МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Отчет по лабораторной работе №3**

**ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ ШИФРОВ НА ОСНОВЕ ПЕРЕСТАНОВКИ СИМВОЛОВ**

Выполнил:

Cтудент 3 курса 1 группы

Парибок И. А.

Вариант 5

Минск 2023

**Цель**: изучение и приобретение практических навыков разработки и использования приложений для реализации перестановочных шифров

**Практическое задание:**

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

• выполнять зашифрование/расшифрование текстовых документов (объемом не менее 500 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 в таблица 1);

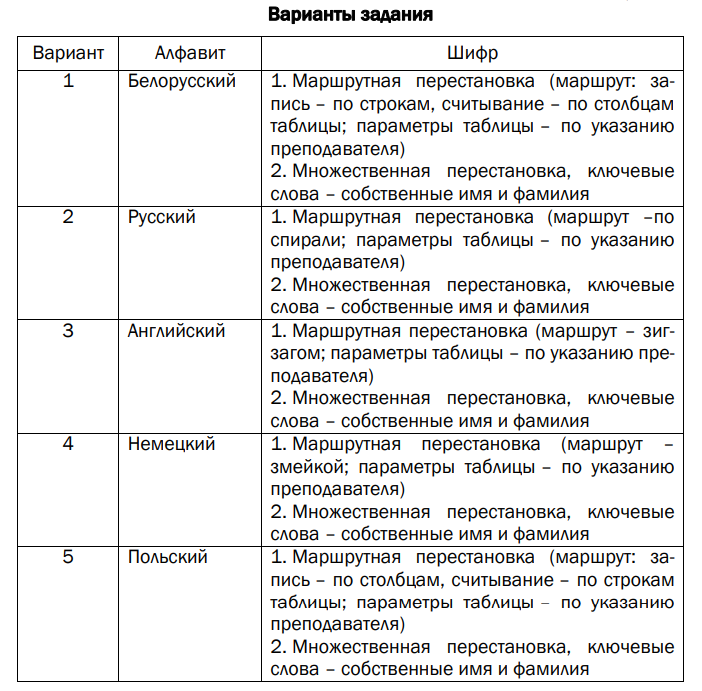


Таблица 1 – Варианты заданий

•формировать гистограммы частот появления символов для исходного и зашифрованного сообщений;

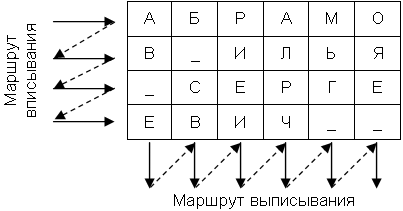
•оценивать время выполнения операций зашифрования/расшифрования (напоминание: во многих языках программирования есть встроенные методы для замеров времени; при отсутствии такового в используемом языке можно воспользоваться разностью двух дат (например, в миллисекундах: время после выполнения программы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

При анализе полученных гистограмм можно сопоставить полученные данные с аналогичными результатами выполнения лабораторной работы № 2 из [2] и лабораторной работы № 2 настоящего пособия. Если указанный в таблице язык исходного текста не известен разработчику программного средства, можно взять документ на требуемом языке и воспользоваться доступным электронным переводчиком (возникающие при этом отдельные семантические неточности не следует считать существенным недостатком выполняемого анализа).

3. Результаты оформить в виде отчета по установленным правилам.

**Выполнение работы:**

Были реализованы маршрутной перестановка (маршрут: запись – по столбцам, считывание – по строкам таблицы) и множественной перестановки с ключевыми словами «ilya» и «paribok» с использованием польского языка. На рисунке 1 изображен маршрут шифрования и дешифрования на основе маршрутной перестановки, запись по столбцам и чтение по строкам. На рисунках 2-3 представлены результаты выполнения приложения, а именно - шифрования и дешифрования текста на основе маршрутной перестановки. На рисунке 4 показана скорость выполнения программного кода шифрования и дешифрования. Исходя из полученных данных, можно сделать вывод, что в данной реализации дешифрование полученных данных более затратно по времени, чем шифрование. Гистограмма использования символов, представленная на рисунке 5, одинакова для сообщений до и после шифрования, так алгоритмы основаны на перестановке символов, и в отличии от подстановочных шифров символы просто перемещаются с старых позиций на новые, что не влияет на частотную гистограмму.



