Учреждение образования

«Белорусский государственный технологический университет»

**Отчет по лабораторной работе №4**

по дисциплине: «Защита информации и надежность информационных систем»

Исполнитель:

Воликов Д. А., ФИТ 4-7

Руководитель:

Асс. Савельева М. Г.

Минск 2024

# Лабораторная работа №4. Изучение устройства и функциональных особенностей шифровальной машины «энигма»

**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. Изучить структуру, принципы функционирования, реализацию процедур зашифрования сообщений в машинах семейства «Энигма».
3. Изучить и приобрести практические навыки выполнения криптопреобразований информации на платформе «Энигма», реализованной в виде симуляторов.
4. Получить практические навыки оценки криптостойкости подстановочных и перестановочных шифров на платформе «Энигма».
5. Результаты выполнения лабораторной работы оформить в виде отчета о проведенных исследованиях, методике выполнения практической части задания и оценке криптостойкости шифров.

## Теоретические сведения

Идея создания шифровального устройства высказана голландцем Гуго Кох де Дельфтю (в некоторых источниках – Гуго Александр Кох, Hugo Alexander Koch) еще в 1919 г. В 1920 г. он же изобрел первую роторную шифровальную машинку. Параллельно с этим немец Артур Шербиус (Arthur Scherbius) изучал проблему криптостойкости (в нашем современном понимании) шифровальных машин. Он же получил патент на такую машину, которую назвали «Энигма». Основная особенность «Энигмы» – все знали в то время алгоритм шифрования, но никто не мог подобрать нужный ключ.

Первая шифровальная машина, «Enigma A», появилась на рынке в 1923 г. Это была большая и тяжелая машина со встроенной пишущей машинкой и весом около 50 кг. Вскоре после этого была представлена «Enigma B», очень похожая на «Enigma A». Вес и размеры этих машин сделали их непривлекательными для использования в военных целях.

По достоинству шифровальную машину оценили в немецкой армии. В 1925 г. ее принял на вооружение сначала военно-морской флот (модель С), а в 1930 г. – Вермахт («Enigma I»). Общее количество шифраторов, произведенных до и во время Второй мировой войны, превысило 100 тысяч. Применялись они всеми видами вооруженных сил Германии, а также военной разведкой и службой безопасности.

Машина «Энигма» – это электромеханическое устройство. Как и другие роторные машины, «Энигма» состоит из комбинации механических и электрических подсистем.

Механическая часть включает в себя клавиатуру, набор вращающихся дисков – роторов, которые расположены вдоль вала и прилегают к нему, и ступенчатого механизма, двигающего один или несколько роторов при каждом нажатии на клавишу.

Электрическая часть, в свою очередь, состоит из электрической схемы, соединяющей между собой клавиатуру, коммутационную панель, лампочки и роторы (для соединения роторов использовались скользящие контакты).

Конкретный механизм мог быть разным, но общий принцип был таков: при каждом нажатии на клавишу самый правый ротор сдвигался на одну позицию, а при определенных условиях сдвигались и другие роторы. Движение роторов приводило к различным криптографическим преобразованиям при каждом следующем нажатии на клавишу на клавиатуре, т. е. зашифрование/расшифрование сообщений основано на выполнении ряда замен (подстановок) одного символа другим. Идея А. Шербиуса состояла в том, чтобы добиться этих подстановок электрическими связями.

Механические части двигались и, замыкая контакты, образовывали меняющийся электрический контур. При нажатии на клавишу клавиатуры контур замыкается, ток проходит через созданную (для зашифрования/расшифрования одного конкретного символа сообщения) цепь и в результате включает одну из набора лампочек, отображающую искомую букву шифртекста (или расшифрованного сообщения).

## 1.2 Практическое задание

**Вариант 2**. По условию в машине используются роторы II, III и V. Рефлектор типа B. С помощью разработанного приложения зашифровать сообщение, применив не менее 5 вариантов начальных установок роторов.

