МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Лабораторная работа № 5**

**ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ**

**ШИФРОВ НА ОСНОВЕ ПЕРЕСТАНОВКИ СИМВОЛОВ**

Разработала: Некрасова А.П.

ФИТ 3 курс 5 группа

Преподаватель: Савельева М.Г.

Минск 2023

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации перестановочных шифров (работа рассчитана на 4 часа аудиторных занятий).

**Задачи:**

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости перестановочных шифров.
2. Ознакомиться с особенностями реализации и свойствами различных перестановочных шифров на основе готового программного средства (L\_LUX).
3. Разработать приложение для реализации указанных преподавателем методов перестановочного 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.

**Практическое задание**

|  |  |  |
| --- | --- | --- |
| Вариант | Алфавит | Шифр |
| 4 | Немецкий | 1. Маршрутная перестановка (маршрут – змейкой; параметры таблицы – по указанию преподавателя) 2. Множественная перестановка, ключевые слова – собственные имя и фамилия |

***Метод маршрутной перестановки (маршрут – змейкой)***

Организация маршрутной перестановки змейкой представлена на рис. 1.

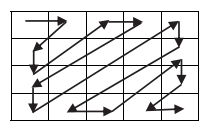


Рис. 1 – Графическое представление метода маршрутной

перестановки змейкой

Описание методов шифрования и расшифрования для данного метода представлены в листинге 1.

public string Encrypt()

{

Console.WriteLine(«------- Ш И Ф Р О В А Н И Е -------\n»);

char[,] table = createMatrix(text);

char[] result = new char[text.Length];

int x = 0, y = 0, l = 0;

while (true)

{

if (x < tableWidth)

{

if (table[y, x] != ‘\0’)

result[l++] = table[y, x++];

else

{

x++;

}

}

else

{

x++;

}

while (x != 0)

{

if (y < tableHeight && x < tableWidth)

{

if (table[y, x] != ‘\0’)

result[l++] = table[y++, x--];

else

{

y++; x--;

}

}

else

{

y++; x--;

}

}

if (y < tableHeight)

{

if (table[y, x] != ‘\0’)

result[l++] = table[y++, x];

else

{

y++;

}

}

else

{

y++;

}

while (y != 0)

{

if (y < tableHeight && x < tableWidth)

{

if (table[y, x] != ‘\0’)

result[l++] = table[y--, x++];

else

{

y--; x++;

}

}

else

{

y--; x++;

}

}

if (l == text.Length) break;

}

return createString(result);

}

public string Decrypt(string input)

{

Console.WriteLine(«------- Р А С Ш И Ф Р О В А Н И Е -------\n»);

char[,] table = new char[tableHeight, tableWidth];

int x = 0, y = 0, l = 0;

int last = tableHeight \* tableWidth – input.Length;

int downY = (last % tableWidth == last ? tableWidth : (tableWidth – last / tableWidth) – 1);

int downX = (last % tableHeight == last ? tableHeight : (tableHeight – last % tableHeight) – 1);

List<KeyValuePair<int, int>> abadonCells = new List<KeyValuePair<int, int>>();

for (int xa = tableHeight – 1, iter = last; ; xa--)

{

for (int ya = tableWidth – 1; ya >= 0; ya--)

{

abadonCells.Add(new KeyValuePair<int, int>(ya, xa));

last--;

if (last == 0) break;

}

if (last == 0) break;

}

while (true)

{

if (x < tableWidth)

{

if (abadonCells.Where(z => z.Key == x && z.Value == y).Count() == 0)

table[y, x++] = input[l++];

else

{

x++;

}

}

else

{

x++;

}

while (x != 0)

{

if (l == text.Length) break;

if (y < tableHeight && x < tableWidth)

{

if (abadonCells.Where(z => z.Key == x && z.Value == y).Count() == 0)

table[y++, x--] = input[l++];

else

{

y++; x--;

}

}

else

{

y++; x--;

}

}

if (y < tableHeight)

{

if (abadonCells.Where(z => z.Key == x && z.Value == y).Count() == 0)

table[y++, x] = input[l++];

else

{

y++;

}

}

else

{

y++;

}

while (y != 0)

{

if (l == text.Length) break;

if (y < tableHeight && x < tableWidth)

{

if (abadonCells.Where(z => z.Key == x && z.Value == y).Count() == 0)

table[y--, x++] = input[l++];

else

{

y--; x++;

}

}

else

{

y--; x++;

}

}

if (l == text.Length) break;

}

return createString(table);

}

Листинг 1 – Шифрование/расшифрование методом

маршрутной перестановки змейкой

Реализация кода программы:

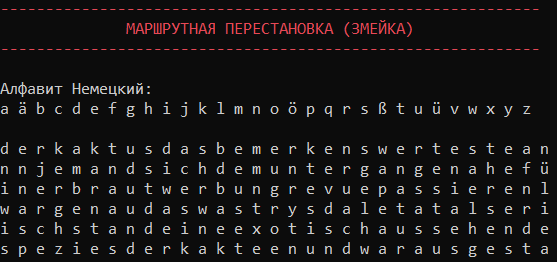
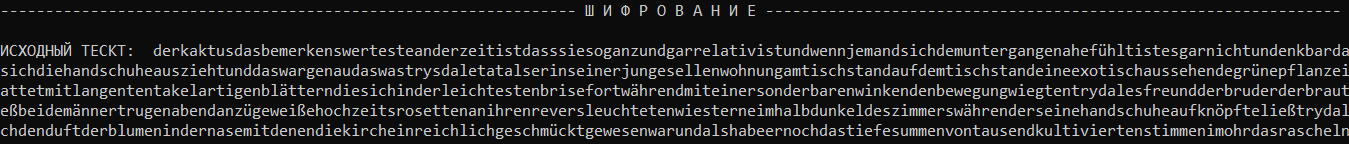


Рис. 2 – Представление текста в виде матрицы



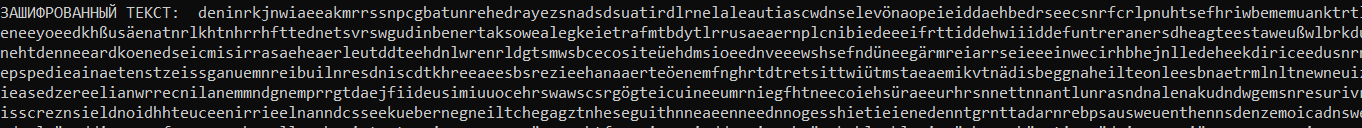


Рис. 3 – Результат выполнения шифрования

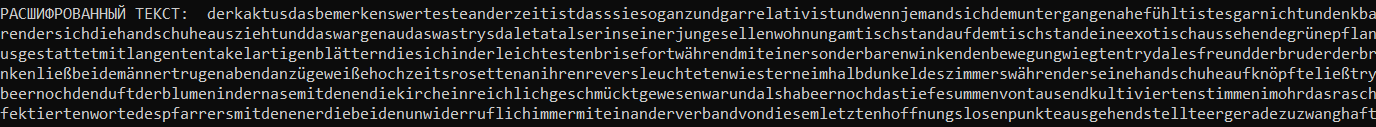
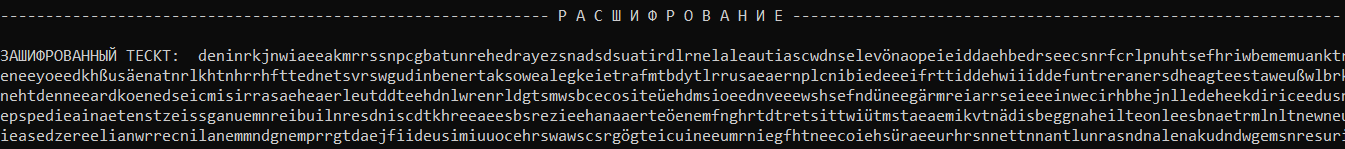


Рис. 4 – Результат выполнения расшифрования

**Метод множественной перестановки**

Описание методов шифрования и расшифрования для данного метода представлены в листинге 2.

public string Encrypt()

{

Console.WriteLine("-----------------------ШИФРОВАНИЕ-----------------------\n");

char[,] table = createMatrix(Text);

int tableHeight = keyVertical.Count + 2;

int tableWidth = KeyHorizontal.Count + 2;

char[,] result = new char[tableHeight, tableWidth];

int iteration = 0;

while (iteration++ < tableWidth - 2)

{

int k = 0;

for (int y = 0; y < tableHeight; y++)

{

while (k != 2)

{

result[y, k] = table[y, k];

k++;

}

k = keyHorizontal.IndexOf(keyHorizontal.Where(l => l.Key == iteration).First());

result[y, iteration + 1] = table[y, k + 2];

k = 0;

}

}

Console.WriteLine("Горизонтальная перестановка:\n");

printMatrix(result);

char[,] resultV = new char[tableHeight, tableWidth];

iteration = 0;

while (iteration++ < tableHeight - 2)