Запись сoобщения

Считывание

сoобщения

Рисунок 1 – Маршрут шифрования и дешифрования на основе маршрутной перестановки

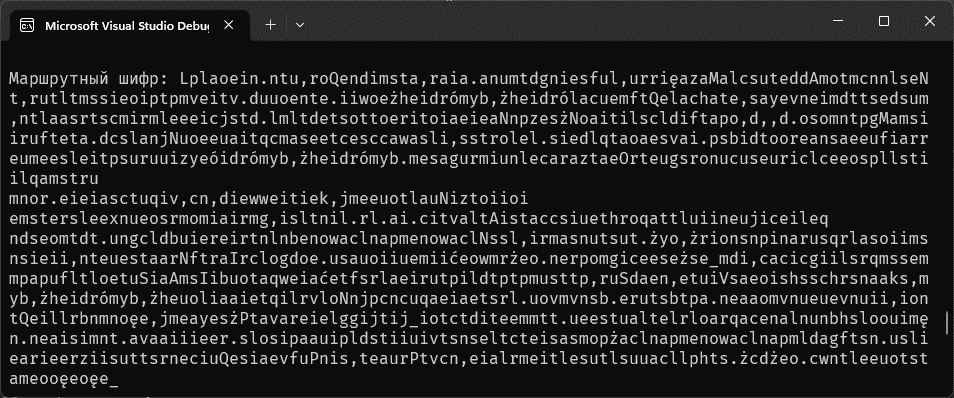


Рисунок 2 – Результат шифрования текста на основе маршрутной перестановки

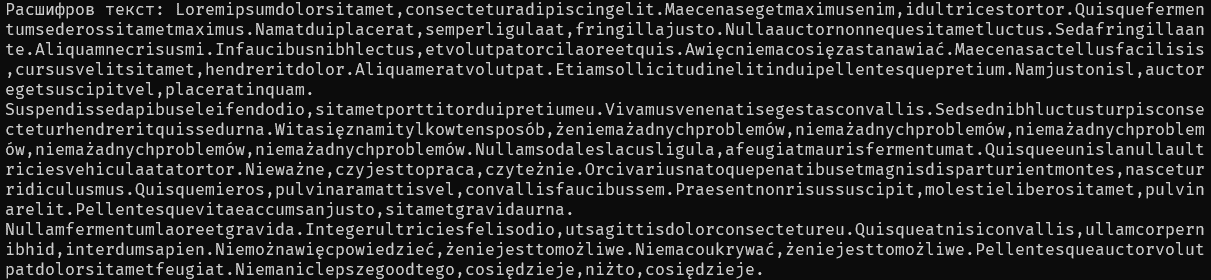


Рисунок 3 – Результат дешифрования текста на основе маршрутной перестановки



Рисунок 4 – Скорость выполнения шифрования и дешифрования текста на основе маршрутной перестановки

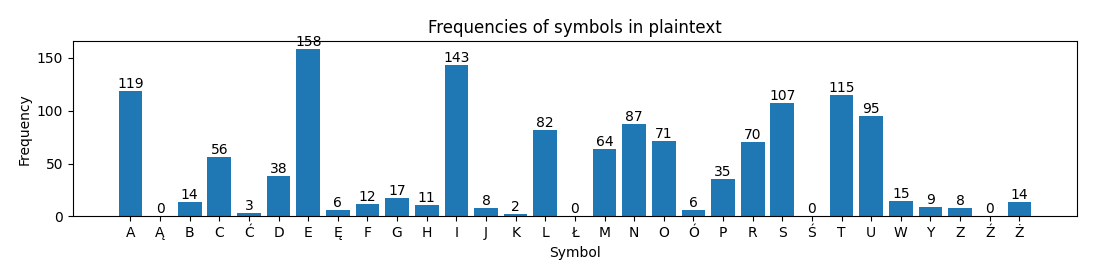


Рисунок 5 – Гистограмма использования символов сообщения

Листинг программы с использованием маршрутной перестановки изложен в приложение А. В качестве ключа использовалось число 6. В файле Text.txt хранится сообщение на польском языке

Шифр маршрутной перестановки - это метод шифрования, который использует перестановку символов в сообщении с помощью определенного ключа. Этот метод шифрования является одним из простых способов защиты конфиденциальной информации.

