Лабораторная работа № 1

по информационной безопасности

студента группы ИТ-32

Манукова Давида Альбертовича

Выполнение: \_\_\_\_\_\_\_\_\_\_ Защита: \_\_\_\_\_\_\_\_\_\_

**Классические шифры подстановки**

Цель работы: изучение классических криптографических алгоритмов моноалфавитной и многоалфавитной подстановки; криптоанализ классических шифров на основе гистограмм частот встречаемости символов алфавита.

Содержание работы

1. Создать программу, реализующую процесс шифрования/дешифрования текста по следующим алгоритмам:

* аддитивный моноалфавитный шифр с задаваемым смещением
* мультипликативный моноалфавитный шифр с задаваемым смещением
* шифр Плейфера.

1. Провести частотный анализ символов зашифрованного

текста для аддитивного и мультипликативного шифров. Вывести

полученный числовые значения на экран.

1. С помощью полученной частоты встречаемости символов

вручную провести и описать процесс дешифрование первых 15

символов зашифрованного сообщения.

Ход работы

1. Разработал программу, реализующую процесс шифрования/дешифрования.

Аддитивный моноалфавитный шифр с задаваемым смещением.

static string Encr(string text, int key)//Шифрование

{

var codeText = new StringBuilder();

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

{

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

{

if (text[i] == alphabet[j])

{

codeText.Append(alphabet[(j + key) % alphabet.Length]);

}

}

}

return codeText.ToString();

}

static string Descr(string codeText, int key) //Дешифрование

{

var decodeText = new StringBuilder();

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

{

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

{

if (codeText[i] == alphabet[j])

{

decodeText.Append(alphabet[(j - key + alphabet.Length) % alphabet.Length]);

}

}

}

return decodeText.ToString();

}

Мультипликативный моноалфавитный шифр с задаваемым смещением.

static string MultipEncr(string text, int key)

{

var codeText = new StringBuilder();

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

{

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

{

if (text[i] == alphabet[j])

{

codeText.Append(alphabet[(j \* key) % alphabet.Length]);

}

}

}

return codeText.ToString();

}

Шифр Плейфера.

class Program

{

public static T ReadValueFromConsole<T>(string message)

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine(message);

Console.ForegroundColor = ConsoleColor.White;

while (true)

{

try

{

return (T)Convert.ChangeType(Console.ReadLine(), typeof(T));

}

catch (Exception)

{

Console.WriteLine("Возникла ошибка. Повторите ввод");

}

}

}

private static void ActivateMenu()

{

var alph ="АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

var key = ReadValueFromConsole<string>("\nВведите ключевое слово");

var pc = new PlayfairCipher(alph, key);

bool active = true;

while (active)

{

Console.Clear();

var choise = ReadValueFromConsole<int>(@"Выберите действие:

1. Зашифровать указанную строку

2. Десшифровать указанную строку

3. THERE IS NO ESCAPE");

switch (choise)

{

case 1:

Console.Clear();

Console.Write("Шифрование строки.");

var normalMessage = ReadValueFromConsole<string>("\nВведите строку для шифрования: ");

var crypted = pc.Crypt(normalMessage);

Console.WriteLine("\nResult: " + crypted);

Console.WriteLine("De-crypted back: " + pc.Uncrypt(crypted));

Console.ReadKey();

break;

case 2:

Console.Clear();

Console.Write("Дешифрование строки.");

var cryptedMessage = ReadValueFromConsole<string>("\nВведите строку для дешифровки: ");

var normal = pc.Uncrypt(cryptedMessage);

Console.WriteLine("\nResult: " + normal);

Console.WriteLine("Crypted back: " + pc.Crypt(normal));

Console.ReadKey();

break;

case 3:

active = new Random().Next(3) != 0;

if (!active)

{

Console.WriteLine("Lucky One");

Thread.Sleep(2 \* 1000);

}

break;

default:

break;

}

}

}

static void Main(string[] args)

{

ActivateMenu();

}

}

class PlayfairCipher

{

string Alphabet { get; set; }

string KeyWord { get; set; }

public PlayfairCipher(string Alphabet, string KeyWord)

{

this.KeyWord = string.Join("",

KeyWord.ToLower()

.Replace("й", "и")

.Replace("ё", "е")

.Replace("ъ", "ь")

.Replace(" ", "")

.Distinct());

this.Alphabet = Alphabet.ToLower()

.Replace("й", "")

.Replace("ё", "")

.Replace(" ", "")

.Replace("ь", "");

}

public string Crypt(string Message)

{

Func<string[,], int[], int[], string> getPair = GetPairForCrypt;

return CryptOrUncrypt(Message, getPair);

}

public string Uncrypt(string Message)

{

Func<string[,], int[], int[], string> getPair = GetPairForUncrypt;

return CryptOrUncrypt(Message, getPair);