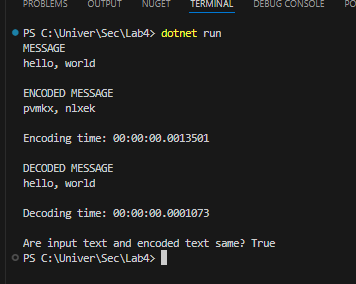


Рисунок 1 – Зашифрование/расшифрование сообщения с установленными значениями роторами 1-2-1

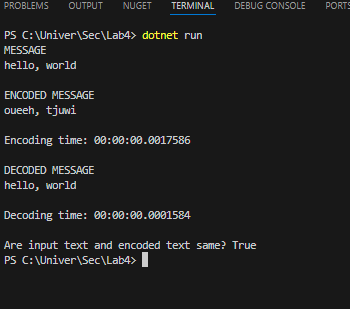


Рисунок 2 – Зашифрование/расшифрование сообщения с установленными значениями роторами 15-20-15

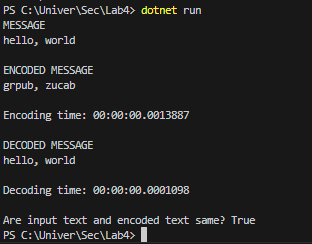


Рисунок 3 – Зашифрование/расшифрование сообщения с установленными значениями роторами 4-5-1

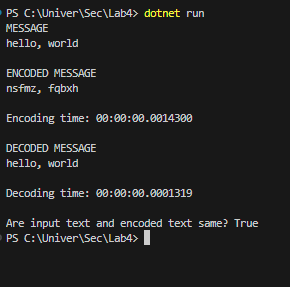


Рисунок 4 – Зашифрование/расшифрование сообщения с установленными значениями роторами 1-2-3

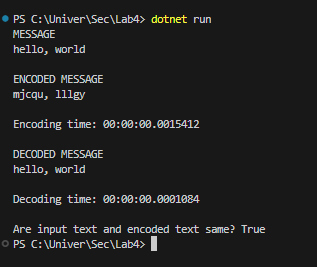


Рисунок 5 – Зашифрование/расшифрование сообщения с установленными значениями роторами 3-5-7

Как можно заметить, сообщение корректно зашифровывается и расшифровывается. Теперь оценим криптостойкость кода. Зная, что используется три ротора, соответственно комбинаций ключей будет

# Листинг «Лабораторная работа №3»

using System.Text;

static void CalculateCharsFrequency(ref Dictionary<char, int> alphabet, string message)

{

foreach (var ch in message) {

if (!alphabet.TryAdd(ch, 1)) {

alphabet[ch] += 1;

}

}

}

static void CharsFrequencyToCSV(ref readonly Dictionary<char, int> alphabet, string fileName)

{

int length = alphabet.Sum(item => item.Value);

File.WriteAllText(fileName, "Char\tFrequency\n");

foreach (var item in alphabet) {

string ch = item.Key == '\n' ? "\\n" :

item.Key == '\r' ? "\\r" :

item.Key == '\"' ? "\\\"" :

item.Key == '\'' ? "\\'" :

item.Key.ToString();

File.AppendAllText(fileName, ch + "\t" + (double)item.Value / length + '\n');

}

}

static int CalculateMatrixRows(int columns, int messageLength) {

return (int)Math.Ceiling((double)messageLength / columns);

}

static string SpiralPermutationEncode(string message, int columns)

{

if (columns < 5) {

throw new Exception("Spiral permutation encoding: Number of columns must be >= 5");

}

int rows = CalculateMatrixRows(columns, message.Length);

char[,] matrix = new char[rows, columns];

string formatMessage = message + new string('\*', columns \* rows - message.Length);

for (int i = 0; i < rows; i++) {

for (int j = 0; j < columns; j++) {

matrix[i, j] = formatMessage[i \* columns + j];

}

}

string encodedMessage = "";

// Вообще не понимаю как читать по спирали, пытался вывести закономерность, но ничего не понял

// Не знаю что здесь вообще происходит, взял готовый алгоритм с какого-то задрыпанного сайта по С

