Учреждение образования

«Белорусский государственный технологический университет»

**Отчет по лабораторной работе №2**

по дисциплине: «Защита информации и надежность информационных систем»

Исполнитель:

Воликов Д. А., ФИТ 4-7

Руководитель:

Асс. Савельева М. Г.

Минск 2024

# Лабораторная работа №2. Исследование криптографических шифров на основе подстановки (замены) символов

**Цель**: изучение и приобретение практических навыков разработки и использования приложений для реализации подстановочных шифров.

**Задачи**:

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости подстановочных шифров.
2. Разработать приложение для реализации указанных преподавателем методов подстановочного зашифрования/расшифрования.
3. Выполнить исследование криптостойкости шифров на основе статистических данных о частотах появления символов в исходном и зашифрованном сообщениях.
4. Оценить скорость зашифрования/расшифрования реализованных способов шифров.
5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

## Теоретические сведения

Сущность подстановочного шифрования состоит в том, что исходный текст (из множества *М*) и зашифрованный текст (из множества *С*) основаны на использовании одного и того же или разных алфавитов, а тайной или ключевой информацией является алгоритм подстановки.

Выделим следующий тип подстановочных шифров – моноалфавитные шифры подстановки. В данных шифрах операция замены производится раздельно над каждым одиночным символом сообщения *М*. Для наглядной демонстрации шифра простой замены достаточно выписать под заданным алфавитом тот же алфавит, но в другом порядке или, например, со смещением. Записанный таким образом алфавит называют алфавитом замены.

Максимальное количество ключей для любого шифра этого вида не превышает *N*!, где *N* – количество символов в алфавите.

Применяя одновременно операции сложения и умножения по модулю *n* над элементами множества (индексами букв алфавита), можно получить систему подстановок, которую называют аффинной системой подстановок Цезаря. Определим процедуру зашифрования в такой системе:

*y* ≡ *ax* + *b* mod *N*

где *a* и *b* – целые числа.

При этом взаимно однозначные соответствия между открытым текстом и шифртекстом будут иметь место при выполнении условий: 0 ≤ *a*, *b* < *N*, наибольший общий делитель (НОД) чисел *a*, *N* равен 1, т. е. эти числа являются взаимно простыми.

Разберём метод шифрования на примере сообщения *M* = «VENIVIDIVICI». Пусть *N* = 26, *а* = 3, *b* = 5. Тогда НОД (3, 26) = 1, и мы получаем следующее соответствие между индексами букв:

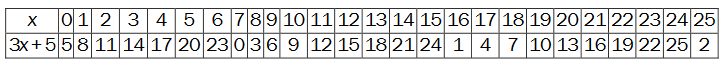


Рисунок 1 – Соответствия между индексами букв

Преобразуя числа в буквы английского алфавита, получаем следующее соответствие для букв открытого текста и шифртекста:

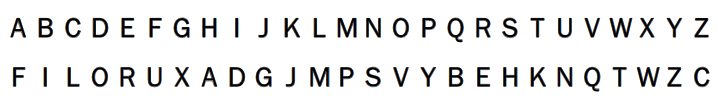


Рисунок 2 – Соответствия между буквами алфавита

Таким образом, зашифрованное сообщение будет иметь следующий вид:   
*С* = «QRSCQCOCQCLC». Расшифрование основано на использовании соотношения

*x* ≡ *a*-1 (*y* + *N* – *b*) mod *N*,

где *a*-1 – обратное к *a* число по модулю *N*, т. е. оно удовлетворяет уравнению

*аa*-1 ≡ 1 mod *N*.

Следующий тип подстановочных шифров – полиграммные шифры. В таких шифрах одна подстановка соответствует сразу нескольким символам исходного текста.

Первым известным шифром этого типа является шифр Порты. Шифр представляется в виде таблицы. Наверху горизонтально и слева вертикально записывается стандартный алфавит. В ячейках таблицы записываются числа в определенном порядке.

С точки зрения криптостойкости рассматриваемый тип шифров имеет преимущества перед моноалфавитными шифрами. Это связано с тем, что, во-первых, распределение частот групп букв значительно более равномерное, чем отдельных символов. Во-вторых, для эффективного частотного анализа требуется больший размер зашифрованного текста, так как число различных групп букв значительно больше, чем мощность алфавита.

Полиалфавитные (или многоалфавитные) шифры состоят из нескольких шифров однозначной замены. Выбор варианта алфавита для зашифрования одного символа зависит от особенностей метода шифрования.

