МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Чувашский государственный университет имени И.Н. Ульянова»

Факультет информатики и вычислительной техники

Кафедра вычислительной техники

Лабораторная работа №8

По дисциплине “Защита информации” на тему:

«Метод изменения интервала между предложениями»

Выполнил: ст. гр. ИВТ-11-21

Еремеев С.А.

Проверил: канд. техн. наук. Ковалев С.В.

Чебоксары 2025

**Теоретическая часть**

# Цель работы

Целью работы является знакомство со стеганографическим методом скрытия информации путем изменения интервала между предложениями.

# Задание на работу

В данной лабораторной работе необходимо программно реализовать алгоритмы прямого и обратного стеганографического преобразования.

Пустой и заполненный текстовые контейнеры должны задаваться пользователем в виде файлов либо через графический интерфейс – стандартный диалог выбора файла и/или поле ввода имени файла, либо через параметры командной строки. Выбор вида интерфейса (графический или командной строки) – на усмотрение студента, выполняющего работу. Аналогичным образом задается скрываемое сообщение.

Перед прямым стеганографическим преобразованием должна выполняться предварительная обработка пустого контейнера для замены двух и более пробелов на один пробел после окончания предложения, а также добавление к сообщению маркера окончания. Маркер окончания выбирается студентом самостоятельно таким образом, чтобы выбранный маркер не являлся частью исходного сообщения. До начала стеганографического преобразования предобработчик должен выводить пользователю информацию о максимальной емкости контейнера и возможности размещения заданного сообщения в выбранном контейнере.

Для простоты реализации можно принять, что сообщение начинает размещаться в контейнере с первой возможной позиции (то есть, после первого же предложения) и размещается последовательно; при этом стеганографический ключ отсутствует.

# Основные сведения

Основные понятия стеганографии и обобщенная структурная схема стеганографической системы приведены в тексте лекций.

Метод изменения интервала между предложениями основан на размещении одного или двух пробелов после каждого символа завершения предложения. При этом, обычно, единичным пробелом кодируется бит «1», а двойным пробелом – бит «0». Признаком завершения предложения можно считать точку, вопросительный и/или восклицательный знаки, после которых следует пробел.Скрываемое сообщение в двоичном формате встраивается в контейнер-текст путем размещения соответствующего каждому биту числа пробелов после каждого предложения.

Поскольку контейнер имеет существенно больший объем, чем сообщение, протокол стеганографического обмена должен предусматривать некоторый маркер завершения сообщения. Этот маркер дописывается в конец сообщения на этапе предварительной обработки, до поступления сообщения в стеганокодер. При извлечении сообщения из контейнера, обнаружение маркера позволяет прервать обработку и не анализировать оставшуюся часть контейнера.

В качестве примера представим себе контейнер следующего вида:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| С | л | о | в | а |  | ф | р | а | з | ы |  | 1 | . |  | С | л | о | в | а |  | ф | р | а | з |
| ы |  | 2 | . |  | С | л | о | в | а |  | ф | р | а | з | ы |  | 3 | . |  | С | л | о | в | а |
|  | ф | р | а | з | ы |  | 4 | . |  | С | л | о | в | а |  | ф | р | а | з | ы |  | 5 | . |  |
| С | л | о | в | а |  | ф | р | а | з | ы |  | 6 | ! |  | С | л | о | в | а |  | ф | р | а | з |
| ы |  | 7 | . |  | С | л | о | в | а |  | ф | р | а | з | ы |  | 8 | ? |  | С | л | о | в | а |
|  | ф | р | а | з | ы |  | 9 | . |  | С | л | о | в | а |  | ф | р | а | з | ы |  | А | . |  |

