Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет по лабораторной работе №5

«Исследование криптографических шифров на основе перестановки символов»

Студентка: Пунько А.А,

ФИТ 3 курс 5 группа

Преподаватель: Берников В. О.

Минск 2020

**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.

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Сущность перестановочного шифрования состоит в том, что, исходный текст (М) и зашифрованный текст (С) основаны на использовании одного и того же алфавита, а тайной или ключевой информацией является алгоритм перестановки. Шифры перестановки относятся к классу симметричных. Элементами текста могут быть отдельные символы (самый распространённый случай), пары, тройки букв и так далее. Классическими примерами перестановочных шифров являются анаграммы. Анаграмма (от греч. ανα – «снова» и γράμμα – «запись») – литературный приём, состоящий в перестановке букв (или звуков), что в результате дает другое слово или словосочетание, например: проездной–подрезной, листовка– вокалист, апельсин–спаниель.

В классической криптографии шифры перестановки делятся на два подкласса: • шифры простой или одинарной перестановки – при зашифровании символы открытого текста Мi перемещаются с исходных позиций в новые (в шифртексте Сi) один раз, • шифры сложной или множественной перестановки – при зашифровании символы открытого текста Мi перемещаются с исходных позиций в новые (в шифртексте Сi) несколько раз.

Среди шифров рассматриваемого подкласса иногда выделяют шифры простой перестановки (или перестановки без ключа). Символы открытого текста Мi перемешиваются по каким-либо правилам. Формально каждое из таких правил может рассматриваться в качестве ключа. Пример 1. Простейшим примером является запись открытого текста в обратной последовательности. Так, если Мi = «шифр перестановки», то Сi = «иквонатсереп рфиш». Если переставляются в соответствующем порядке пары букв, то Сi = «киованстрепе фрши». При более длинных сообщениях можно таким же образом перемещать целые слова или блоки слов. Подобную перестановку можно трактовать как транспозицию. В общем случае для использования шифров одинарной перестановки используется таблица, состоящая из двух строк: в первой строке записываются буквы, во второй – цифры J. Строки состоят из n столбцов. Буквы составляют шифруемое сообщение. Цифры J = j1, j2, …, jn, где j1 – номер позиции в зашифрованном сообщении первого символа открытого текста, где j2 – номер позиции в зашифрованном сообщении второго символа открытого текста и т. д. Таким образом, порядок следования цифр определяется используемым правилом (ключом) перестановки символов открытого текста для получения шифрограммы.

Указанные шифры строятся по тем же правилам, что и шифры простой перестановки. Блок должен состоять из 2-х или более символов. Если общее число таких символов в сообщении не кратно длине сообщения, то последний блок можно дополнить произвольными знаками.

Основой современных шифров рассматриваемого типа является геометрическая фигура. Обычно прямоугольник или прямоугольная матрица. В ячейки этой фигуры по определенному маршруту (слева-направо, сверху-вниз или каким-либо иным образом) записывается открытый текст. Для получения

шифрограммы нужно записать символы этого сообщения в иной последовательности, т.е. по иному маршруту (см. аналогию с методами перемежения/деперемежения данных в лабораторной работе №7 [1]).

Маршруты могут быть значительно более изощренными. Например, обход конем шахматной доски таким образом, чтобы в каждой клетке конь побывал один раз. Один из таких маршрутов был найден Л. Эйлером в 1759 г. Для примера на рис. 3.3 показан такой маршрут для обхода таблицы размером 5 х 4. Не менее занимательным и не менее сложным является организация маршрутов на основе «магических квадратов» – квадратных матриц со вписанными в каждую клетку неповторяющимися последовательными числами от 1, сумма которых по каждому столбцу, каждой строке и каждой диагонали дает одно и то же число. Создание новых оригинальных маршрутов приветствуется и поощряется при выполнении данной лабораторной работы.

Шифр вертикальной перестановки является разновидностью шифра маршрутной перестановки. К особенностям вертикального шифра можно отнести следующие: - количество столбцов в таблице фиксируется и определяется длиной ключа; - маршрут вписывания: слева-направо, сверху-вниз; - шифрограмма выписывается по столбцам в соответствии с их нумерацией (ключом).

Ключ может задавиться в виде текста (слова или словосочетания). Лексикографическое местоположение символов в ключевом выражении определяет порядок считывания столбцов.

