МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-приложений)

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ:

по дисциплине «Информационная безопасность»

Исполнитель

студент (ка) 3 курса группы 6 Розель Станислав Александрович

(Ф.И.О.)

Минск 2025

1. **Метод маршрутных перестановок**

В ходе лабораторной работы нужно было создать программу, реализующую метод маршрутных перестановок, с маршрутом на запись по строкам, на чтение по столбцам. Количество строк и столбцов должен задаваться свободно. По условию нужно использовать белорусский язык. На листинге 1.1 представлен код для реализации данного метода.

|  |
| --- |
| function routeTranspositionEncrypt(text, rows, cols) {  let cipherText = '';  const blockSize = rows \* cols;  for (let start = 0; start < text.length; start += blockSize) {  let block = text.substring(start, start + blockSize);  while (block.length < blockSize) {  block += ' '; // дополнение пробелами  }  const matrix = [];  for (let i = 0; i < rows; i++) {  matrix[i] = block.slice(i \* cols, (i + 1) \* cols).split('');  }  for (let j = 0; j < cols; j++) {  for (let i = 0; i < rows; i++) {  cipherText += matrix[i][j];  }  }  }  return cipherText;  }    function routeTranspositionDecrypt(cipherText, rows, cols) {  let plainText = '';  const blockSize = rows \* cols;  for (let start = 0; start < cipherText.length; start += blockSize) {  let block = cipherText.substring(start, start + blockSize);  const matrix = [];  for (let j = 0; j < cols; j++) {  for (let i = 0; i < rows; i++) {  if (!matrix[i]) matrix[i] = [];  matrix[i][j] = block[j \* rows + i] || ' ';  }  }  for (let i = 0; i < rows; i++) {  plainText += matrix[i].join('');  }  }  return plainText.trim();  } |

Листинг 1.1 – Реализация метода маршрутных перестановок.

Так же нужно было по заданию выполнить сравнение количества символов в исходном тексте и зашифрованном. В таблице 1.1 представлено сравнение количества символов.

Таблица 1.1 – Сравнение количества символов

|  |  |
| --- | --- |
| Исходный текст | Зашифрованный текст |
|  |  |

Так же в ходе лабораторной работы было измерено время занимаемое на зашифрование и расшифрование текста:

* Время шифрования: 0.977ms
* Время расшифровки: 0.597ms

1. **Метод множественных перестановок**

В ходе лабораторной работы также был разобран метод множественных перестановок и был написан код, который реализует метод множественных перестановок. Ключевыми словами были: Станіслаў, Розель. На листинге 2.1 представлен код реализации функций шифрования и расшифрования.

|  |
| --- |
| function createKeyFromKeyword(keyword, alphabet) {  const uniqueChars = [...new Set(keyword.toLowerCase())].filter((char) =>  alphabet.includes(char)  );  const sortedChars = [...uniqueChars].sort();  return uniqueChars.map((char) => sortedChars.indexOf(char));  }    function multipleTranspositionEncrypt(text, key1, key2, alphabet) {  const perm1 = createKeyFromKeyword(key1, alphabet);  const perm2 = createKeyFromKeyword(key2, alphabet);    const cols1 = perm1.length;  const rows1 = Math.ceil(text.length / cols1);  while (text.length < rows1 \* cols1) {  text += ' ';  }    const matrix1 = [];  for (let i = 0; i < rows1; i++) {  matrix1.push(text.slice(i \* cols1, (i + 1) \* cols1).split(''));  }    const intermediateText = perm1  .map((col) => matrix1.map((row) => row[col]).join(''))  .join('');    const cols2 = perm2.length;  const rows2 = Math.ceil(intermediateText.length / cols2);  while (intermediateText.length < rows2 \* cols2) {  intermediateText += ' ';  }    const matrix2 = [];  for (let i = 0; i < rows2; i++) {  matrix2.push(intermediateText.slice(i \* cols2, (i + 1) \* cols2).split(''));  }    return perm2  .map((col) => matrix2.map((row) => row[col]).join(''))  .join('');  }    function multipleTranspositionDecrypt(cipherText, key1, key2, alphabet) {  const perm1 = createKeyFromKeyword(key1, alphabet);  const perm2 = createKeyFromKeyword(key2, alphabet);    const cols2 = perm2.length;  const rows2 = Math.ceil(cipherText.length / cols2);  const matrix2 = Array.from({ length: rows2 }, () =>  new Array(cols2).fill('')  );    let index = 0;  for (const col of perm2) {  for (let i = 0; i < rows2; i++) {  matrix2[i][col] = cipherText[index++] || ' ';  }  }    const intermediateText = matrix2.map((row) => row.join('')).join('');    const cols1 = perm1.length;  const rows1 = Math.ceil(intermediateText.length / cols1);  const matrix1 = Array.from({ length: rows1 }, () =>  new Array(cols1).fill('')  );    index = 0;  for (const col of perm1) {  for (let i = 0; i < rows1; i++) {  matrix1[i][col] = intermediateText[index++] || ' ';  }  }  return matrix1.map((row) => row.join('')).join('').trim();  }  return processedText;  } catch (error) {  console.error('Ошибка при обработке файла:', error.message);  throw error;  }  } |

Листинг 2.1 – Реализация метода множественных перестановок

В ходе лабораторной работы так же было проведено сравнение количества символов в исходном тексте и зашифрованном. Сравнение представлено в таблице 2.1.

Таблица 2.1 – Сравение количества символов

|  |  |
| --- | --- |
| Исходный текст | Зашифрованный текст |
|  |  |

Так же было измерено время для шифрования и расшифрования текста:

* Время шифрования: 1.766ms
* Время расшифровки: 1.512ms