Этот контейнер состоит из десяти предложений и, следовательно, может быть использован для размещения десяти бит скрываемого сообщения. Сообщение представляется в двоичном формате; каждый пробел после очередного предложения контейнера заменяется *двумя* пробелами, если очередной бит сообщения равен нулю, или *одним* пробелом (то есть, остается без изменений), если очередной бит сообщения равен единице. Пусть в нашем примере символ сообщения имеет значение **"М"**, в двоичной форме код символа **"М"** будет представлен битами ASCII("М") = 140d = 10001100b. Тогда заполненный контейнер будет выглядеть следующим образом:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| С | л | о | в | а |  | ф | р | а | з | ы |  | 1 | . |  | С | л | о | в | а |  | ф | р | а | з |
| ы |  | 2 | . |  |  | С | л | о | в | а |  | ф | р | а | з | ы |  | 3 | . |  |  | С | л | о |
| в | а |  | ф | р | а | з | ы |  | 4 | . |  |  | С | л | о | в | а |  | ф | р | а | з | ы |  |
| 5 | . |  | С | л | о | в | а |  | ф | р | а | з | ы |  | 6 | ! |  | С | л | о | в | а |  | ф |
| р | а | з | ы |  | 7 | . |  |  | С | л | о | в | а |  | ф | р | а | з | ы |  | 8 | ? |  |  |
| С | л | о | в | а |  | ф | р | а | з | ы |  | 9 | . |  | С | л | о | в | а |  | ф | р | а | з |
| ы |  | А | . |  |

Затемнением выделены пробелы после окончания предложений, кодирующие биты скрываемого сообщения. Эта иллюстрация также более наглядно демонстрирует необходимость добавления маркера окончания сообщения до начала стеганографического преобразования: в противном случае на конце сообщения у нас ошибочно «декодировались» бы два единичных бита из той части контейнера, которая не понадобилась для скрытия сообщения.

На практике, стеганографические системы, реализующие рассмотренный и подобные методы, осуществляют предварительную обработку пустого контейнера. В ходе предобработки из текста контейнера удаляются «лишние» пробелы, которые могли оказаться в конце предложений на этапе создания контейнера: это может произойти из-за особенностей конкретного текстового редактора, системы распознавания сканированного текста, или действий человека, набиравшего текст.

**Практическая часть**

# Полный текст:

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

class Program

{

static List<string> PreprocessContainer(string text)

{

List<string> sentences = new List<string>();

string currentSentence = "";

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

{

currentSentence += text[i];

if (".!?".Contains(text[i]) && (i + 1 >= text.Length || text[i + 1] == ' '))

{

sentences.Add(currentSentence);

currentSentence = "";

while (i + 1 < text.Length && text[i + 1] == ' ') i++;

}

}

if (!string.IsNullOrEmpty(currentSentence))

sentences.Add(currentSentence);

return sentences;

}

static string TextToBits(string text)

{

StringBuilder bits = new StringBuilder();

foreach (char c in text)

{

bits.Append(Convert.ToString(c, 2).PadLeft(8, '0'));

}

return bits.ToString();

}

static string BitsToText(string bits)

{

StringBuilder text = new StringBuilder();

for (int i = 0; i < bits.Length; i += 8)

{

if (i + 8 <= bits.Length)

{

string byteStr = bits.Substring(i, 8);

text.Append((char)Convert.ToInt32(byteStr, 2));

}

}

return text.ToString();

}

static void EmbedMessage(string containerFile, string messageFile, string outputFile)

{

string containerText = File.ReadAllText(containerFile, Encoding.UTF8);

string message = File.ReadAllText(messageFile, Encoding.UTF8);

List<string> sentences = PreprocessContainer(containerText);

int capacity = sentences.Count - 1;

message += "#";

string messageBits = TextToBits(message);

if (messageBits.Length > capacity)

{

Console.WriteLine($"Ошибка: сообщение ({messageBits.Length} бит) превышает емкость контейнера ({capacity} бит)");

return;

}

StringBuilder result = new StringBuilder();

for (int i = 0; i < sentences.Count; i++)

{

result.Append(sentences[i]);

if (i < sentences.Count - 1)

{

if (i < messageBits.Length)

{

result.Append(messageBits[i] == '0' ? " " : " ");

}

else

{

result.Append(" ");

}

}

}

File.WriteAllText(outputFile, result.ToString(), Encoding.UTF8);

Console.WriteLine($"Сообщение зашифровано в {outputFile}");

}

static void ExtractMessage(string containerFile)