В 1586 г. французский дипломат Блез Виженер представил перед комиссией Генриха III описание простого, но довольно стойкого шифра, в основе которого лежит таблица Трисемуса. В этом шифре мы имеем дело с последовательностью сдвигов, циклически повторяющейся. Основная идея заключается в следующем. Создается таблица (таблица Виженера) размером *N*×*N* (*N* – число знаков в используемом алфавите). Эти знаки могут включать не только буквы, но и, например, пробел или иные знаки. В первой строке таблицы записывается весь используемый алфавит. Каждая последующая строка получается из предыдущего циклического сдвига последней на 1 символ влево. Таким образом, при мощности алфавита (английского языка), равной 26, необходимо выполнить последовательно 25 сдвигов для формирования всей таблицы.

## 1.2 Практическое задание

**Вариант 2**. По условию язык алфавита – русский. Использовать следующие шифры:

* Шифр аффинной системы подстановок Цезаря; *a* = 7, *b* = 10.
* Шифр Виженера, ключевое слово – собственная фамилия.

**Задание 1**. Разработать авторское приложение в соответствии с целью лабораторной работы. Приложение должно выполнять зашифрование/расшифрование текстовых документов (объемом не менее 5 тысяч знаков), созданных на основе алфавита языка в соответствии с вариантом.

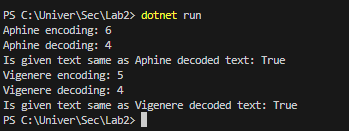


Рисунок 3 – Результат работы программы

Программа выводит время зашифрования/расшифрования текстового файла, размером в 5000+ символов. А также можно увидеть результат успешного декодирования зашифрованного текста.

**Задание 2**. Сформировать гистограммы частот появления символов для исходного и зашифрованных сообщений.

Гистограмма частот появления символов для исходного текста предоставлена на рисунке 4.

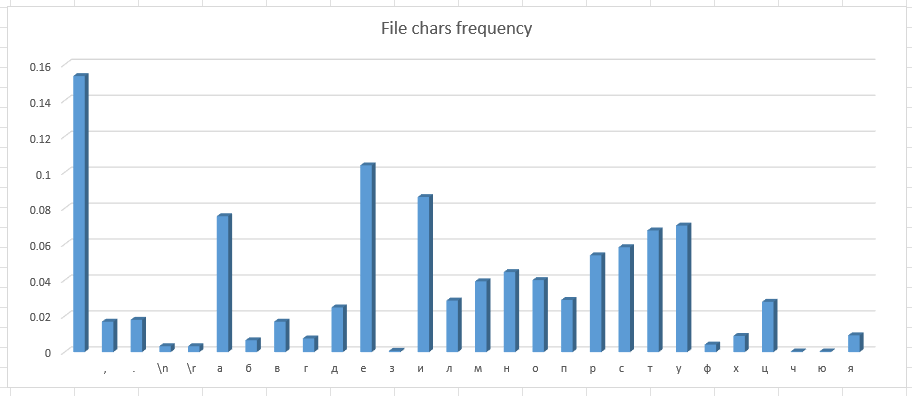


Рисунок 4 – Гистограмма частот появления символов для исходного сообщения

Гистограмма частот появления символов для зашифрованного текста методом Аффинной подстановки Цезаря предоставлена на рисунке 5.

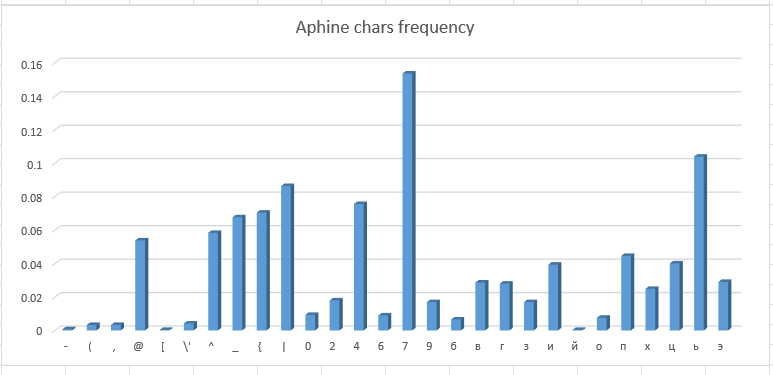


Рисунок 5 – Гистограмма частот появления символов для зашифрованного сообщения методом Аффинной подстановки

Гистограмма частот появления символов для зашифрованного текста методом Виженера предоставлена на рисунке 6.