}

string CryptOrUncrypt(string Message, Func<string[,], int[], int[], string> GetPair)

{

var alphMatrix = CreateAlphMatrix();

var bigrams = CreateBigrams(Message);

var returnMessage = "";

var fIndex = new int[] { 0, 0 };

var sIndex = new int[] { 0, 0 };

foreach (var bi in bigrams)

{

fIndex = GetIndex(alphMatrix, bi.FirstLetter);

sIndex = GetIndex(alphMatrix, bi.SecondLetter);

returnMessage += GetPair(alphMatrix, fIndex, sIndex);

}

return returnMessage;

}

string GetPairForUncrypt(string[,] AplhMatrix, int[] FirstLetterIndexs, int[] SecondLetterIndexs)

{

string pair = "";

if (FirstLetterIndexs[0] == SecondLetterIndexs[0])

{

pair += AplhMatrix[

FirstLetterIndexs[0],

((FirstLetterIndexs[1] - 1 + AplhMatrix.GetLength(1)) % AplhMatrix.GetLength(1))];

pair += AplhMatrix[

SecondLetterIndexs[0],

((SecondLetterIndexs[1] - 1 + AplhMatrix.GetLength(1)) % AplhMatrix.GetLength(1))];

}

// В одном столбце

else if (FirstLetterIndexs[1] == SecondLetterIndexs[1])

{

pair += AplhMatrix[

((FirstLetterIndexs[0] - 1 + AplhMatrix.GetLength(0)) % AplhMatrix.GetLength(0)),

FirstLetterIndexs[1]];

pair += AplhMatrix[

((SecondLetterIndexs[0] - 1 + AplhMatrix.GetLength(0)) % AplhMatrix.GetLength(0)),

SecondLetterIndexs[1]];

}

// В одном блоке

else

{

pair += AplhMatrix[

FirstLetterIndexs[0],

SecondLetterIndexs[1]];

pair += AplhMatrix[

SecondLetterIndexs[0],

FirstLetterIndexs[1]];

}

return pair;

}

string GetPairForCrypt(string[,] AplhMatrix, int[] FirstLetterIndexs, int[] SecondLetterIndexs)

{

string pair = "";

if (FirstLetterIndexs[0] == SecondLetterIndexs[0])

{

pair += AplhMatrix[

FirstLetterIndexs[0],

((FirstLetterIndexs[1] + 1) % AplhMatrix.GetLength(1))];

pair += AplhMatrix[

SecondLetterIndexs[0],

((SecondLetterIndexs[1] + 1) % AplhMatrix.GetLength(1))];

}

// В одном столбце

else if (FirstLetterIndexs[1] == SecondLetterIndexs[1])

{

pair += AplhMatrix[

((FirstLetterIndexs[0] + 1) % AplhMatrix.GetLength(0)),

FirstLetterIndexs[1]];

pair += AplhMatrix[

((SecondLetterIndexs[0] + 1) % AplhMatrix.GetLength(0)),

SecondLetterIndexs[1]];

}

// В одном блоке

else

{

pair += AplhMatrix[

FirstLetterIndexs[0],

SecondLetterIndexs[1]];

pair += AplhMatrix[

SecondLetterIndexs[0],

FirstLetterIndexs[1]];

}

return pair;

}

int[] GetIndex(string[,] matrix, string Letter)

{

for (int i = 0; i < matrix.GetLength(0); i++)

{

for (int j = 0; j < matrix.GetLength(1); j++)

{

if (matrix[i, j] == Letter)

return new int[] { i, j };

}

}

throw new Exception();

}

IEnumerable<Bigram> CreateBigrams(string normalMessage)

{

normalMessage = normalMessage.ToLower()

.Replace("й", "и")

.Replace("ё", "е")

.Replace("ь", "ъ")

.Replace(" ", "");

var temp = new List<Bigram>();

for (int i = 0; i < normalMessage.Length; i += 2)

{

if (i == normalMessage.Length - 1)

{

temp.Add(new Bigram()

{

FirstLetter = normalMessage[i].ToString(),

SecondLetter = "х"

});

}

else if (normalMessage[i] == normalMessage[i + 1])

{

temp.Add(new Bigram()

{

FirstLetter = normalMessage[i].ToString(),

SecondLetter = "х"

});

i--;

}

else

{

temp.Add(new Bigram()

{

FirstLetter = normalMessage[i].ToString(),

SecondLetter = normalMessage[i + 1].ToString()

});

}

}

return temp;

}

string[,] CreateAlphMatrix()

{

var let = (KeyWord + Alphabet).Select(x => x.ToString()).Distinct().ToArray();

var alphMatrix = new string[5, 6];

var temp\_index = 0;

Console.WriteLine("New alphabet:");

for (int i = 0; i < alphMatrix.GetLength(0); i++)

