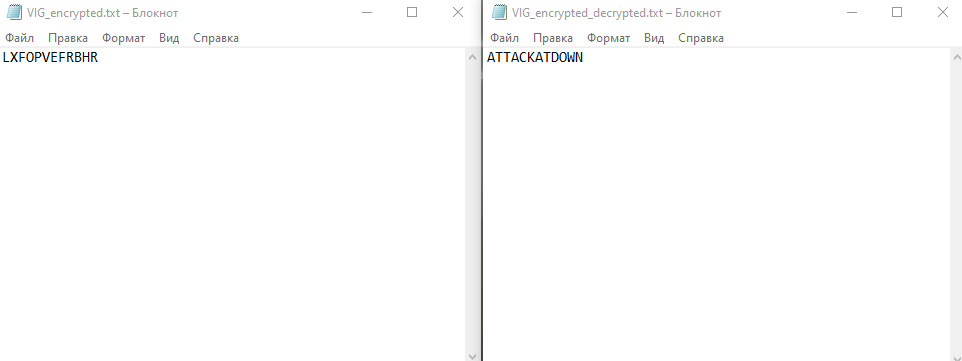


Рисунок 5 – Результат дешифрования шифром Виженера с ключом LEMON

# **ВЫВОДЫ**

В результате выполнения данной лабораторной работы было разработано программное средство для шифрования и дешифрования текстовых файлов используя такие методы шифрования как Шифр Цезаря, (шифр сдвига, код Цезаря) и шифр Виженера.

# **ПРИЛОЖЕНИЕ А (обязательное) Листинг кода**

Листинг 1 **–** Файл *script.js*

const getEncryptedFile = document.getElementById('getEncryptedFile');

const getDecryptedFile = document.getElementById('getDecryptedFile');

const keyInput = document.getElementById('keyInput');

const fileInput = document.getElementById('fileInput');

const languageSelect = document.getElementById('languageSelect');

const ALPHABETS = new Map();

const VIGINER\_SQUARES = new Map();

const LANGUAGE\_REGEX = new Map();

const LETTER\_REGEX = /^[A-Za-zА-Яа-яЁё]$/;

const DIGITS\_REGEX = /^\d+$/;

ALPHABETS.set('en', 'ABCDEFGHIJKLMNOPQRSTUVWXYZ');

ALPHABETS.set('ru', 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ');

LANGUAGE\_REGEX.set('en', /^[A-Za-z]+$/);

LANGUAGE\_REGEX.set('ru', /^[А-Яа-яЁё]+$/);

function generateSquares() {

    for (let [key, value] of ALPHABETS) {

        let square = [];

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

            let row = value.slice(i);

            row += value.slice(0, i);

            square.push(row);

        }

        VIGINER\_SQUARES.set(key, square);

    }

}

generateSquares();

function repeatString(*firstString*, *secondString*, *langRegex*) {

    let resultString = "";

    let firstStringLength = *firstString*.length;

    let it = 0;

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

        let currentChar = *secondString*[i];

        if (*langRegex*.test(currentChar)) {

            resultString += *firstString*[it];

            it = (it + 1) % firstStringLength;

        }else{

            resultString += currentChar;

        }

    }

    return resultString;

}

function encryptText(*text*, *key*, *viginerSquare*, *langRegex*) {

    let encryptedText = '';

    let newKey = repeatString(*key*, *text*, *langRegex*);

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

        let i = *viginerSquare*[0].indexOf(*text*[it].toUpperCase());

        let j = *viginerSquare*[0].indexOf(newKey[it]);

        if(i === -1 || j === -1)

            encryptedText += *text*[it];

        else {

            encryptedText +=

                (*text*[it] === *text*[it].toUpperCase()) ?

*viginerSquare*[i][j] :

*viginerSquare*[i][j].toLowerCase();

        }

    }

    return encryptedText;

}

function decryptText(*text*, *key*, *viginerSquare*, *langRegex*) {

    let decryptedText = '';

    let newKey = repeatString(*key*, *text*, *langRegex*);

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

        let i = *viginerSquare*[0].indexOf(newKey[it]);

        if(i === -1)

            decryptedText += *text*[it];

        else {

            let j = *viginerSquare*[i].indexOf(*text*[it].toUpperCase());

            decryptedText +=

                (*text*[it] === *text*[it].toUpperCase()) ?

*viginerSquare*[0][j] :

*viginerSquare*[0][j].toLowerCase();

        }

    }

    return decryptedText;

}

function readFile(*event*) {

    let action = *event*.target.id;

    const file = fileInput.files[0];

    if (!file) {

      alert('Please choose the file');

      return;

    }

    const reader = new FileReader();

    reader.onload = function(*event*) {

        const text = *event*.target.result;

        const language = languageSelect.value;

        let key = keyInput.value.toUpperCase();

        const onlyLetters = LANGUAGE\_REGEX.get(language).test(key);

        const onlyDigits = DIGITS\_REGEX.test(key);

        if (!onlyLetters && !onlyDigits) {

            alert('Error: The value must contain either only letters of selected alphabet or only numbers');

            return;

        }

        if(onlyDigits)

            key = VIGINER\_SQUARES.get(language)[0][key];

        let result = (action === 'getEncryptedFile') ?

            encryptText(text, key, VIGINER\_SQUARES.get(language), LANGUAGE\_REGEX.get(language)) :

            decryptText(text, key, VIGINER\_SQUARES.get(language), LANGUAGE\_REGEX.get(language));

        console.log(result);

        (action === 'getEncryptedFile') ?

        saveFile(file.name.replace(/\.[^/.]+$/, '\_encrypted.txt'), result) :

        saveFile(file.name.replace(/\.[^/.]+$/, '\_decrypted.txt'), result);

    };

    reader.onerror = function(*event*) {

        alert(*event*.target.text);

    };

    reader.readAsText(file);

}

function saveFile(*fileName*, *text*) {

    const blob = new Blob([*text*], {type: 'text/plain'});

    const url = URL.createObjectURL(blob);

    const link = document.createElement('a');

    link.href = url;

    link.download = *fileName*;

    link.click();

    URL.revokeObjectURL(url);

}

getEncryptedFile.addEventListener('click', readFile);

getDecryptedFile.addEventListener('click', readFile);

# **ПРИЛОЖЕНИЕ Б (обязательное) Блок-схема алгоритма**