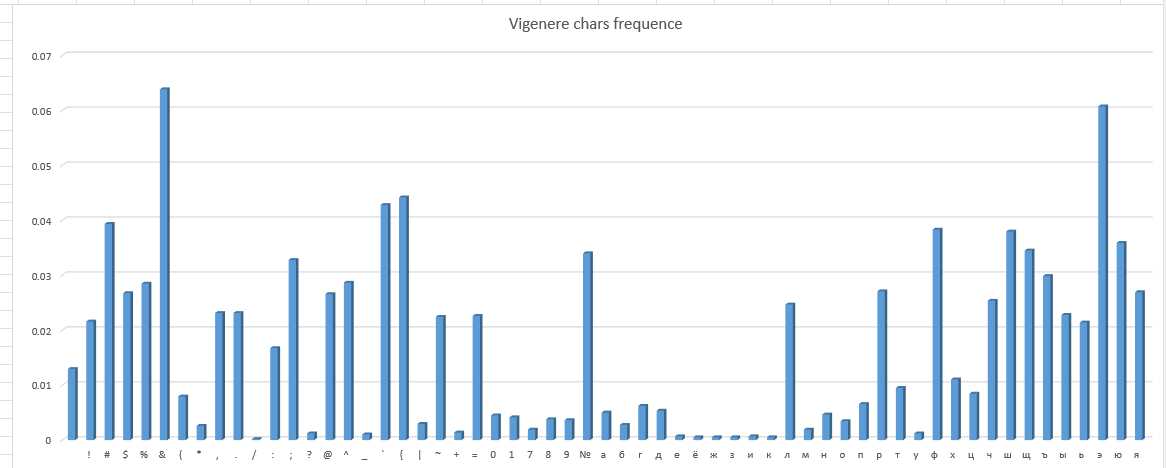


Рисунок 6 – Гистограмма частот появления символов для зашифрованного сообщения методом Виженера

**Задание №3**. Оценить время выполнения операций зашифрования/расшифрования авторского приложения.

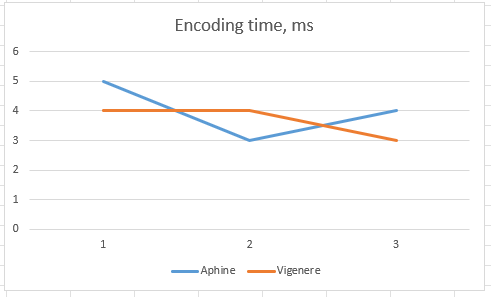


Рисунок 7 – Время зашифрования текста в миллисекундах

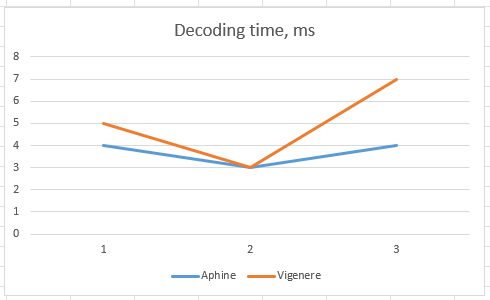


Рисунок 8 – Время расшифрования текста в миллисекундах

**Вывод**: в ходе этой лабораторной работы я изучил различные криптографические методы шифрования на основе подстановки (замены) символов. Написал программу, которая зашифровывает/расшифровывает исходный текст. Рассчитал время исполнения каждого процесса, а также построил графики частот появления символов на основе исходного текста, а также на основе зашифрованных текстов.

# Листинг «Лабораторная работа №2»

static int GCD(int first, int second) {

if (first <= 0 || second <= 0)

{

return -1;

}

int remainder = 0;

while (first % second != 0)

{

remainder = first % second;

first = second;

second = remainder;

}

return remainder;

}

static int ReverseModuloNumber(int a, int N) {

int left = 0;

int right = 1;

int reverse = 0;

int NN = N;

while (N % a != 0) {

int q = N / a;

int remainder = N % a;

N = a;

a = remainder;

reverse = left - right \* q;

left = right;

right = reverse;

}

if (reverse < 0) {

reverse += NN;

}

return reverse;

}

static string AphineEncode(int a, int b, int N, string message, string characters) {

if (a < 0 || a > N)

{

throw new Exception("Invalid param a. Must be 0 <= a < N");

}

if (b < 0 || b > N)