{

for (int j = 0; j < alphMatrix.GetLength(1); j++)

{

alphMatrix[i, j] = let[temp\_index];

Console.Write(alphMatrix[i, j] + " ");

temp\_index++;

}

Console.WriteLine("");

}

return alphMatrix;

}

class Bigram

{

public string FirstLetter { get; set; }

public string SecondLetter { get; set; }

public override string ToString()

{

return FirstLetter + " " + SecondLetter;

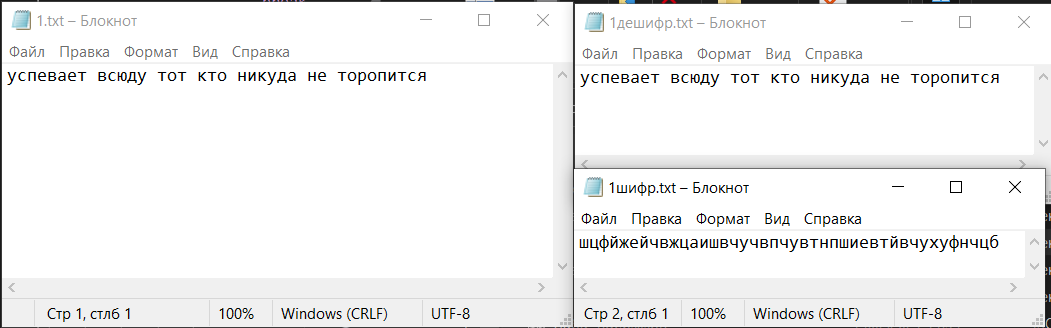
}

}

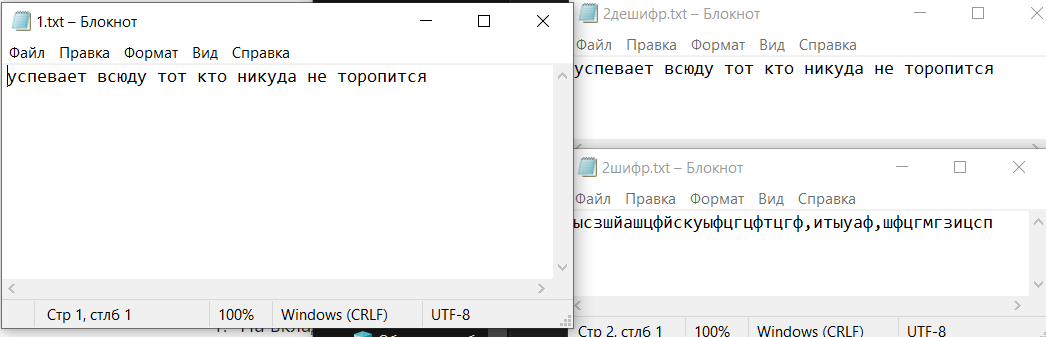
}

1. Результаты работы

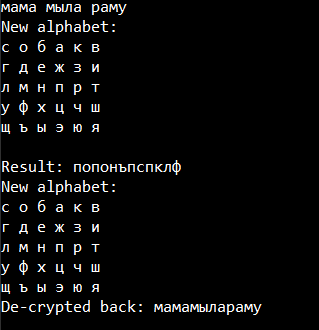
Аддитивный моноалфавитный шифр с задаваемым смещением.



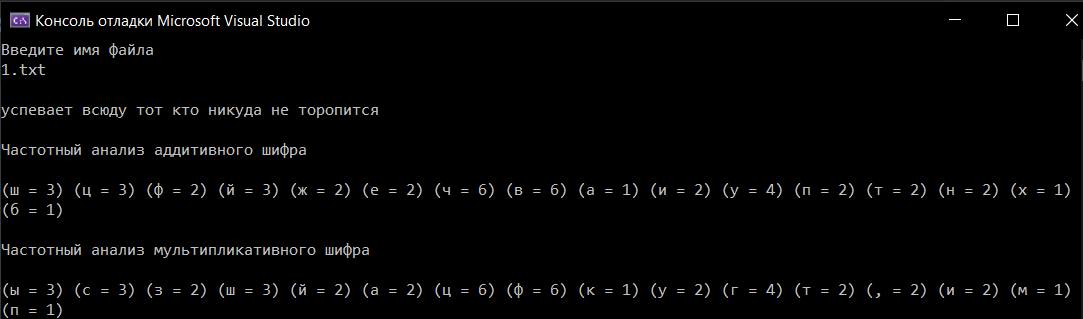
Мультипликативный моноалфавитный шифр с задаваемым смещением.