{

string text = File.ReadAllText(containerFile, Encoding.UTF8);

List<string> sentences = PreprocessContainer(text);

StringBuilder bits = new StringBuilder();

int currentPosition = 0;

for (int i = 0; i < sentences.Count - 1; i++)

{

int startIndex = text.IndexOf(sentences[i], currentPosition);

if (startIndex == -1) break;

startIndex += sentences[i].Length;

int endIndex = text.IndexOf(sentences[i + 1], startIndex);

if (endIndex == -1) break;

int spaceCount = text.Substring(startIndex, endIndex - startIndex).Length;

bits.Append(spaceCount == 1 ? "1" : "0");

currentPosition = endIndex;

}

string markerBits = TextToBits("#");

string messageBits = "";

for (int i = 0; i <= bits.Length - markerBits.Length; i++)

{

if (i + markerBits.Length <= bits.Length && bits.ToString().Substring(i, markerBits.Length) == markerBits)

{

messageBits = bits.ToString().Substring(0, i);

break;

}

}

string message = BitsToText(messageBits);

Console.WriteLine($"Извлеченное сообщение: {message}");

}

static void Main(string[] args)

{

if (args.Length < 1)

{

PrintUsage();

return;

}

string command = args[0].ToLower();

if (command == "embed" && args.Length == 4)

{

string containerFile = args[1];

string messageFile = args[2];

string outputFile = args[3];

if (!File.Exists(containerFile) || !File.Exists(messageFile))

{

Console.WriteLine("Ошибка: один из файлов не найден");

return;

}

EmbedMessage(containerFile, messageFile, outputFile);

}

else if (command == "extract" && args.Length == 2)

{

string containerFile = args[1];

if (!File.Exists(containerFile))

{

Console.WriteLine("Ошибка: файл не найден");

return;

}

ExtractMessage(containerFile);

}

else

{

PrintUsage();

}

}

static void PrintUsage()

{

Console.WriteLine("Использование:");

Console.WriteLine(" Для встраивания: Lab8.exe embed <container\_file> <message\_file> <output\_file>");

Console.WriteLine(" Для извлечения: Lab8.exe extract <container\_file>");

}

}

# Прямое преобразование

static void EmbedMessage(string containerFile, string messageFile, string outputFile)

{

string containerText = File.ReadAllText(containerFile, Encoding.UTF8);

string message = File.ReadAllText(messageFile, Encoding.UTF8);

List<string> sentences = PreprocessContainer(containerText);

int capacity = sentences.Count - 1;

message += "#";

string messageBits = TextToBits(message);

if (messageBits.Length > capacity) return;

StringBuilder result = new StringBuilder();

for (int i = 0; i < sentences.Count; i++)

{

result.Append(sentences[i]);

if (i < sentences.Count - 1)

{

if (i < messageBits.Length)

result.Append(messageBits[i] == '0' ? " " : " ");

else

result.Append(" ");

}

}

File.WriteAllText(outputFile, result.ToString(), Encoding.UTF8);

}

# Обратное преобразование

static void ExtractMessage(string containerFile)

{

string text = File.ReadAllText(containerFile, Encoding.UTF8);

List<string> sentences = PreprocessContainer(text);

StringBuilder bits = new StringBuilder();

int currentPosition = 0;

for (int i = 0; i < sentences.Count - 1; i++)

{

int startIndex = text.IndexOf(sentences[i], currentPosition);

if (startIndex == -1) break;

startIndex += sentences[i].Length;

int endIndex = text.IndexOf(sentences[i + 1], startIndex);

if (endIndex == -1) break;

int spaceCount = text.Substring(startIndex, endIndex - startIndex).Length;

bits.Append(spaceCount == 1 ? "1" : "0");

currentPosition = endIndex;

}

string markerBits = TextToBits("#");

string messageBits = "";

for (int i = 0; i <= bits.Length - markerBits.Length; i++)

{

if (i + markerBits.Length <= bits.Length && bits.ToString().Substring(i, markerBits.Length) == markerBits)

{

messageBits = bits.ToString().Substring(0, i);

break;

}

}

string message = BitsToText(messageBits);

}

# Текст скрываемого сообщения

Скрываемое сообщение: message.txt.