{

throw new Exception("Invalid param b. Must be 0 <= b < N");

}

if (GCD(a, N) != 1) {

throw new Exception("Invalid param a\nGCD(a, N) must be 1");

}

string encodedMessage = "";

foreach (var ch in message) {

int index = (a \* characters.IndexOf(ch) + b) % N;

encodedMessage += characters[index];

}

return encodedMessage;

}

static string AphineDecode(int a, int b, int N, string message, string characters) {

if (a < 0 || a > N)

{

throw new Exception("Invalid param a. Must be 0 <= a < N");

}

if (b < 0 || b > N)

{

throw new Exception("Invalid param b. Must be 0 <= b < N");

}

if (GCD(a, N) != 1) {

throw new Exception("Invalid param a\nGCD(a, N) must be 1");

}

var reversedModulo = ReverseModuloNumber(a, N);

string decodedMessage = "";

foreach (var ch in message) {

int index = reversedModulo \* (characters.IndexOf(ch) + N - b) % N;

decodedMessage += characters[index];

}

return decodedMessage;

}

static string VigenereEncode(string message, string key, string characters) {

string encodedMessage = "";

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

int keyIndex = i % key.Length;

int messageCharIndex = characters.IndexOf(message[i]);

int keyCharIndex = characters.IndexOf(key[keyIndex]);

int index = (messageCharIndex + keyCharIndex) % characters.Length;

encodedMessage += characters[index];

}

return encodedMessage;

}

static string VigenereDecode(string encodedMessage, string key, string characters) {

string decodedMessage = "";

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

int keyIndex = i % key.Length;

int messageCharIndex = characters.IndexOf(encodedMessage[i]);

int keyCharIndex = characters.IndexOf(key[keyIndex]);

int index = (messageCharIndex - keyCharIndex + characters.Length) % characters.Length;

decodedMessage += characters[index];

}

return decodedMessage;

}

string characters = "0123456789абвгдеёжзийклмнопрстуфхцчшщъыьэюя `~!@№#$;%:^&?\*()-\_=+.,/|{[]}\n\r\"\'";

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

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

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

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

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

string csvAphineFilePath = @"./aphine.csv";

string csvVigenereFilePath = @"./vigenere.csv";

foreach (var ch in fileContent) {

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

alphabet[ch] += 1;

}

}

File.WriteAllText(csvFileContent, "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(csvFileContent, ch + "\t" + (double)item.Value / fileContent.Length + '\n');

}

var a = 7;

var b = 10;

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

var aphineEncodedMessage = AphineEncode(a, b, characters.Length, fileContent, characters);

watch.Stop();

var elapsedMs = watch.ElapsedMilliseconds;

Console.WriteLine("Aphine encoding: " + elapsedMs);

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

var aphineDecodedMessage = AphineDecode(a, b, characters.Length, aphineEncodedMessage, characters);

watch.Stop();

elapsedMs = watch.ElapsedMilliseconds;

Console.WriteLine("Aphine decoding: " + elapsedMs);

foreach (var ch in aphineEncodedMessage) {

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

aphineAlphabet[ch] += 1;

}

}

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

foreach (var item in aphineAlphabet) {

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

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

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

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

item.Key.ToString();

File.AppendAllText(csvAphineFilePath, ch + "\t" + (double)item.Value / aphineEncodedMessage.Length + '\n');

}

Console.WriteLine("Is given text same as Aphine decoded text: " + (fileContent == aphineDecodedMessage));

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

var vigenereEncodedMessage = VigenereEncode(fileContent, "воликов", characters);

watch.Stop();

elapsedMs = watch.ElapsedMilliseconds;

Console.WriteLine("Vigenere encoding: " + elapsedMs);

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

var vigenereDecodedMessage = VigenereDecode(vigenereEncodedMessage, "воликов", characters);

watch.Stop();

elapsedMs = watch.ElapsedMilliseconds;

Console.WriteLine("Vigenere decoding: " + elapsedMs);

Console.WriteLine("Is given text same as Vigenere decoded text: " + (vigenereDecodedMessage == fileContent));

foreach (var ch in vigenereEncodedMessage) {

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

vigenereAlphabet[ch] += 1;

}

}

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

foreach (var item in vigenereAlphabet) {

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

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

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

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

item.Key.ToString();

File.AppendAllText(csvVigenereFilePath, ch + "\t" + (double)item.Value / vigenereEncodedMessage.Length + '\n');

}

Листинг 1 – Код программы на языке C#