Для шифрования сообщения с помощью метода маршрутной перестановки необходимо выполнить следующие шаги:

1. Задайте ключ, который определяет количество столбцов, в которых будут расположены символы сообщения.
2. Разделите сообщение на равные части по количеству столбцов, указанных в ключе.
3. Запишите каждый столбец сообщения подряд в порядке, указанном в ключе.
4. Полученную строку символов можно дополнить случайными символами, чтобы получить сообщение фиксированной длины.

Для расшифровки сообщения, полученного методом маршрутной перестановки, необходимо выполнить обратные шаги.

На рисунках 6-7 представлены результаты выполнения приложения, а именно - шифрование и дешифрование с использованием множественной перестановки. В качестве ключа использовалось слова «ilya» и «paribok». На рисунке 8 показана скорость выполнения программного кода шифрования и дешифрования. Исходя из полученных данных, можно сделать вывод, что в данной реализации дешифрование полученных данных занимает больше времени, чем шифрование. Так как использовался тот же текст что и при использования маршрутной перестановки, то гистограмма на рисунке 5 применима и для данного случая.

Алгоритмы шифрования и дешифрования текста на основе множественной перестановки представлены в Приложение Б.

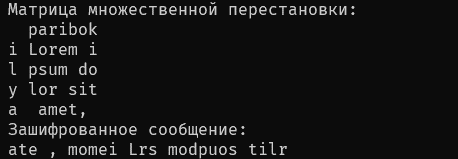


Рисунок 6 – Результат дешифрования текста на основе множественной перестановки

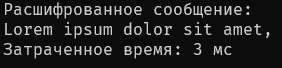


Рисунок 7 – Результат дешифрования текста на основе множественной перестановки



Рисунок 8 – Скорость выполнения шифрования и дешифрования текста на основе множественной перестановки

В ходе работы использовался шифр множественной перестановки с ключами «ilya» и «paribok». Шифр множественной перестановки - это метод шифрования, который основан на множественных перестановках символов в сообщении.

Алгоритм шифрования, следующий:

1. Задается ключ, который состоит из нескольких перестановок. Например, если ключ состоит из трех перестановок, то ключ может выглядеть следующим образом: (3 1 4 2), (1 4 3 2), (2 3 1 4).
2. Разбиваем исходное сообщение на блоки равной длины. Длина блока должна быть равна длине ключа.
3. Каждый блок переставляется согласно первой перестановке ключа, затем результат перестановки переставляется согласно второй перестановке, затем согласно третьей и так далее до последней перестановки.
4. Полученные переставленные блоки объединяются в зашифрованное сообщение.

Алгоритм расшифрования следующий:

1. Используем тот же ключ, который использовался для шифрования.
2. Разбиваем зашифрованное сообщение на блоки равной длины. Длина блока должна быть равна длине ключа.
3. Каждый блок переставляется в обратном порядке согласно последней перестановке ключа, затем результат перестановки переставляется в обратном порядке согласно предпоследней перестановке и так далее до первой перестановки.
4. Полученные переставленные блоки объединяются в исходное сообщение.

**Вывод:** в результате выполненной работы были реализованы два метода шифрования: на основе маршрутной перестановки и множественной перестановки. В качестве шифруемых данных использовался файл с более чем 1000 символами на польском языке. Также были получены данные о скорости работы шифрования.

1. В чем заключается основная идея криптографических преобразований на основе шифров перестановки?

Основная идея криптографических преобразований на основе шифров перестановки заключается в замене порядка символов в открытом тексте на новый порядок, заданный ключом шифрования. Это делает текст неразборчивым для противника, так как символы заменены на другие без изменения самих символов.

2. Привести классификационные признаки и дать сравнительную характеристику разновидностям перестановочных шифров.

Классификационные признаки перестановочных шифров:

Метод генерации ключа: ручной или автоматический

Метод перестановки: простая или сложная

Размерность ключа: фиксированная или переменная

Тип алфавита: латинский, кириллический или другой

Сравнительная характеристика разновидностей перестановочных шифров зависит от конкретного вида шифра, его принципа работы и используемых методов защиты.

3. Сколько разновидностей шифров, подобных шифру Цезаря, можно составить для алфавитов русского и белорусского языков?

Для алфавита русского языка можно составить 32 разновидности шифров, а для белорусского языка - 30 разновидностей шифров.

4. Охарактеризовать криптостойкость перестановочных и подстановочных шифров.

Перестановочные шифры в целом менее криптостойки, чем подстановочные шифры, так как они не изменяют сам символ, а только его позицию в тексте. Криптостойкость перестановочных шифров может быть усилена путем использования сложных методов перестановки, например, с помощью шифров Хилла или перестановок с использованием блоков.

5. Привести примеры и дать характеристику перестановочным шифрам, не рассмотренным в материалах к данной лабораторной работе.

Шифр Колосса - метод перестановки символов с использованием нескольких таблиц и комбинаций

Шифр Кардано - метод перестановки символов с помощью механического устройства

6. Имеются ли предпочтения в выборе размеров используемой таблицы для перестановочных шифров?

Размер таблицы для перестановочных шифров может быть любым, в зависимости от размера алфавита и желаемой степени защиты текста. Однако слишком большой размер таблицы может усложнить процесс шифрования и увеличить вероятность ошибки.

7. Охарактеризовать основные методы взлома перестановочных шифров

Основные методы взлома перестановочных шифров включают:

1. Метод частотного анализа: путем анализа частотности символов в шифре можно определить порядок символов и восстановить открытый текст.
2. Метод индекса совпадений: основывается на том, что вероятность совпадения символов в тексте зависит от его длины и структуры. Используется для определения периода шифра.
3. Метод криба: метод, основанный на знании небольшой части открытого текста, что позволяет сократить количество возможных перестановок.
4. Метод перебора: путем перебора всех возможных вариантов перестановок можно найти правильный ключ.
5. Комбинированные методы: сочетают в себе несколько методов для повышения эффективности взлома шифра.

Для повышения устойчивости перестановочных шифров к взлому необходимо использовать дополнительные методы защиты, такие как добавление случайных символов в текст, использование нескольких таблиц перестановок или использование сложных методов перестановки, например, с помощью шифров Хилла.

**Приложение А**

using System.Text;

namespace lab3\_1

{

class Program

{

static void Main(string[] args)

{

RouteCipher route = new RouteCipher();

Console.OutputEncoding = Encoding.Unicode;

//encrypt

const int key = 6;

string msg = "";

long oldTicks = DateTime.Now.Ticks;

using (StreamReader sr = new StreamReader("Text.txt"))

{

msg = (sr.ReadToEnd());

msg = msg.Replace(" ", "");

}

Console.WriteLine("Длинна сообщения:" + msg.Length);

string resultRoute = route.Encrypt(msg, key);

long timeCipher = (DateTime.Now.Ticks - oldTicks) / 1000;

Console.WriteLine("\nИсходный текст: " + msg);

Console.WriteLine("Маршрутный шифр: " + resultRoute);

Console.WriteLine("Расшифров текст: " + route.Decrypt(resultRoute, key));

long timeUnCipher = ((DateTime.Now.Ticks - oldTicks ) / 1000)-timeCipher;

Console.WriteLine("Затрачено на шифрование " + timeCipher + " мс\n");

Console.WriteLine("Затрачено на дешифрование " + timeUnCipher + " мс\n");}}}

class RouteCipher

{

static readonly char[] Alphabet = {'a','ą','b','c','ć','d','e','ę','f','g','h','i','j','k',

'l','ł','m','n','ń','o','ó','p','r','s','ś','t','u','w',

'y','z','ź','ż'};

static readonly int N = Alphabet.Length;

int[] colFx1 = new int[N];

int[] colFx2 = new int[N];

public string Encrypt(string msg, int key)

{

int len = msg.Length;

//кол-во вхождений

for (int i = 0; i < N; i++) //каждая буква алфавита провер. кол-во вхожд во фразе

{

colFx1[i] = msg.Count(el => el == Alphabet[i]);

}

if (len % key != 0)

{

for (int i = 0; i < (key - (len % key)); i++) //каждая буква алфавита провер. кол-во вхожд во фразе

{

msg += '\_';

}

}

string result = string.Empty;

string[] msgInArray = new string[(msg.Length / key) + 1];

for (int i = 0; i < (msg.Length / key); i++)

{

msgInArray[i] = msg.Substring(i \* key, key );

Console.WriteLine("msgInArray[" + i + "] = " + msgInArray[i]);

}

for (int i = 0; i < key; i++)

{

for (int k = 0; k < msgInArray.Length - 1; k++)

{

result += msgInArray[k].Substring(i, 1);

}

}

return result;

}

public string Decrypt(string msg, int key)

{

int len = msg.Length;

//кол-во вхождений

for (int i = 0; i < N; i++) //каждая буква алфавита провер. кол-во вхожд во фразе

{

colFx2[i] = msg.Count(el => el == Alphabet[i]);

}

string result = string.Empty;

string[] msgInArray = new string[(len / key)];

for (int i = 0; i < key; i++)

{

msgInArray[i] = msg.Substring(i \* (len / key), (len / key));

Console.WriteLine("msgInArray[" + i + "] = " + msgInArray[i]);

}

for (int i = 0; i < (len / key); i++)

{

for (int k = 0; k < key; k++)

{

if (msgInArray[k].Length <= i) continue;

result += msgInArray[k].Substring(i, 1);

}

}

//for (int i = 0; i < N; i++) // вывод кол-ва вхождений

//{

// Console.WriteLine(colFx2[i] + " --- " + Alphabet[i]);

//}

return result;

}

}}

**Приложение Б**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.IO;

using System.Linq;

using System.Text;

namespace lab3

{

internal static class Program

{

private const ushort NumberColumns = 8;

private const string Surname = "ilya";

private const string Name = "paribok";

private const string Alfabet = "aąbcćdeęfghijklłmnoóprssśtuwyzźż";

private static void Main(string[] args)

{

Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);

StreamReader sr = new StreamReader("Text.txt", Encoding.GetEncoding(1251));

string open = sr.ReadLine();

Console.WriteLine(open);

var message = open;

var numberRows = (ushort)Math.Ceiling((float)message.Length / NumberColumns);

var textUsage = SymbolsUsage(message);

var amountSymbols = textUsage.Select((c) => c.Value).Sum();

Console.WriteLine("Частота каждого символа:");

foreach (var (ch, amount) in textUsage)

Console.WriteLine($"{ch} - {amount} ({amount \* 100 / amountSymbols:N1}%)");

var watch = new Stopwatch();

var multPurmutationMatrix = new char[Surname.Length, Name.Length];

Console.WriteLine($"Матрица множественной перестановки:\n {Name}");

for (var i = 0; i < Surname.Length; i++)

{

Console.Write($"{Surname[i]} ");

for (var j = 0; j < Name.Length; j++)

{

multPurmutationMatrix[i, j] = message[i \* Name.Length + j];

Console.Write(multPurmutationMatrix[i, j]);

}

Console.WriteLine();

}

var finalPermutationMatrix = PermutateColumns(PermutateRows(multPurmutationMatrix));

var encrPermutationMessage = new StringBuilder();

for (var i = 0; i < Surname.Length; i++)

for (var j = 0; j < Name.Length; j++)

encrPermutationMessage.Append(finalPermutationMatrix[i, j]);

watch.Stop();

Console.WriteLine($"Зашифрованное сообщение:\n{encrPermutationMessage}\n" +

$"Затраченное время шифрования: {watch.ElapsedMilliseconds} мс");

var textUsageEncrMultPermut = SymbolsUsage(message);

var amountSymbolsMultPermut = textUsageEncrMultPermut.Select((c) => c.Value).Sum();