{

int k = 0;

for (int y = 0; y < tableWidth; y++)

{

while (k != 2)

{

resultV[k, y] = result[k, y];

k++;

}

k = keyVertical.IndexOf(keyVertical.Where(l => l.Key == iteration).First());

resultV[iteration + 1, y] = result[k + 2, y];

k = 0;

}

}

Console.WriteLine("Вертикальная перестановка:\n");

printMatrix(resultV);

encryptedMatrix = resultV;

return createString(encryptedMatrix);

}

public string Decrypt()

{

Console.WriteLine("---------------------РАСШИФРОВАНИЕ----------------------\n");

char[,] table = encryptedMatrix;

int tableHeight = keyVertical.Count + 2;

int tableWidth = KeyHorizontal.Count + 2;

char[,] resultV = new char[tableHeight, tableWidth];

int iteration = 0;

while (iteration++ < tableHeight - 2)

{

int k = 0;

for (int y = 0; y < tableWidth; y++)

{

while (k != 2)

{

resultV[k, y] = table[k, y];

k++;

}

k = keyVertical.IndexOf(keyVertical.Where(l => l.Key == iteration).First());

resultV[k + 2, y] = table[iteration + 1, y];

k = 0;

}

}

Console.WriteLine("Вертикальная перестановка:\n");

printMatrix(resultV);

char[,] result = new char[tableHeight, tableWidth];

iteration = 0;

while (iteration++ < tableWidth - 2)

{

int k = 0;

for (int y = 0; y < tableHeight; y++)

{

while (k != 2)

{

result[y, k] = resultV[y, k];

k++;

}

k = keyHorizontal.IndexOf(keyHorizontal.Where(l => l.Key == iteration).Min());

result[y, k + 2] = resultV[y, iteration + 1];

k = 0;

}

}

Console.WriteLine("Горизонтальная перестановка:\n");

printMatrix(result);

decryptedMatrix = result;

return createString(decryptedMatrix);

}

Листинг 2 – Шифрование/расшифрование методом

множественной перестановки

Реализация кода программы:

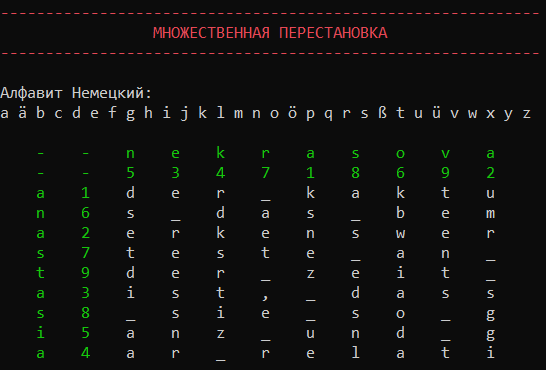


Рис. 5 – Изначальная матрица

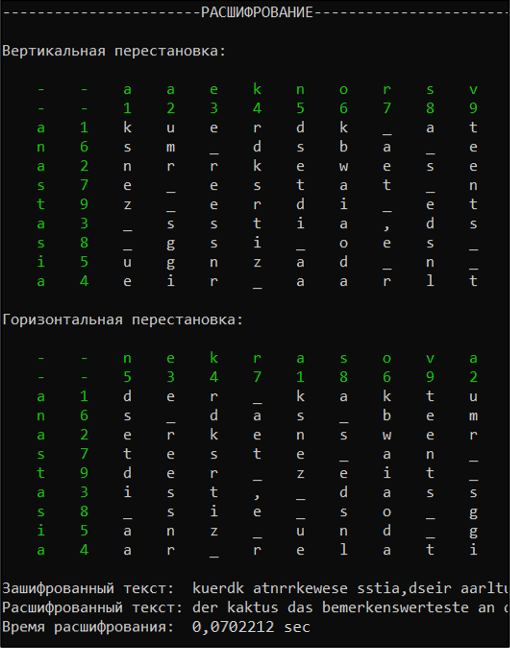
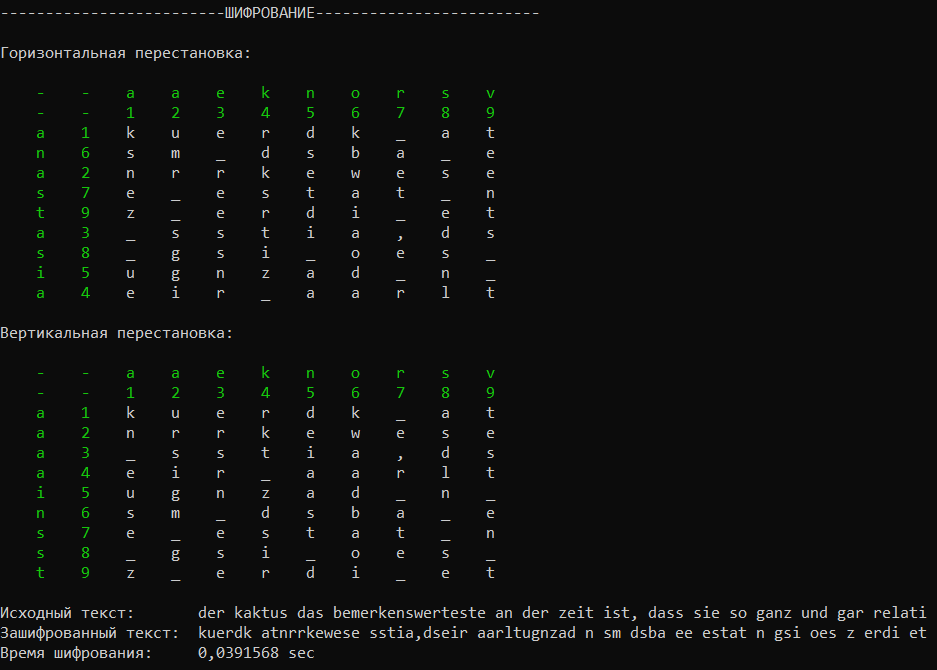


Рис. 6 – Реализация шифрования/расшифрования для метода

множественной перестановки

Графики:



Рис. 7 – Частота появления символов при использовании

маршрутной перестановки змейкой



Рис. 8 – Частота появления символов при использовании

множественной перестановки

В таблице 1 представлена зависимость времени от количества символов в сообщении для методов множественной и маршрутной перестановок.

Табл. 1 – Зависимость времени от кол-ва символов

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Шифрование | Расшифрование | Кол-во символов |
| Множественная перестановка | 0,1134693 | 0,2048861 | 81 |
| 0,0649794 | 0,1167977 | 81 |
| 0,1534709 | 0,2510501 | 81 |
| Маршрутная перестановка змейкой | 0,0054441 | 0,0145458 | 5625 |
| 0,0010996 | 0,0057915 | 2500 |
| 0,0012183 | 0,0069641 | 729 |

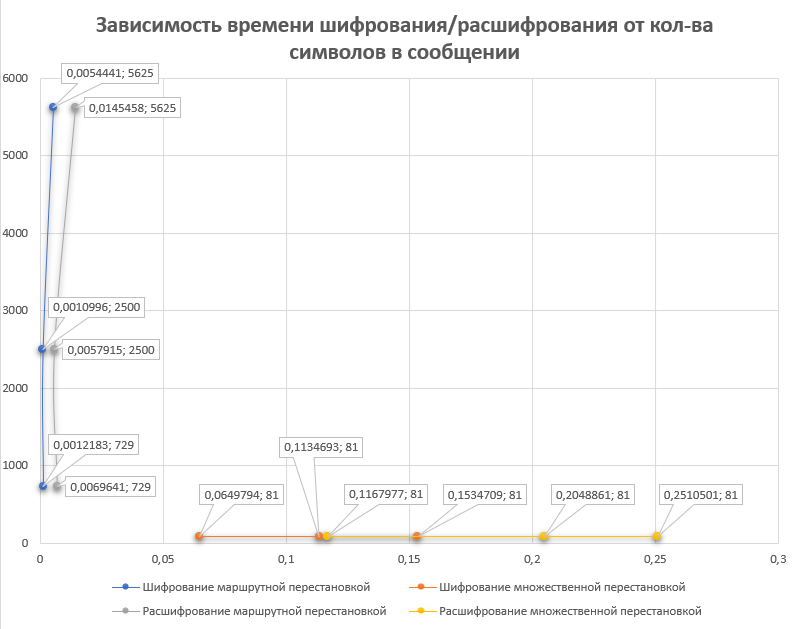


Рис. 8 – Зависимость времени от количества символов в сообщении

**Вывод**: в отличии от шифрования методом подстановки символов, шифрование методом перестановки сохраняет частотность появления символов, т. к. символы входящие в исходное сообщение лишь изменяют свои позиции, в то время как в методах подстановки они заменяются на другие.