// https://www.haikson.com/programming/zapolnenie-dvumernoj-matritsyi-po-spirali/

int Ibeg = 0;

int Ifin = 0;

int Jbeg = 0;

int Jfin = 0;

int counter = 0;

int I = 0;

int J = 0;

while (counter < rows \* columns){

encodedMessage += formatMessage[I \* columns + J];

if (I == Ibeg && J < columns - Jfin - 1)

++J;

else if (J == columns - Jfin - 1 && I < rows - Ifin - 1)

++I;

else if (I == rows - Ifin - 1 && J > Jbeg)

--J;

else

--I;

if ((I == Ibeg + 1) && (J == Jbeg) && (Jbeg != columns - Jfin - 1)){

++Ibeg;

++Ifin;

++Jbeg;

++Jfin;

}

counter++;

}

// -------------------------------------------

return encodedMessage;

}

static string SpiralPermutationDecode(string message, int columns)

{

if (columns < 5) {

throw new Exception("Spiral permutation encoding: Number of columns must be >= 5");

}

int rows = CalculateMatrixRows(columns, message.Length);

char[,] matrix = new char[rows, columns];

int Ibeg = 0;

int Ifin = 0;

int Jbeg = 0;

int Jfin = 0;

int counter = 0;

int I = 0;

int J = 0;

while (counter < rows \* columns){

matrix[I, J] = message[counter];

if (I == Ibeg && J < columns - Jfin - 1)

++J;

else if (J == columns - Jfin - 1 && I < rows - Ifin - 1)

++I;

else if (I == rows - Ifin - 1 && J > Jbeg)

--J;

else

--I;

if ((I == Ibeg + 1) && (J == Jbeg) && (Jbeg != columns - Jfin - 1)){

++Ibeg;

++Ifin;

++Jbeg;

++Jfin;

}

counter++;

}

string decodedMessage = "";

for (int i = 0; i < rows; i++) {

for (int j = 0; j < columns; j++) {

decodedMessage += matrix[i, j].ToString();

}

}

return decodedMessage[..decodedMessage.IndexOf('\*')];

}

static string GenerateRandomMessage(int length)

{

string characters = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";

string message = "";

for (int i = 0; i < length; i++) {

int rand = new Random().Next() % characters.Length;

message += characters[rand];

}

return message;

}

static string MultiPermutationEncode(string message, string key1, string key2)

{

string encodedMessage = "";

var key1copy = new StringBuilder(key1);

var key2copy = new StringBuilder(key2);

char[][] matrix = new char[key1.Length][];

string sortedKey1 = string.Concat(key1.OrderBy(c => c));

string sortedKey2 = string.Concat(key2.OrderBy(c => c));

for (int i = 0; i < key1.Length; i++) {

matrix[i] = new char[key2.Length];

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

matrix[i][j] = message[i \* key2.Length + j];

}

}

char[][] matrixBuf = (char[][])matrix.Clone();

for (int i = 0; i < sortedKey1.Length; i++) {

int rowIndex = key1copy.ToString().IndexOf(sortedKey1[i]);

key1copy[rowIndex] = '\_';

matrix[i] = (char[])matrixBuf[rowIndex].Clone();

}

for (int i = 0; i < key1.Length; i++) {

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

matrixBuf[i][j] = matrix[i][j];

}

}

for (int i = 0; i < key1.Length; i++) {

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

int columnIndex = key2copy.ToString().IndexOf(sortedKey2[j]);

key2copy[columnIndex] = '\_';

matrix[i][j] = matrixBuf[i][columnIndex];

}

key2copy = new StringBuilder(key2);

}

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

for (int j = 0; j < key1.Length; j++) {

encodedMessage += matrix[j][i];

}

}

return encodedMessage;

}

static string MultiPermutationDecode(string message, string key1, string key2)

