**Министерство науки и высшего образования РФ**

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

факультет БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ

ТЕХНОЛОГИЙ

Основы стеганографии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

«Основы текстовой стеганографии»

Выполнил:

Студент группы N3351

Фам Хю Хоанг



Проверил: ассистент ФБИТ,

Университет ИТМО,

Давыдов Вадим Валерьевич

Санкт-Петербург

2020

**Цель работы:**

Целью данной лабораторной работы :

* Применение текстовых стеганографических методов для сокрытия
* Извлечение сообщения из стегоконтейнера
* Анализ исходного текста и стегоконтейнера.

**Теоретическая часть:**

Стеганография -Cпособ передачи или хранения информации с учётом сохранения в тайне самого факта такой передачи (хранения). Этот термин ввёл в 1499 году аббат бенедиктинского монастыря Св. Мартина в Шпонгейме Иоганн Тритемий в своём трактате «Стеганография», зашифрованном под магическую книгу. В данной лабораторной работе были рассмотрены следующие 3 алгоритма:

1. Метод замены символов (метод знаков одинакового начертания);
2. Метод с использованием пробелов (метод хвостовых пробелов);
3. Метод с добавлением служебных символов.

Метод знаков одинакового начертания предполагает подмену (бит 1) или отказ от такой подмены (бит 0) русского символа латинским того же начертания. Например, латинская “c” на кириллическую “c”. В моем случае была реализована замена букв с и р .

Метод хвостовых пробелов предполагает дописывание в конце каждой строки файла-контейнера одного пробела, в случае кодирования единичного бита стеганосообщения. Если нужно закодировать нулевой бит, пробел в конце строки не дописывается.

Метод с добавление служебных символов предполагает добавление служебных символов или подмену одних служебных символов на другие. Кодирование осуществляется таким же образом, как описано выше.

Для того, чтобы зашифровать исходное сообщение, оно преобразуется в двоичный код. Далее двоичная последовательность записывается одним из указанных выше методом. Извлечение информации происходит следующим образом: исходя из выбранного метода текстовый контейнер считывается и записывает полученную последовательность битов. Далее двоичный код преобразуется в зашифрованное сообщение.

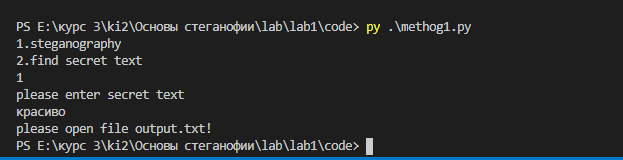
**Практическая часть:**

Исходным контейнером в моей работе является рассказ А. П. Чехова «Агафья» состоящий из 19271 символа в кодировке UTF-8.

Все алгоритмы были написаны на языке Python. Для каждого метода был написан отдельный файл.

Далее следуют три файла methog1.py, methog2.py и methog3.py, отвечающие за первый, второй и третий методы соответственно. Каждый из скриптов принимает ключи и аргументы. При передаче “1” скрипт будет работать в режиме шифрования. В режиме шифрования нужно один файл становится контейнером(orginal\_document.txt), а во второй происходит запись(output.txt). Ключ “2” нужен для режима дешифрования.

**Метод замены символов (метод знаков одинакового начертания)**

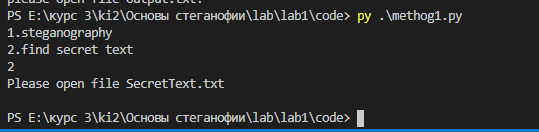
В файле methog1.py происходит посимвольное считывание исходного контейнера и посимвольная запись в файл. Шифрование и извлечение информации происходит описанным выше способом. В моем случае была реализована замена букв с и р .