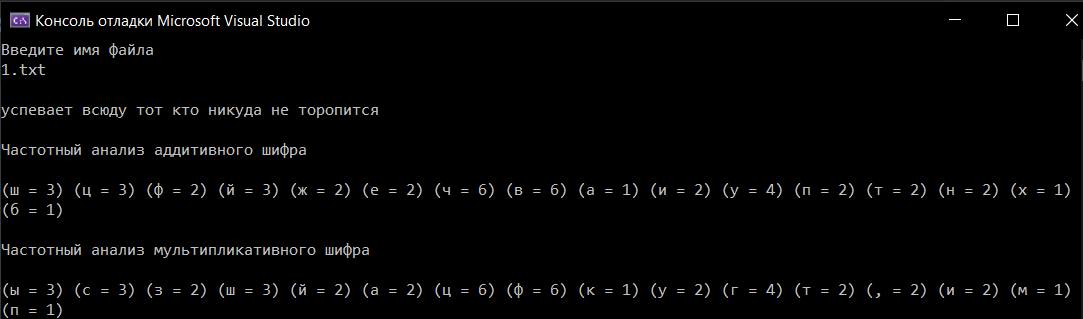
Шифр Плейфера.

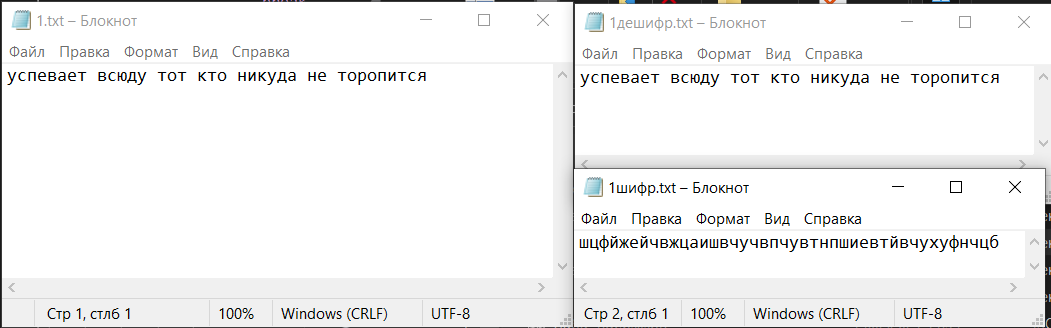


1. Частотный символов зашифрованного текста



1. Дешифровка на основе частного анализа для аддитивного шифра:





Первые символов из зашифрованного сообщения: ш, ц, ф, й, ж, е, й, ч, в, ж, ц, а, и, ш, в.

абвгдеёжзийклмнопрстуфхцчшщъыьэюя\_,.

Часто встречающиеся буквы в русском языке: о (11%), е(8,5%), а(8%).

Предположим, что «ч» из зашифрованного текста = «о» из алфавита. Тогда сдвиг будет равен 9.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| а | б | в | г | д | е | ё | ж | з | и | й | к | л | м | н | о | п | р | с | т | у | ф | х | ц | ч | ш | щ | ъ | ы | ь | э | ю | я | \_ | , | . |
| и | й | к | л | м | н | о | п | р | с | т | у | ф | х | ц | ч | ш | щ | ъ | ы | ь | э | ю | я | \_ | , | . | а | б | в | г | д | е | ё | ж | з |

Неверно, т.к. первый символ = «,» => величина сдвига неверна.

Предположим, что «в» из зашифрованного текста = «о» из алфавита. Тогда сдвиг будет равен 13.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| а | б | в | г | д | е | ё | ж | з | и | й | к | л | м | н | о | п | р | с | т | у | ф | х | ц | ч | ш | щ | ъ | ы | ь | э | ю | я | \_ | , | . |
| ц | ч | ш | щ | ъ | ы | ь | э | ю | я | \_ | , | . | а | б | в | г | д | е | ё | ж | з | и | й | к | л | м | н | о | п | р | с | т | у | ф | х |

Неверно, т.к. первый символ = «л» => величина сдвига неверна.

Предположим, что «у» из зашифрованного текста = «о» из алфавита. Тогда сдвиг будет равен 5.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| а | б | в | г | д | е | ё | ж | з | и | й | к | л | м | н | о | п | р | с | т | у | ф | х | ц | ч | ш | щ | ъ | ы | ь | э | ю | я | \_ | , | . |
| е | ё | ж | з | и | й | л | л | м | н | о | п | р | с | т | у | ф | х | ц | ч | ш | щ | ъ | ы | ь | э | ю | я | \_ | , | . | а | б | в | г | д |

Успевает тот кт….

Таким образом следует, что сдвиг равный 5 верный.

Вывод: в ходе выполнения лабораторной работы были изучены классические криптографические алгоритмы и классические шифра: Цезаря, Плейфера, Афинный. На основе частного анализа текста, а также информации о наиболее часто встречающихся в текстах написанных на русском языке символах был описан процесс дешифровки двух классических шифров.