Особенностью шифров множественной перестановки является минимум двукратная перестановка символов шифруемого сообщения. В простейшем случае это может задаваться перемешиваем не только столбцов, но и строк. Таким образом, этот случай соответствует использованию двух основных ключей: длина одного из них равна числу столбцов, другого – числу строк. К ключевой информацию мы можем относить также способы вписывания сообщения и считывания отдельных символов из текущего столбца матрицы.

Шифры гаммирования рассматриваются как самостоятельный класс. Такие шифры схожи с перестановочными тем, что в обоих случаях можно использовать табличное представление выполняемых операций на основе ключей. Вместе с тем, шифры гаммирования имеют много общего с подстановочными шифрами, поскольку на самом деле при зашифровании происходит подмена одних символов на другие.

**Ход работы**

В соответствии с заданием было необходимо разработать алгоритмы маршрутной перестановки бля белорусского языка (маршрут: запись – по строкам, считывание – по столбцам таблицы; параметры таблицы – по указанию преподавателя), а также алгоритм множественной перестановки с ключевыми словами именем и фамилией.

Класс, реализующий алгоритм маршрутной перестановки, приведен в листинге 1.

class RouteEncryption

{

private int s;

private int k;

char[,] table;

public void SetKey(int countString, int lengthMessage)

{

s = countString;

k = (lengthMessage - 1) / countString + 1;

}

public string Encrypt(string input)

{

var map1 = new Dictionary<char, int>();

foreach (char c in input)

{

if (!map1.ContainsKey(c))

map1.Add(c, 1);

else

map1[c] += 1;

}

int len1 = input.Length;

var orderkey1 = from i in map1 orderby i.Key select i;

foreach (var item in orderkey1)

{

using (StreamWriter sw1 = new StreamWriter("info1.log", true, Encoding.Default))

{

sw1.Write($"{DateTime.Now} Количество символов {item.Key} = {item.Value} частота {(double)item.Value / len1}\n");

}

}

int l = 0;

string result = "";

table = new char[k, s];

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

{

for (int j = 0; j < s; j++)

{

if (l < input.Length)

{

table[i, j] = input[l];

l++;

}

else

{

table[i, j] = ' ';

}

}

}

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

{

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

{

result += table[j, i];

}

}

var map2 = new Dictionary<char, int>();

foreach (char c in result)

{

if (!map2.ContainsKey(c))

map2.Add(c, 1);

else

map2[c] += 1;

}

int len2 = result.Length;

var orderkey2 = from i in map2 orderby i.Key select i;

foreach (var item in orderkey2)

{

using (StreamWriter sw2 = new StreamWriter("info2.log", true, Encoding.Default))

{

sw2.Write($"{DateTime.Now} Количество символов {item.Key} = {item.Value} частота {(double)item.Value / len2}\n");

}

}

return result;

}

public string Decrypt(string output)

{

int p = 0;

string result = "";

table = new char[k, s];

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

{

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

{

if (p < output.Length)

{

table[j, i] = output[p];

p++;

}

else

{

table[j, i] = ' ';

}

}

}

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

{

for (int j = 0; j < s; j++)

{

result += table[i, j];

}

}

return result;

}

}

Листинг1 – Реализация алгоритма маршрутной перстановки

Класс, реализующий алгоритм множественной перестановки, представлен в листинге 2.

class ManyEncryption

{

private int[] key = null;

private int[] key2 = null;

char[,,] table;

public void SetKey(string \_key, string \_key2)

{

Dictionary<int, char> matrix = new Dictionary<int, char>();

int[] codedStr = new int[\_key.Length];

char[] alphabet = "абвгдеёжзiйклмнопрстуўфхцчшыьэюя".ToCharArray();

int counter = 1;

bool found = false;

foreach (char с in alphabet)

{

for (int i = 0; i < \_key.Length; i++)

{

if (\_key[i] == с)

{

codedStr[i] = counter;

if (matrix.ContainsKey(counter) == false)

{

matrix.Add(counter, \_key[i]);

}

found = true;

}

}

if (found)

{

counter++;

found = false;

}

}

counter = 1;

codedStr = new int[\_key2.Length];

found = false;

key = codedStr;

foreach (char с in alphabet)

{

for (int i = 0; i < \_key2.Length; i++)

{

if (\_key2[i] == с)

{

codedStr[i] = counter;

if (matrix.ContainsKey(counter) == false)