Маркер окончания: "#"

Сообщение в виде битовой строки:

"Eremeev Sergey" →

E: 01000101

r: 01110010

e: 01100101

m: 01101101

e: 01100101

e: 01100101

v: 01110110

(пробел): 00100000

S: 01010011

e: 01100101

r: 01110010

g: 01100111

e: 01100101

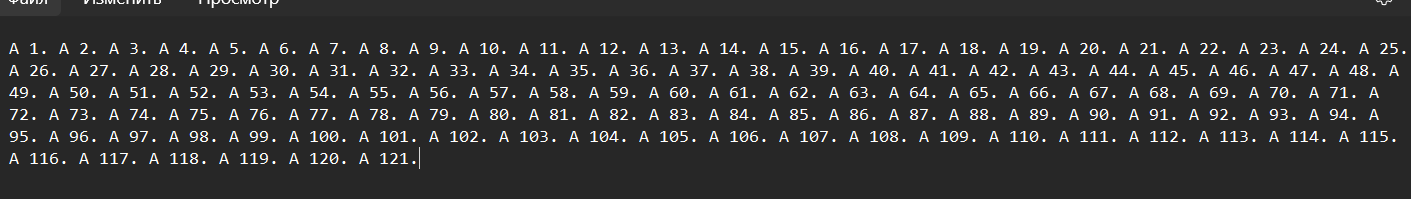
y: 01111001

Итог: "0100010101110010011001010110110101100101011001010111011000100000010100110110010101110010011001110110010101111001"

Маркер в виде битовой строки:

"#" → 00100011

# Открытый файл empty.txt:



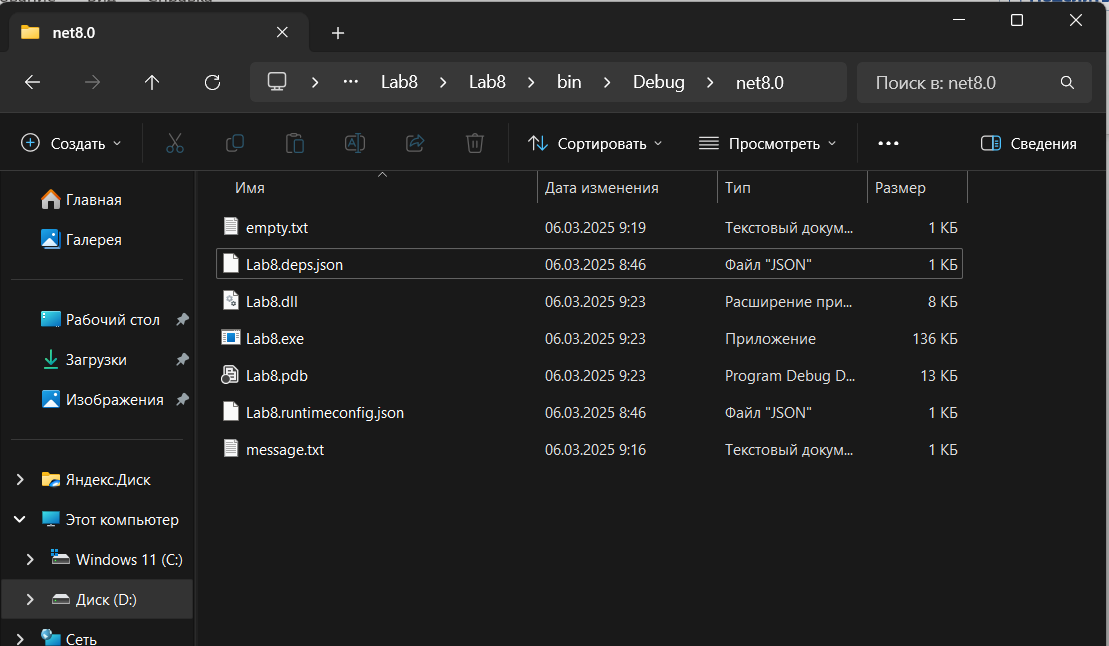
# Заполненный файл empty.txt:

Формируется после команды:

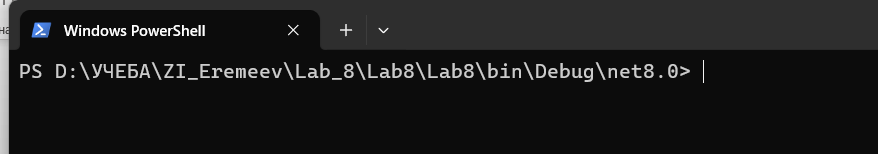
PS D:\УЧЕБА\ZI\_Eremeev\Lab\_8\Lab8\Lab8\bin\Debug\net8.0> .\Lab8.exe embed empty.txt message.txt filled.txt

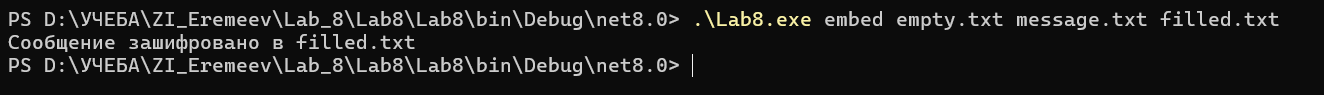
# Примеры

Изначальный вид директории:

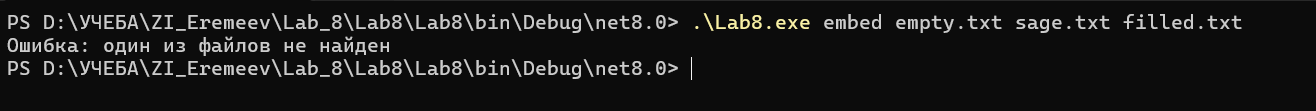


Запускаем powershell:

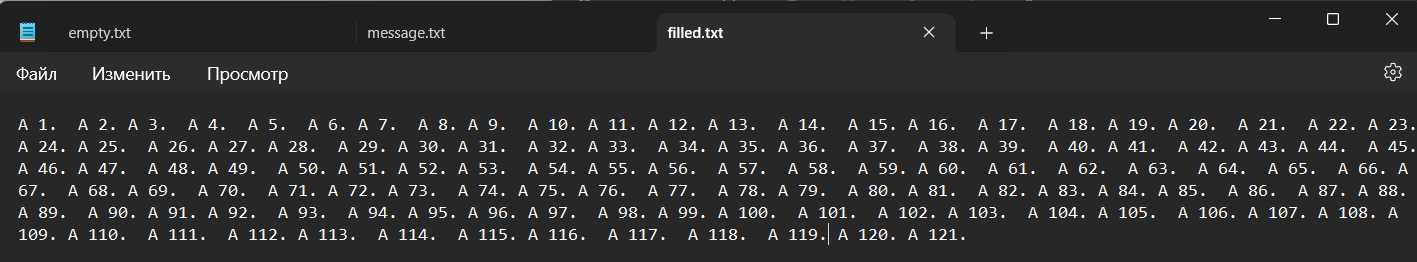




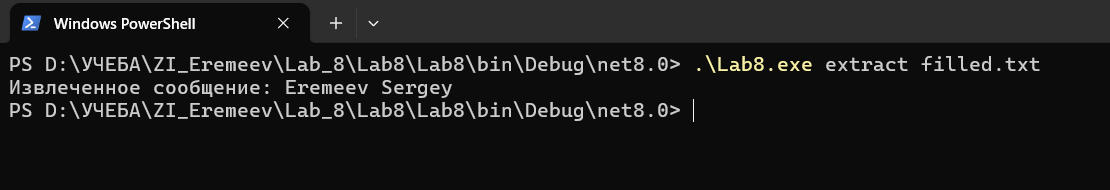
Обработка ошибкок:



Содержимое filed.txt



Для извлечения исходного текста:



# Вывод

В ходе работы был реализован стеганографический метод, использующий пробелы между предложениями для скрытия информации.