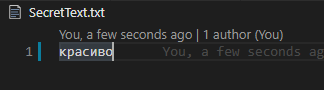
**File output.txt**

A screenshot of a social media post

Description automatically generated

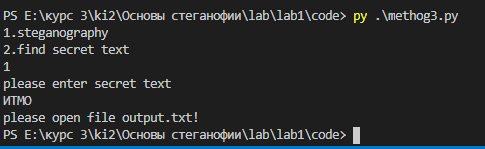
**File Secret.txt**





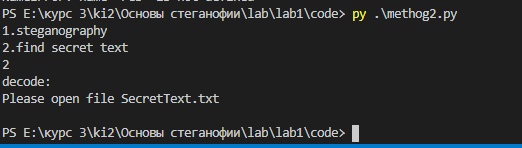
**Метод с использованием пробелов**

В файле methog2.py происходит построковое считывание исходного файла и последующая запись с добавлением или добавлением ”\t” в конце строки.

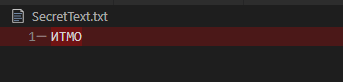


A screenshot of a social media post

Description automatically generated**File output.txt**

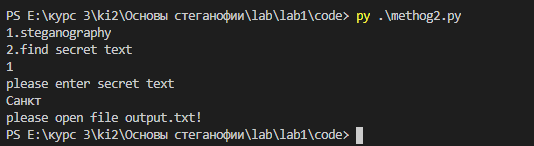


**File SecretText.txt**



**Метод с добавлением служебных символов**

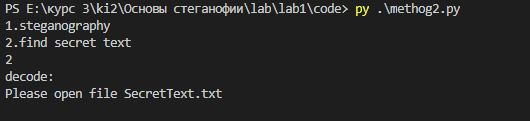
В файле methog3.py также реализовано посимвольное чтение и последующее посимвольная запись с добавление особеных символов



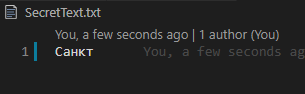
**File output.txt**

A screenshot of a social media post

Description automatically generated



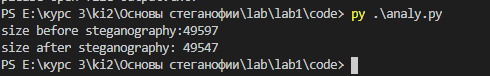
**File SecretText.txt**



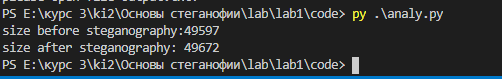
**Проведение экспертной оценки:**

Также мною был написан файл оценщик analy.py. Этот скрипт сравнивает два файла до вставки сообщения и после – выводит размер каждого файла.

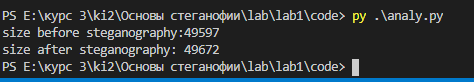
**Метод замены символов**



**Метод с добавлением служебных символов**



**Метод с добавлением служебных символов:**



Можно сделать вывод, что в первых двух случаях встраивание целесообразно, так как текст сам по себе весит внушительное количество байт, из-за чего сообщение скрыто, а вот в третьем случае – нет. Это произошло потому, что изменяются не только необходимые символы, но и все последующие, чтобы обеспечить визуальную незаметность сообщения в стегоконтейнере. К сожалению, размер файла при этом сильно меняется.

Если для просмотра сообщения пользоваться программой, которая не поддерживает отображение всех знаков, то во всех случаях обнаружить сообщение крайне сложно. Если не знать наверняка, что в тексте что-то содержится, то неподготовленный человек ничего не обнаружит. Если же использовать программу, которая позволит отображать все символы, то первый случай сразу же вызовет подозрения, во втором, если приглядываться, тоже можно понять, что что-то не так, а вот в третьем все не так просто. Создается впечатление, что это просто такой стиль письма. Но если иметь исходный текст и стегоконтейнер, полученный с помощью третьего метода, то можно с легкостью сделать вывод, что что-то было встроено, посмотрев на размеры файлов. Этот факт стал для меня решающим при проведении экспертной оценки – данный метод скрытия я не могу назвать успешным.

**Выводы:**

При выполнении данной лабораторной работы мною были изучены основные методы текстовой стеганографии. Я научился применять их и проводить последующую оценку их применению. По результатам работы были сделаны следующие выводы:

Метод замены символов одинакового начертания легко обнаружим при использовании многофункциональных текстовых редакторов, метод хвостовых пробелов трудно заметить даже при использовании таких редакторов, а реализованный мною метод добавления служебных символов вовсе не целесообразен, хоть и обнаружить его при отсутствии исходного текста довольно сложно.