{

matrix.Add(counter, \_key2[i]);

}

found = true;

}

}

if (found)

{

counter++;

found = false;

}

}

key2 = codedStr;

}

public string Encrypt(string input)

{

string result = "";

int length = 0;

int countTable = input.Length / (key.Length \* key2.Length) + 1;

table = new char[countTable, key.Length, key2.Length];

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

{

for (int j = 0; j < key.Length; j++)

{

for (int k = 0; k < key2.Length; k++)

{

if (length < input.Length)

{

table[i, j, k] = input[length];

length++;

}

else

{

table[i, j, k] = ' ';

}

}

}

}

char temp;

int index, index2 = 0,

number = 1, temp2;

int[] massusingkeys = new int[key2.Length];

for (int f = 0; f < key2.Length; f++)

{

massusingkeys[f] = 999;

}

for (int i = 0; i < key2.Length; i++)

{

index = Array.IndexOf(key2, number);

for (int j = 0; j < countTable; j++)

{

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

{

if (Array.IndexOf(massusingkeys, index) == -1)

{

temp = table[j, k, index];

table[j, k, index] = table[j, k, index2];

table[j, k, index2] = temp;

}

}

if (Array.IndexOf(massusingkeys, index) == -1)

{

temp2 = key2[index];

key2[index] = key2[index2];

key2[index2] = temp2;

}

}

massusingkeys[i] = number - 1;

number = number + 1;

index2++;

}

index2 = 0;

number = 1;

massusingkeys = new int[key2.Length];

for (int f = 0; f < key2.Length; f++)

{

massusingkeys[f] = 999;

}

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

{

index = Array.IndexOf(key, Convert.ToInt32(number));

for (int j = 0; j < countTable; j++)

{

for (int k = 0; k < key2.Length; k++)

{

if (Array.IndexOf(massusingkeys, index) == -1)

{

temp = table[j, index, k];

table[j, index, k] = table[j, index2, k];

table[j, index2, k] = temp;

}

}

if (Array.IndexOf(massusingkeys, index) == -1)

{

temp2 = key[index];

key[index] = key[index2];

key[index2] = temp2;

}

}

massusingkeys[i] = number - 1;

number = number + 1;

index2++;

}

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

{

for (int j = 0; j < key.Length; j++)

{

for (int k = 0; k < key2.Length; k++)

{

result += table[i, j, k];

}

}

}

return result;

}

public string Decrypt(string output)

{

string result = "";

int length = 0;

int countTable = (output.Length - 1) / (key.Length \* key2.Length) + 1;

table = new char[countTable, key.Length, key2.Length];

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

{

for (int j = 0; j < key.Length; j++)

{

for (int k = 0; k < key2.Length; k++)

{

if (length < output.Length)

{

table[i, j, k] = output[length];

length++;

}

else

{

table[i, j, k] = ' ';

}

}

}

}

char temp;

int index, index2 = 0,

number = 1;

int[] massusingkeys = new int[key2.Length];

for (int f = 0; f < key2.Length; f++)

{

massusingkeys[f] = 999;

}

for (int i = 0; i < key2.Length; i++)

{

index = Array.IndexOf(key2, number);

for (int j = 0; j < countTable; j++)

{

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

{

if (Array.IndexOf(massusingkeys, index2) == -1)

{

temp = table[j, k, index];

table[j, k, index] = table[j, k, index2];

table[j, k, index2] = temp;

}

}

}

massusingkeys[i] = index;

number = number + 1;

index2++;

}

index2 = 0;

number = 1;

massusingkeys = new int[key2.Length];

for (int f = 0; f < key2.Length; f++)

{

massusingkeys[f] = 999;

}

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

{

index = Array.IndexOf(key, Convert.ToInt32(number));

for (int j = 0; j < countTable; j++)

{

for (int k = 0; k < key2.Length; k++)

{

if (Array.IndexOf(massusingkeys, index2) == -1)

{

temp = table[j, index, k];

table[j, index, k] = table[j, index2, k];

table[j, index2, k] = temp;

}

}

}

massusingkeys[i] = index;

number = number + 1;

index2++;

}

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

{

for (int j = 0; j < key.Length; j++)

{

for (int k = 0; k < key2.Length; k++)

{

result += table[i, j, k];

}

}

}

return result;

}

}

Листинг1 – Реализация алгоритма множественной перестановки