Console.WriteLine("Частота каждого символа:");

foreach (var (ch, amount) in textUsageEncrMultPermut)

Console.WriteLine($"{ch} - {amount} ({amount \* 100 / amountSymbolsMultPermut:N1}%)");

watch.Restart();

var decrPermutationMatrix = new char[Surname.Length, Name.Length];

for (var i = 0; i < Surname.Length; i++)

for (var j = 0; j < Name.Length; j++)

decrPermutationMatrix[i, j] = encrPermutationMessage[i \* Name.Length + j];

var finalDecrPermutationMatrix = DepermutateRows(DepermutateColumns(decrPermutationMatrix));

var decrPermutationMessage = new StringBuilder();

for (var i = 0; i < Surname.Length; i++)

for (var j = 0; j < Name.Length; j++)

decrPermutationMessage.Append(finalDecrPermutationMatrix[i, j]);

watch.Stop();

Console.WriteLine($"Расшифрованное сообщение:\n{decrPermutationMessage}\n" +

$"Затраченное время дешифрования: {watch.ElapsedMilliseconds} мс");

}

private static Dictionary<char, int> SymbolsUsage(string text)

{

var usage = new Dictionary<char, int>();

foreach (var ch in text.ToLower().Where(c => Alfabet.Contains(c)))

if (usage.ContainsKey(ch))

usage[ch]++;

else

usage.Add(ch, 1);

return usage;

}

private static char[,] DepermutateRows(char[,] permutationMatrix)

{

var baseCode = Surname;

var permutedMessage = baseCode.OrderBy(c => c).ToArray();

var changedPermutationMatrix = new char[Surname.Length, Name.Length];

for (var i = 0; i < permutedMessage.Length; i++)

{

var index = baseCode.IndexOf(permutedMessage[i]);

baseCode = baseCode.Remove(index, 1);

baseCode = baseCode.Insert(index, " ");

for (var j = 0; j < Name.Length; j++)

changedPermutationMatrix[index, j] = permutationMatrix[i, j];

}

return changedPermutationMatrix;

}

private static char[,] DepermutateColumns(char[,] permutationMatrix)

{

var baseCode = Name;

var permutedMessage = baseCode.OrderBy(c => c).ToArray();

var changedPermutationMatrix = new char[Surname.Length, Name.Length];

for (var i = 0; i < permutedMessage.Length; i++)

{

var index = baseCode.IndexOf(permutedMessage[i]);

baseCode = baseCode.Remove(index, 1);

baseCode = baseCode.Insert(index, " ");

for (var j = 0; j < Surname.Length; j++)

changedPermutationMatrix[j, index] = permutationMatrix[j, i];

}

return changedPermutationMatrix;

}

private static char[,] PermutateRows(char[,] permutationMatrix)

{

var rowsInEncryption = Surname.Select((c, i) => (c, i))

.ToLookup(

tuple => tuple.c,

tuple => Enumerable.Range(0, Name.Length)

.Select(x => permutationMatrix[tuple.i, x]).ToArray())

.OrderBy(gr => gr.Key);

var changedPermutationMatrix = new char[Surname.Length, Name.Length];

{

var i = 0;

foreach (var row in rowsInEncryption)

foreach (var arr in row)

{

for (var j = 0; j < arr.Length; j++)

changedPermutationMatrix[i, j] = arr[j];

i++;

}

}

return changedPermutationMatrix;

}

private static char[,] PermutateColumns(char[,] permutationMatrix)

{

var columnsInEncryption = Name.Select((c, i) => (c, i))

.ToLookup(

tuple => tuple.c,

tuple => Enumerable.Range(0, Surname.Length)

.Select(x => permutationMatrix[x, tuple.i]).ToArray())

.OrderBy(gr => gr.Key);

var changedPermutationMatrix = new char[Surname.Length, Name.Length];

{

var i = 0;

foreach (var row in columnsInEncryption)

foreach (var arr in row)

{

for (var j = 0; j < arr.Length; j++)

changedPermutationMatrix[j, i] = arr[j];

i++;

}

}

return changedPermutationMatrix;

}

}

}