Если я захочу в будущем поместить стего в текст, то, скорее всего, воспользуюсь метод знаков одинакового начертания ,который предполагает замену символа из текста на его аналог из другого языка, которые выглядит точно так же, но имеет другую кодировку

**Список использованной литературы:**

1. Стеганография & путешествия [Электронный ресурс] – URL: http://www.nestego.ru/2012/05/blog-post\_05.html

2. Стеганография & путешествия [Электронный ресурс] – URL: http://www.nestego.ru/2012/05/blog-post\_03.html

3. Основные положения стеганографии [Электронный ресурс] – URL:

http://citforum.ru/internet/securities/stegano.shtml

**Приложение**

Программа 1. Метод замены символов одинакового начертания.

def ChangeLetterToBinnary(letter):

    binary=''

    binary = '{0:08b}'.format(ord(letter)-1040)

    return binary

# transform string text to binary

def ChangeToBinary(text):

    binary = ''

    for i in range(len(text)):

        binary = binary + ChangeLetterToBinnary(text[i])

    return binary

# transform binary to string text

def ChangetoString(text):

    result = ''

    for i in range(len(text)//8):

        a = text[i\*8:(i+1)\*8] # every symbol using a 8bit-encode

        b = int(a,2)

        result += chr(b+1040)

    return result

def steganography(text,data):

    binary = list(ChangeToBinary(text))

    data = list(data)

    index = -1

    for i in range(len(binary)):

        if binary[i] == '0':

            index = data.index('с',index+1) # find index of symbol c in russian

            data[index] = 'c'               # change to c in english

        elif binary[i] == '1':

            index = data.index('р',index+1) # find index of symbol p in russian

            data[index] =  'p'              # change to p in english

    return ''.join(data)

def findSecretText(data):

    result = ''

    index = -1

    while True:

        index\_c = data.find('c',index+1)

        index\_p = data.find('p',index+1)

        if (index\_c == -1) and (index\_p == -1): break

        if ((index\_c < index\_p) and (index\_c != -1))or(index\_p == -1):

            index = index\_c

            result += '0'  # add bit 0 to result

        elif index\_p != -1:

            index = index\_p

            result += '1'  # add bit 1 to result

    return ChangetoString(result)

# ---------------------------------main programme--------------------------#

option = int(input('1.steganography \n2.find secret text\n'))

if option==1:

    file\_in = open('orginal\_document.txt',mode = 'r',encoding='UTF-8')

    file\_out = open('output.txt',mode = 'w',encoding='UTF-8')

    data = file\_in.read() # read data from orginal\_document

    text = input('please enter secret text \n') # input the text, that we must encode

    cipher\_text = steganography(text,data)

    file\_out.write(cipher\_text)

    print('please open file output.txt!')

    file\_in.close()

    file\_out.close()

elif option==2:

    file\_encode = open('output.txt',mode = 'r',encoding='UTF-8')

    data = file\_encode.read()

    file\_secret = open('SecretText.txt',mode = 'w',encoding='UTF-8')

    text\_secret=findSecretText(data)

    file\_secret.write(text\_secret)

    print('Please open file SecretText.txt\n')

    file\_secret.close()

    file\_encode.close()

Программа 2. Метод хвостовых пробелов.

def ChangeLetterToBinnary(letter):

    binary=''

    binary = '{0:08b}'.format(ord(letter)-1040)

    return binary

# transform string text to binary

def ChangeToBinary(text):

    binary = ''

    for i in range(len(text)):

       binary = binary + ChangeLetterToBinnary(text[i])

    return binary

# transform binary to string text

def ChangetoString(text):

    result = ''

    for i in range(len(text)//8):

        a = text[i\*8:(i+1)\*8] # every symbol using a 8bit-encode

        b = int(a,2)

        result += chr(b+1040)

    return result

def steganography(text,data):

    binary = list(ChangeToBinary(text))

    data = list(data)

    index = -1

    for i in range(len(binary)):

        if binary[i] == '0':

            index = data.index('\n',index+1) # find index of endline

            data[index] = ' \n'               # we add 1 spacebar right before endline

        elif binary[i] == '1':

            index = data.index('\n',index+1) # find index of endline

            data[index] =  '\t\n'              # we add 1 tab right before endline

    return ''.join(data)

#find secret text

def findSecretText(data):

    result = ''

    index = -1

    while True:

        index1 = data.find(' \n',index+1)

        index2 = data.find('\t\n',index+1)

        if (index1 == -1) and (index2 == -1): break

        if ((index1 < index2) and (index1 != -1))or(index2 == -1):

            index = index1

            result += '0'  # add bit 0 to result

        elif index2 != -1:

            index = index2

            result += '1'  # add bit 1 to result

    return ChangetoString(result)

# ---------------------------------main programme--------------------------#

option = int(input('1.steganography\n2.find secret text\n'))

if option==1:

    file\_in = open('orginal\_document.txt',mode = 'r',encoding='UTF-8')

    file\_out = open('output.txt',mode = 'w',encoding='UTF-8')

    data = file\_in.read()

    text = input('please enter secret text \n') # input the text, that we must encode

    file\_out.write( steganography(text,data))

    print('please open file output.txt!')

    file\_in.close()

    file\_out.close()

elif option==2:

    file\_encode = open('output.txt',mode = 'r',encoding='UTF-8')

    data = file\_encode.read()

    print('decode:')

    file\_secret = open('SecretText.txt',mode = 'w',encoding='UTF-8')

    text\_secret=findSecretText(data)

    file\_secret.write(text\_secret)

    print('Please open file SecretText.txt\n')

    file\_secret.close()

    file\_encode.close()

Программа 3. Добавление служебных символов.

def ChangeLetterToBinnary(letter):

    binary=''

    binary = '{0:08b}'.format(ord(letter)-1040)

    return binary

# transform string text to binary

def ChangeToBinary(text):

    binary = ''

    for i in range(len(text)):

        binary = binary + ChangeLetterToBinnary(text[i])

    return binary

# transform binary to string text

def ChangetoString(text):

    result = ''

    for i in range(len(text)//8):

        a = text[i\*8:(i+1)\*8] # every symbol using a 8bit-encode

        b = int(a,2)

        result += chr(b+1040)

    return result

def steganography(text,data):

    binary = list(ChangeToBinary(text))

    data = list(data)

    index = -1

    for i in range(len(binary)):

        if binary[i] == '0':

            index = data.index('\n',index+1) # find index of endline

            data[index] = chr(127)+'\n'        # we add a special character (that don't show when we open document)

        elif binary[i] == '1':

            index = data.index('\n',index+1) # find index of endline

            data[index] =  chr(8196)+'\n'              # we add a special character (that don't show when we open document)

    return ''.join(data)

def finSecretText(data):

    result = ''

    index = -1

    while True:

        index1 = data.find( chr(127)+'\n',index+1)

        index2 = data.find(chr(8196)+'\n' ,index+1)

        if (index1 == -1) and (index2 == -1): break

        if ((index1 < index2) and (index1 != -1))or(index2 == -1):

            index = index1

            result += '0'  # add bit 0 to result

        elif index2 != -1:

            index = index2

            result += '1'  # add bit 1 to result

    return ChangetoString(result)

# ---------------------------------main programme--------------------------#

option = int(input('1.steganography\n2.find secret text\n'))

if option==1:

    file\_in = open('orginal\_document.txt',mode = 'r',encoding='UTF-8')

    file\_out = open('output.txt',mode = 'w',encoding='UTF-8')

    data = file\_in.read() # read data from container

    text = input('please enter secret text \n') # input the text, that we must encode

    file\_out.write( steganography(text,data))

    print('please open file output.txt!')

    file\_in.close()

    file\_out.close()

elif option==2:

    file\_encode = open('output.txt',mode = 'r',encoding='UTF-8')

    data = file\_encode.read()

    print('decode:')

    file\_secret = open('SecretText.txt',mode = 'w',encoding='UTF-8')

    text\_secret=finSecretText(data)

    file\_secret.write(text\_secret)

    print('Please open file SecretText.txt\n')

    file\_secret.close()

    file\_encode.close()

Програм4: analy.py

from os.path import getsize

size\_before=getsize('orginal\_document.txt')

size\_after=getsize('output.txt')

print(f'size before steganography:{size\_before}')

print(f'size after steganography: {size\_after}')