Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и программной инженерии

Лабораторная работа №6

по дисциплине "Администрирование и безопасность информационных систем"

ТЕМА РАБОТЫ:

Использование гаммирования при шифровании

Выполнил:

студент гр. ПРИ-120

Парахин К.В.

Принял:

Доцент кафедры ИСПИ

Курочкин С.В.

Владимир 2023 г.

Цель работы:

Освоить на практике применение алгоритма шифрования методом гаммирования.

Выполнение работы:

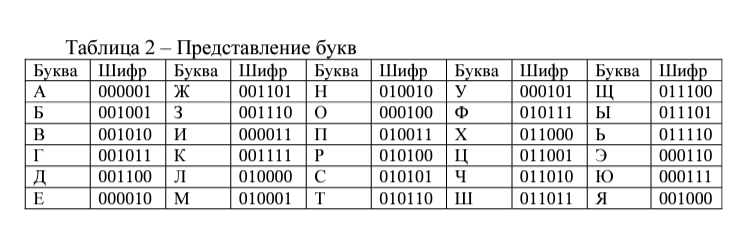
Задание.

Зашифруйте следующее предложение методом гаммирования.

«Помехоустойчивое кодирование – это кодирование с возможностью восстановления потерянных или ошибочно принятых данных.»

Знаки гаммы: 2 3 10 4 1 5 6 7 8 11 15 14 12 13 9 0.

Шифрование происходит в цикле (знак гаммы повторяется циклически).



Полученное решение:

Зашифрованное сообщение - МЮШЭЦАИНЬКСЖЭБГ КЭКБЛУКЮСГБ БХЯ ЕЖДАФЗЗ\*ФУУ Ы АГ\*ЫБОНЭТЩЧЭ КЕНЫЫЗЩЯЮЦЕЛ\*Е ФУПОП\*ЦЫПР ЗЦИ ЭХБЖУ\*РИ Ш\*ДЩОШРХ ЗЕХТЩЫ

Листинг программы на языке C#:

public class Crypter

{

private List<int> \_signs;

private readonly List<char> \_alphabet;

public Crypter(List<int> gammaSigns)

{

\_signs = gammaSigns ?? throw new ArgumentNullException();

\_alphabet = \_lettersTable.Select(x => x.Key).ToList();

}

public string Crypt(string text)

{

var firstMessage = text.ToUpper();

var message = new StringBuilder("");

foreach(var index in firstMessage)

{

if (\_alphabet.Contains(index) || index == ' ')

{

message.Append(index);

}

}

var i = 0;

var encryptedMessage = new StringBuilder("");

Console.WriteLine($"message = {message}");

foreach(var symbol in message.ToString())

{

if (symbol == ' ')

{

encryptedMessage.Append(symbol);

}

else

{

var firstValue = \_lettersTable.First(x => x.Key == symbol).Value;

var secondValue = \_signsTable.First(x => x.Key == \_signs[i]).Value;

\_ = i != \_signs.Count - 1 ? i++ : i = 0;

var targetValue = firstValue ^ secondValue;

if (\_lettersTable.ContainsValue(targetValue))

{

encryptedMessage.Append(\_lettersTable.First(x => x.Value == targetValue).Key);

}

else

{

encryptedMessage.Append('\*');

}

}

}

return encryptedMessage.ToString();

}

private readonly Dictionary<char, int> \_lettersTable = new Dictionary<char, int>

{

{ 'А', 0b000001 },

{ 'Б', 0b001001 },

{ 'В', 0b001010 },

{ 'Г', 0b001011 },

{ 'Д', 0b001100 },

{ 'Е', 0b000010 },

{ 'Ж', 0b001101 },

{ 'З', 0b001110 },

{ 'И', 0b000011 },

{ 'К', 0b001111 },

{ 'Л', 0b010000 },

{ 'М', 0b010001 },

{ 'Н', 0b010010 },

{ 'О', 0b000100 },

{ 'П', 0b010011 },

{ 'Р', 0b010100 },

{ 'С', 0b010101 },

{ 'Т', 0b010110 },

{ 'У', 0b000101 },

{ 'Ф', 0b010111 },

{ 'Х', 0b011000 },

{ 'Ц', 0b011001 },

{ 'Ч', 0b011010 },

{ 'Ш', 0b011011 },

{ 'Щ', 0b011100 },

{ 'Ы', 0b011101 },

{ 'Ь', 0b011110 },

{ 'Э', 0b000110 },

{ 'Ю', 0b000111 },

{ 'Я', 0b001000 },

};

private readonly Dictionary<int, int> \_signsTable = new Dictionary<int, int>

{

{ 0, 0b000000 },

{ 1, 0b000001 },

{ 2, 0b000010 },

{ 3, 0b000011 },

{ 4, 0b000100 },

{ 5, 0b000101 },

{ 6, 0b000110 },

{ 7, 0b000111 },

{ 8, 0b001000 },

{ 9, 0b001001 },

{ 10, 0b001010 },

{ 11, 0b001011 },

{ 12, 0b001100 },

{ 13, 0b001101 },

{ 14, 0b001110 },

{ 15, 0b001111 }

};

}

public class Program

{

public static void Main()

{

Console.WriteLine("Введите гамма массив (16 чисел от 0 до 15) для шифрования");

int[] signs = new int[16];

for (int i = 0; i < 16; i++)

{

signs[i] = Convert.ToInt32(Console.ReadLine());

}

Console.WriteLine("Введите сообщение, которое требуется зашифровать");

var message = Console.ReadLine();

Console.WriteLine();

if (!string.IsNullOrEmpty(message))

{

var normalCount = 0;

for (int i = 0; i < 16; i++)

{

if (!signs.Contains(i))

{

break;

}

normalCount++;

}

if (normalCount != 16)

{

Console.WriteLine("На вход поступила некорректная гамма-последовательность");

}

var crypter = new Crypter(signs.ToList());

var encryptedMessage = crypter.Crypt(message);

Console.WriteLine($"Зашифрованное сообщение - {encryptedMessage}");

}

else

{

Console.WriteLine("На вход поступило пустое сообщение");

}

}

}

Тестирование программы:

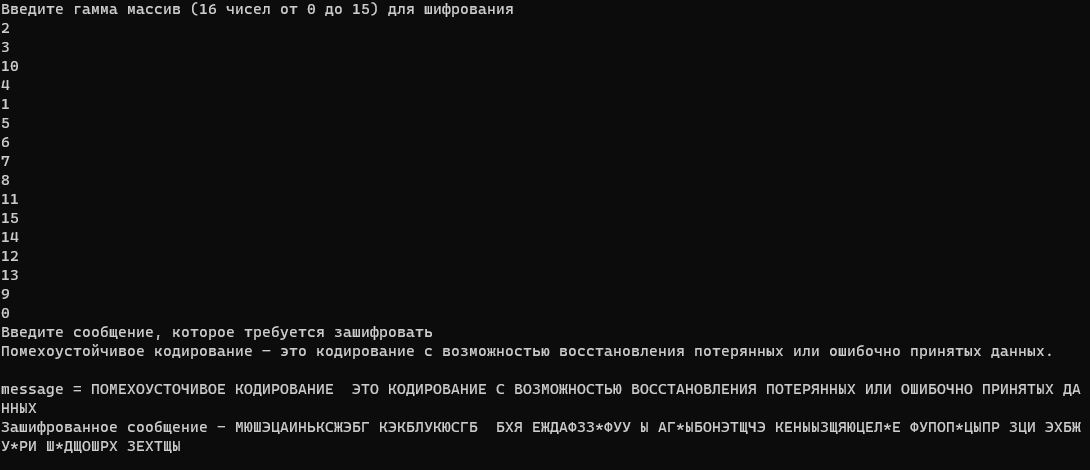


Рисунок 1. Шифрование сообщения методом гаммирования

Вывод

В результате выполнения работы я освоил выполнение шифрования методом гаммирования.