**АЛГОРИТМ ШИФРОВАНИЯ ПЕРЕСТАНОВКОЙ**

**Вариант №3**

отчет о лабораторной работе №2

по дисциплине

*МЕТОДЫ И СРЕДСТВА КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ*

Выполнила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ст. гр. №230711, Павлова В.С.

Проверила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доцент каф. ИБ, Басалова Г.В.

**ХОД РАБОТЫ**

**Задание.** Реализовать программно алгоритм шифрования перестановкой по таблице размера 6х3 (6 столбцов на 3 строки). Ключ – последовательность чтения столбцов при кодировании – задаётся с клавиатуры.

**Листинг 1** – Метод Main (метод обработки данных)

static void Main(string[] args)

{

Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);

string path = @"C:\Users\Вика\Desktop\test.txt";

byte[] data = File.ReadAllBytes(path);

int rowCount = 3; //строки

int columnCount = 6; //столбцы

Console.Write("Write the key in 123456 format: ");

string key = Console.ReadLine(); //ключ

using (BinaryWriter encryptedOutput = new BinaryWriter(

new FileStream(@"C:\Users\Вика\Desktop\encrypted.txt", FileMode.Create), Encoding.UTF8))

{

using (BinaryWriter decryptedOutput = new BinaryWriter(

new FileStream(@"C:\Users\Вика\Desktop\decrypted.txt", FileMode.Create), Encoding.UTF8))

{ //количество таблиц

int tableCount = data.Length / (rowCount \* columnCount) + 1;

int counter = 0;

while (counter < tableCount)

{ //если таблица не последняя (т.е. полная)

if (counter < tableCount - 1)

{ //берём блок данных целиком

byte[] dataBlock = new byte[rowCount \* columnCount];

Array.Copy(data,

counter \* (rowCount \* columnCount),

dataBlock,

0,

rowCount \* columnCount);

byte[] cryptedBlock = EncryptData(dataBlock, columnCount, rowCount, key);

encryptedOutput.Write(cryptedBlock);

byte[] decryptedBlock = DecryptData(cryptedBlock, cryptedBlock.Length, columnCount, rowCount, key);

decryptedOutput.Write(decryptedBlock);

}

else

{ //запись оставшихся символов

List<byte> dataBlock = new List<byte>(rowCount \* columnCount);

for (int i = 0; i < data.Length % (rowCount \* columnCount); i++)

{

dataBlock.Add(data[counter \* (rowCount \* columnCount) + i]);

}

byte[] cryptedBlock = EncryptData(dataBlock.ToArray(), columnCount, rowCount, key);

encryptedOutput.Write(cryptedBlock);

byte[] decryptedBlock = DecryptData(cryptedBlock, cryptedBlock.Length, columnCount, rowCount, key);

decryptedOutput.Write(decryptedBlock);

}

counter++;

}

}

}

}

**Листинг 2** – Программная реализация алгоритма шифрования

public static byte[] EncryptData(byte[] data, int columnCount, int rowCount, string key)

{

Table table = new Table(rowCount, columnCount);

//число полных строк = длина/число столбцов

int fullRowCount = data.Length / columnCount;

//число символов в последней строке

int lastRowSymbolsCount = data.Length % columnCount;

int k = 0;

for (int i = 0; i < fullRowCount; i++) //заполняем все целые строки

{

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

{

table.SourceTable[i][j] = ((byte)data[k]);

k++;

}

}

for (int i = 0; i < lastRowSymbolsCount; i++) //заполянем последнюю строку

{

table.SourceTable[fullRowCount][i] = ((byte)data[k + i]);

}

//ШИФРОВАНИЕ

List<byte> cryptedData = new List<byte>(data.Length);

for (int i = 0; i < columnCount; i++) //проход по ключу

{ //индекс текущего столбца

int curColumnIndex = (int)Char.GetNumericValue(key[i]) - 1;

if (curColumnIndex > lastRowSymbolsCount - 1)

{ //если текущий столбец неполный

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

{

cryptedData.Add(table.SourceTable[j][curColumnIndex]);

}

}

else //текущий столбец полный

{

for (int j = 0; j < fullRowCount + 1; j++)

{

cryptedData.Add(table.SourceTable[j][curColumnIndex]);

}

}

}

return cryptedData.ToArray();

}

**Листинг 3** – Программная реализация алгоритма дешифрования

public static byte[] DecryptData(byte[] encryptedData, int dataLength, int columnCount, int rowCount, string key)

{ //ДЕШИФРОВАНИЕ

Table table = new Table(rowCount, columnCount);

//число полных строк = длина/число столбцов

int fullRowCount = dataLength / columnCount;

//число символов в последней строке

int lastRowSymbolsCount = dataLength % columnCount;

List<byte> decryptedData = new List<byte>(dataLength);

int k = 0;

for (int i = 0; i < columnCount; i++) //проход по ключу

{ //индекс текущего столбца

int curColumnIndex = (int)Char.GetNumericValue(key[i]) - 1;

//если текущий столбец неполный

if (curColumnIndex > lastRowSymbolsCount - 1)

{

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

{

table.SourceTable[j][curColumnIndex] = encryptedData[k];

k++;

}

}

else

{

for (int j = 0; j < fullRowCount + 1; j++)//текущий столбец полный

{

table.SourceTable[j][curColumnIndex] = encryptedData[k];

k++;

}

}

}

k = 0;

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

{

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

{

if (k < dataLength)

{

decryptedData.Add(table.SourceTable[i][j]);

k++;

}

else

return decryptedData.ToArray();

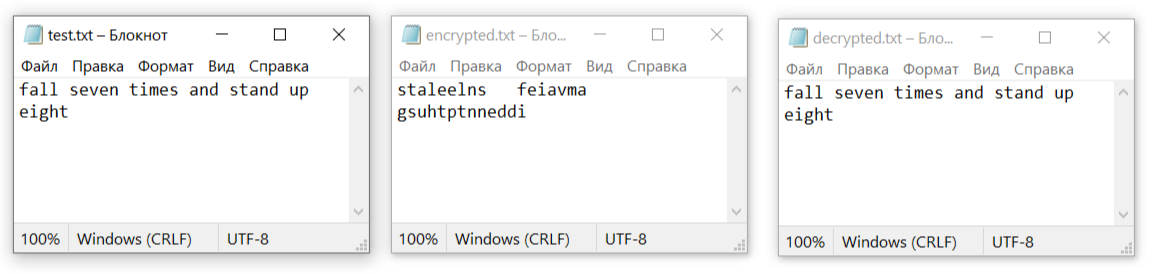
}

}

return decryptedData.ToArray();

}

**Демонстрационный пример №1.** Файл .txt, ключ *k* = 634512



**Демонстрационный пример №2.** Файл .jpg, ключ *k* = 152643

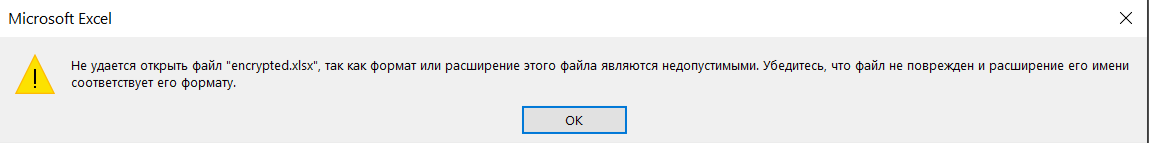


Представление изображения с помощью HEX Editor:

|  |  |
| --- | --- |
|  |  |

**Демонстрационный пример №3** Файл .xlsx, ключ *k* = 354621

Зашифрованный файл не открывается:



После расшифрования всё работает:

