**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ
ТЕХНОЛОГИЙ

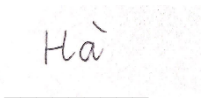
Основы стеганографии

# ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

«Основы текстовой стеганографии»

Работу выполнил:

Студент группы №3349
Данг Х.Х.



Работу проверил:

Давыдов В. В. _____

Санкт-Петербург

2020

## Цель работы:

- ✓ Научиться работать с текстовыми алгоритмами стеганографии.
- ✓ Исследовать 3 метода:
  - Метод замены символов (метод знаков одинакового начертания);
  - Метод с использованием пробелов (метод хвостовых пробелов);
  - Метод с добавлением служебных символов.

## Теоретическая часть:

**Стеганография** — это наука о скрытой передаче информации путём сохранения в тайне самого факта передачи. Главная задача сделать так, чтобы человек не подозревал, что внутри передаваемой информации, не представляющей внешне абсолютно никакой ценности, содержится скрытая ценная информация. Тем самым стеганография позволяет передавать секретную информацию через открытые каналы, скрывая сам факт её передачи. Криптография защищает сообщение, делая его бесполезным в случае перехвата, а стеганография стремится сделать саму передачу сообщения скрытой. Криптография и стеганография могут применяться вместе: тогда сообщение сначала шифруется, а потом скрытно передается. Если применять криптографию без стеганографии, то остается риск, что наблюдатель, перехвативший сообщение, силой заставит отправителя или получателя его расшифровать.

В данной лабораторной работе были рассмотрены следующие 3 алгоритма:

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

Метод знаков одинакового начертания символы предполагает подмену (бит 1) или ее отсутствие (бит 0) одной графемы другой, имеющей такой же вид начертания. Например, латинская “o” на кириллическую “о”. В моем случае была реализована замена букв о, а и р.

Метод хвостовых пробелов хвостовых пробелов предполагает дописывание (бит 1) или его отсутствие (бит 0) или дописывание одного или двух пробелов в конце строки. Мною был реализован алгоритм дописывания/отсутствие.

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: рассказ Украденные мощи (Автор: Стас Сенькин)

(Украденные мощи http://afon-ru.com/Stas-Senkin-Ukradennye-moshi-rasskaz-Stanislava-Senkina-Veru-my-dolzhny-pri-obresti-vystradat-poteryat-i-najti-vnov-uzhe-navsegd)

Язык кодирования: Python

Для каждого метода был написан отдельный файл:

**Общий файл** obshi.py, где хранятся функции, используемые в каждом из методов.

Файл содержит массив text, в котором хранятся символы, которые можно зашифровать.

- Файл zamena.py, отвечающий за первый метод
- Файл probel.py, отвечающий за первый метод
- Файл simvol.py, отвечающий за третий метод

Кажды из скриптов принимает ключи и аргументы.

- Ключ “-e”: шифрования.
- Ключ “-t” : сообщение, которое нужно зашифровать.
- Ключ “-d” : дешифрования

В режиме шифрования нужно подавать также аргументы – название одного или двух файлов – случае, если файл один, происходит запись в этот же файл; если файлов два – один файл становится контейнером, а во второй происходит запись.. Подается совместно с аргументом – названием файла, откуда нужно считывать информацию.

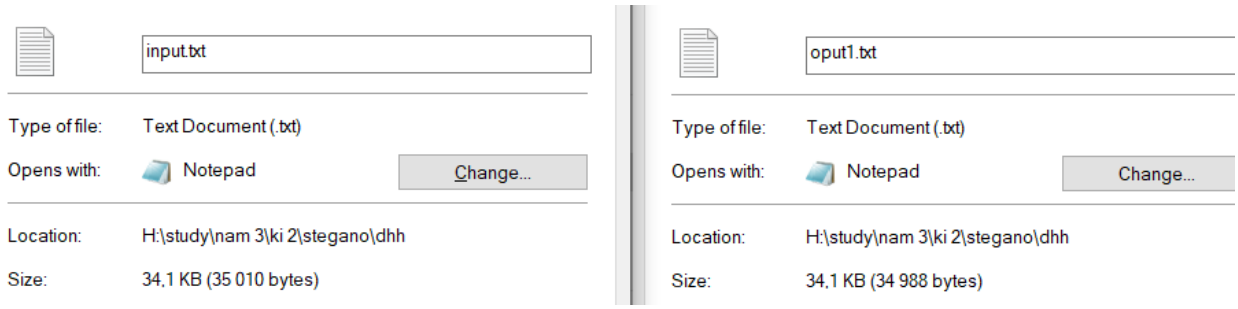
### 1. Метод замены сиволов:

```
H:\study\nam 3\ki 2\stegano\dhh>py zamena.py -e -t "прекрасно" input.txt oput1.txt

H:\study\nam 3\ki 2\stegano\dhh>py zamena.py -d oput1.txt
прекрасно
```

Извлеченное сообщение: «прекрасно»

После ширования, размеры файлов изменялся: (показаны в следующий рисунок)




Размер файла input.txt: 35010 bytes

Размер файла oput1.txt: 34988 bytes (уменьшен: 22bytes)

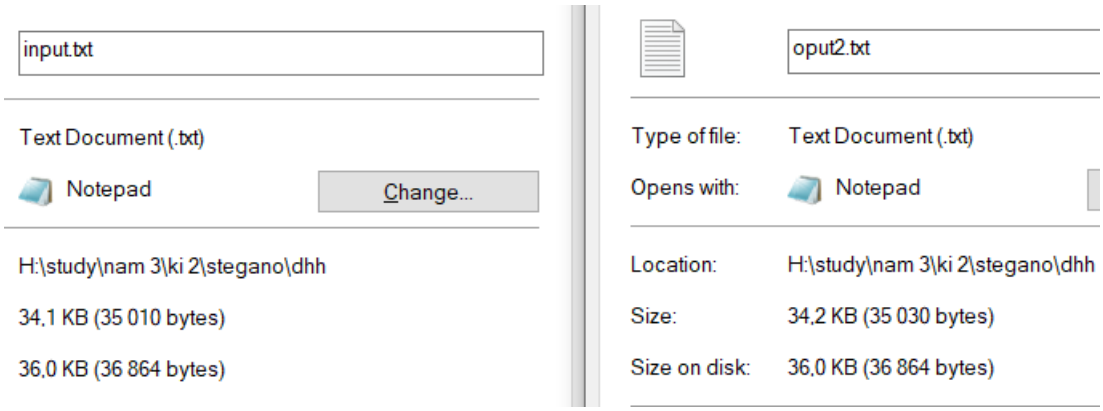
## 2. Метод с использованием пробелов (метод хвостовых пробелов):

```
H:\study\nam 3\ki 2\stegano\dhh>py probel.py -e -t "прекрасно" input.txt oput2.txt

H:\study\nam 3\ki 2\stegano\dhh>py probel.py -d oput2.txt
прекрасно
```

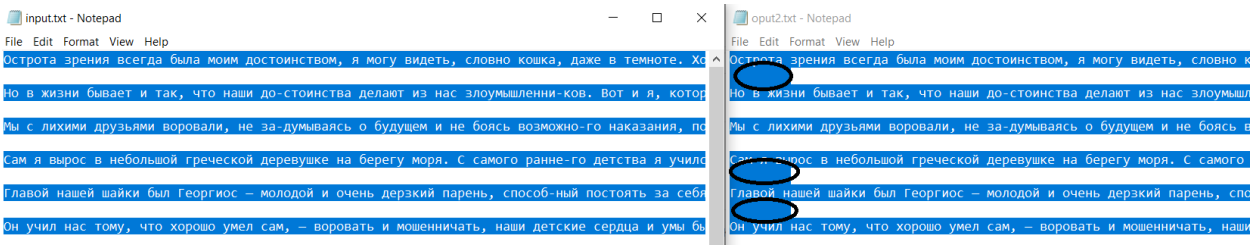
Извлеченное сообщение: «прекрасно»

После ширования, размеры файлов изменялся: (показаны в следующий рисунок)




Размер файла input.txt: 35010 bytes

Размер файла oput2.txt: 35030 bytes (уменьшен: 20bytes)




файл input.txt и файл oput2.txt (после ширования), пробелы появляются между строками

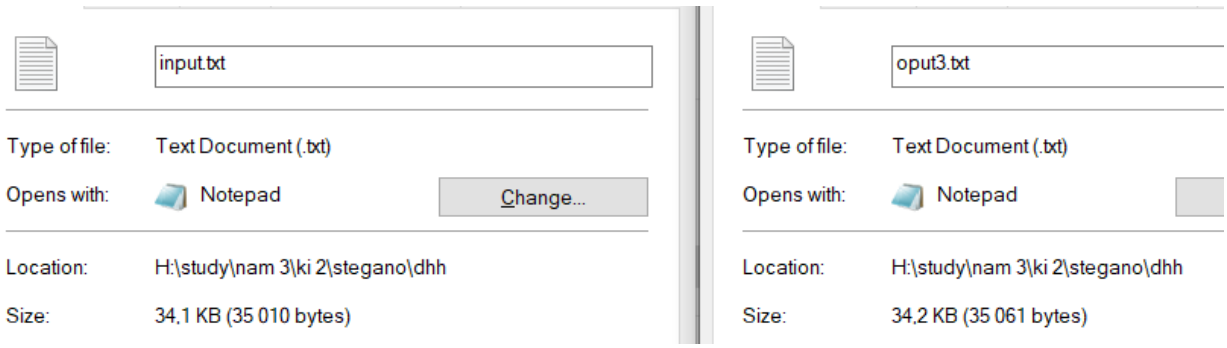
## 3. Метод с использованием служебных символов:

```
H:\study\nam 3\ki 2\stegano\dhh>py simvol.py -e -t "прекрасно" input.txt oput3.txt

H:\study\nam 3\ki 2\stegano\dhh>py simvol.py -d oput3.txt
прекрасно
```

Извлеченное сообщение: «прекрасно»

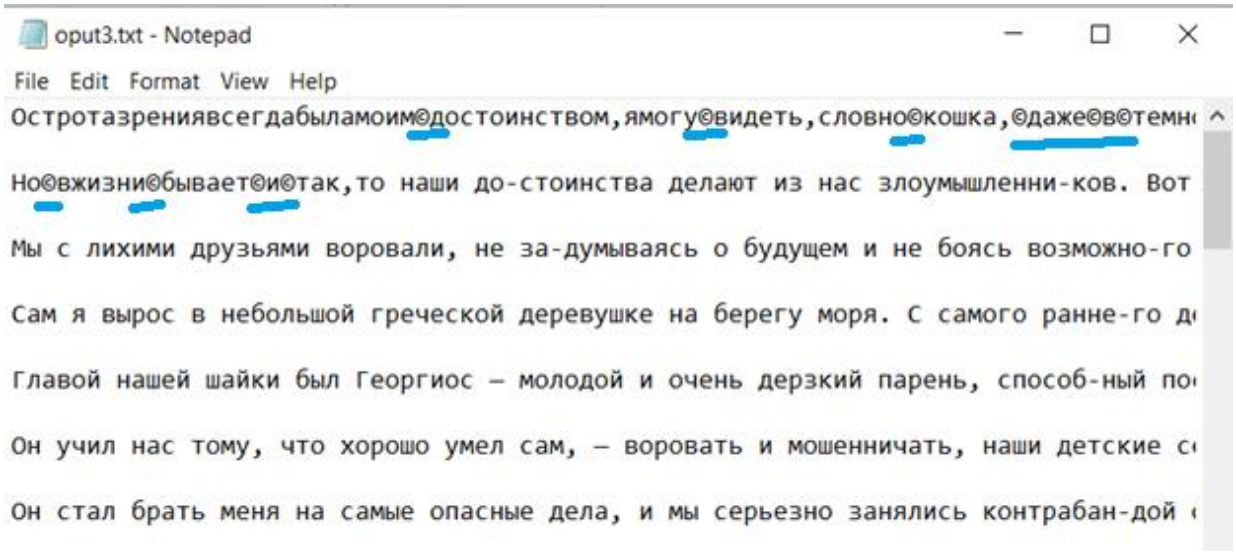
После ширования, размеры файлов изменялся: (показаны в следующий рисунок)




Размер файла input.txt: 35010 bytes

Размер файла oput3.txt: 35061 bytes (увеличился 51bytes)

Подмена одних служебных символов на другие (показаны в файле oput3.txt)



## Выводы:

В ходе данной лабораторной работы мною были изучены основы текстовой стеганографии. После работы, в заключение можно сделать вывод: метод хвостовых пробелов является самым целесообразным, так как длина сообщения, которое можно вписать достаточно велика, и при этом зрительно отличить пробел от символа ‘\t’ невозможно в большинстве редакторов.

## Список использованной литературы


Стеганографияhttps://urbanculture.in/%D0%A1%D1%82%D0%B5%D0%B3%D0%B0%D0%BD%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F

Текстовая стеганография: Метод знаков одинакового начертания [Электронный ресурс]: 2012 г. URL: http://www.nestego.ru/2012/05/blog-post_05.html

Стеганография в XXI веке. Цели. Практическое применение. Актуальность [Электронный ресурс]: 2015г. URL: https://habr.com/en/post/253045/

Текстовая стеганография: Метод хвостовых пробелов [Электронный ресурс]. – Электрон. дан. – URL: http://www.nestego.ru/2012/05/blog-post_11.html (дата обращения: 09.04.2020)

# Приложение

Файл «obshi.py»

```
import sys
import math
from getopt import GetoptError,getopt

text = [chr(i) for i in range(ord('а'),ord('я')+1)]

bits_per_lt = 5;

def hide_letter(letter):
    index = text.index(letter)
    return bin(index)[2:].zfill(bits_per_lt)

def decode_fileo(sequence):
    return text[int(sequence,2)]

def get_opts():
    opts = {'max-len': 8}
    options, args = getopt(sys.argv[1:], 't:ed')
    for key, value in options:
        opts[key.replace('-', '')] = value
    if 'e' in opts.keys() and 't' not in opts.keys():
        raise GetoptError('Error!! Please add -t in the righ of  -e!')
    return opts, args

def encode(text, bits_to_len=None):
    bits_to_len = 8 if bits_to_len is None else bits_to_len
    hided_text = ''
    lenghth = len(text)
    for letter in text:
        hided_text += hide_letter(letter)
    return bin(lenghth)[2:].zfill(bits_to_len) + hided_text

def decode(text, bits_to_len=None):
    bits_to_len = 8 if bits_to_len is None else bits_to_len
    size = int(text[:bits_to_len], 2)
    text = text[bits_to_len:]
    decoded_text = ''
    for i in range(size):
        decoded_text += decode_fileo(text[i * bits_per_lt: (i+1)*bits_per_lt])
    return decoded_text
```

Файл «zamena.py»

```
import os
from obshi import encode, decode, get_opts

opts,args = get_opts()

if 'e' in opts.keys():
    file_o = args[1]
    with open(args[0], 'r', encoding='UTF-8') as file_in,\
            open(file_o,'w',encoding='UTF-8') as file_out:
        hided_text = encode(opts['t'])
        index = 0
        letter = ''
        replace = {'o': 'o', 'p': 'p'} #rus -> eng
        while index < len(hided_text):
            while letter not in ['o','p']: #o,p rus
                file_out.write(letter)
                letter = file_in.read(1)
                if not letter and index < len(hided_text):
                    raise IOError('Error!! Please try with longer file input!')
            if hided_text[index] == '1':
                letter = replace[letter]
            file_out.write(letter)
            letter = file_in.read(1)
            index += 1
        file_out.write(file_in.read())

elif 'd' in opts.keys():
    with open(args[0],'r',encoding='UTF-8') as file_in:
        text = {
            'o': '0', 'p': '0', #rus
            'o': '1', 'p': '1', #eng
        }
        hided_text =''.join([text[x] if x in text.keys() else '' for x in file_i
n.read()])
        print(decode(hided_text))
```

Файл «probel.py»

```
import os
from obshi import encode,decode,get_opts

opts,args = get_opts()

if 'e' in opts.keys():
    file_o = args[1]
    with open(args[0],'r', encoding='UTF-8') as file_in, \
            open(file_o,'w', encoding='UTF-8') as file_out:
        hided_text = encode(opts['t'], bits_to_len=5)
        index = 0
        while index < len(hided_text):
            line = file_in.readline()
            if not line:
                raise IOError('Error!! Please try with longer file input!')
            if hided_text[index] == '1':
                file_out.write(line.replace('\n' ,'\t\n'))
            else:
                file_out.write(line)
            index += 1
        file_out.write(file_in.read())
    if len(args) == 1:
        os.rename(file_o, args[0])
elif 'd' in opts.keys():
    with open(args[0],'r',encoding='UTF-8') as file_in:
        hided_text = ''.join(['1' if '\t\n' in line else '0' for line in file_in
.readlines()])
        print(decode(hided_text, bits_to_len=5))
```

Файл «simvol.py»

```
import os
from obshi import encode,decode,get_opts

opts, args = get_opts()

if 'e' in opts.keys():
    file_o = args[1]
    with open(args[0],'r', encoding='UTF-8') as file_in, \
            open(file_o,'w',encoding='UTF-8') as file_out:
        hided_text = encode(opts['t'])
        index = 0
        letter = ''
        while index < len(hided_text):
            while letter not in [' ','\0']:
                file_out.write(letter)
                letter = file_in.read(1)
                if not letter and index < len(hided_text):
                    raise IOError('Error!! Please try with longer file input!')
            if hided_text[index] == '1':
                letter = chr(169)
            else:
                letter = chr(137)
            file_out.write(letter)
            letter = file_in.read(1)
            index += 1
        file_out.write(file_in.read())
    if len(args) == 1:
        os.rename(file_o,args[0])
elif 'd' in opts.keys():
    with open(args[0],'r',encoding='UTF-8') as file_in:
        text = {chr(137): '0', chr(169): '1'}
        hided_text=''.join([text[x] if x in text.keys() else '' for x in file_in
.read()])
        print(decode(hided_text))
```