{

string decodedMessage = "";

char[][] matrix = new char[key1.Length][];

char[][] matrixBuf = new char[key1.Length][];

var sortedKey1 = new StringBuilder(string.Concat(key1.OrderBy(c => c)));

var sortedKey2 = new StringBuilder(string.Concat(key2.OrderBy(c => c)));

for (int i = 0; i < key1.Length; i++) {

matrix[i] = new char[key2.Length];

matrixBuf[i] = new char[key2.Length];

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

matrix[i][j] = message[j \* key1.Length + i];

matrixBuf[i][j] = message[j \* key1.Length + i];

}

}

for (int i = 0; i < key1.Length; i++) {

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

int columnIndex = sortedKey2.ToString().IndexOf(key2[j]);

sortedKey2[columnIndex] = '\_';

matrix[i][j] = matrixBuf[i][columnIndex];

}

sortedKey2 = new StringBuilder(string.Concat(key2.OrderBy(c => c)));

}

for (int i = 0; i < key1.Length; i++) {

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

matrixBuf[i][j] = matrix[i][j];

}

}

for (int i = 0; i < sortedKey1.Length; i++) {

int rowIndex = sortedKey1.ToString().IndexOf(key1[i]);

sortedKey1[rowIndex] = '\_';

matrix[i] = (char[])matrixBuf[rowIndex].Clone();

}

for (int i = 0; i < key1.Length; i++) {

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

decodedMessage += matrix[i][j];

}

}

return decodedMessage;

}

var fileContent = File.ReadAllText("file.txt").ToLower();

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

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

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

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

string csvFileContent = @"./file.csv";

string csvSpiral = @"./spiral.csv";

string csvMultiBefore = @"./multiBefore.csv";

string csvMultiAfter = @"./multiAfter.csv";

CalculateCharsFrequency(ref alphabet, fileContent);

CharsFrequencyToCSV(ref alphabet, csvFileContent);

int spiralColumns = 20;

var watch = System.Diagnostics.Stopwatch.StartNew();

var spiralEncodedMessage = SpiralPermutationEncode(fileContent, spiralColumns);

watch.Stop();

Console.WriteLine("Spiral encoding time, ms: " + watch.Elapsed);

watch = System.Diagnostics.Stopwatch.StartNew();

var spiralDecodedMessage = SpiralPermutationDecode(spiralEncodedMessage, spiralColumns);

watch.Stop();

Console.WriteLine("Spiral decoding time, ms: " + watch.Elapsed);

CalculateCharsFrequency(ref spiralAlphabet, spiralEncodedMessage);

CharsFrequencyToCSV(ref spiralAlphabet, csvSpiral);

Console.WriteLine($"Spiral decoded == file: {spiralDecodedMessage == fileContent}");

const string key1 = "воликов";

const string key2 = "дмитрийй";

string randomMessage = GenerateRandomMessage(key1.Length \* key2.Length);

CalculateCharsFrequency(ref multiAlphabetBefore, randomMessage);

CharsFrequencyToCSV(ref multiAlphabetBefore, csvMultiBefore);

Console.WriteLine();

Console.WriteLine("Random message for multi permutation: " + randomMessage);

watch = System.Diagnostics.Stopwatch.StartNew();

var multiEncodedMessage = MultiPermutationEncode(randomMessage, key1, key2);

watch.Stop();

Console.WriteLine("Multi encoding time, ms: " + watch.Elapsed);

Console.WriteLine("Encoded message: " + multiEncodedMessage);

CalculateCharsFrequency(ref multiAlphabetAfter, multiEncodedMessage);

CharsFrequencyToCSV(ref multiAlphabetAfter, csvMultiAfter);

watch = System.Diagnostics.Stopwatch.StartNew();

var multiDecodedMessage = MultiPermutationDecode(multiEncodedMessage, key1, key2);

watch.Stop();

Console.WriteLine("Multi decoding time, ms: " + watch.Elapsed);

Console.WriteLine("Decoded message: " + multiDecodedMessage);

Console.WriteLine($"Multi permutation decoded == random message: {multiDecodedMessage == randomMessage}");

Листинг 1 – Код программы на языке C#