**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет безопасности информационных технологий**

**Лабораторная работа №1**

**По теме: Основы текстовой стеганографии**

**Выполнил:**

Студент гр. N3349

Ле Тиен Зунг

**Проверил:**

Давыдов Вадим Валерьевич



Санкт-Петербург

2020г.

1. **Цель работы**

Целью данной лабораторной работы :

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

1. **Теория**

Стеганография, использующая текстовые контейнеры для скрытия данных, называется текстовой. При скрытии информации используются допущения при расположении и количестве символов в тексте, не учитываемые при прочтении человеком и компьютерном анализе текстового файла. Это может быть дополнительное количество пробелов и знаков табуляции в разных частях строки, чередование некоторых не учитываемых служебных символов, больших и маленьких букв, букв из разных алфавитов, но похоже выглядящих. К методам текстовой стеганографии относят: форматирование, изменение порядка следования маркеров конца строки, метод хвостовых пробелов, метод знаков одинакового начертания и изменения кода пробела.

Несмотря на простоту реализации текстовой стеганографии и её возможное широкое распространение, в настоящее время практически не реализованы методики её выявления. Удивление вызывает тот факт, что из автоматических методов текстовой стеганографии в открытой литературе упомянут только один: форматирование (то есть выравнивание) текста с помощью пробелов. При этом, среди распространенных программных средств, реализующих методы текстовой стеганографии, имеются отечественные разработки, практикующие метод знаков одинакового начертания.

В лабораторной работе рассматриваются три метода сокрытия информации:

* метод знаков одинакового начертания
* метод хвостовых пробелов
* добавление служебных символов.

Метод знаков одинакового начертания предполагает замену символа из текста на его аналог из другого языка, которые выглядит точно так же, но имеет другую кодировку. Каждая такая буква будет обозначать единичный или нулевой бит. Единственная проблема – если противник пользуется для просмотра текстовых файлов программой Microsoft Office Word или Notepad ,другим средством с автоматической проверкой орфографии, то благодаря ей он сможет быстро заметить подозрительные замены и заподозрить наличие стего в файле.

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.

            При сокрытии стего текстовый файл считывается построчно. Из конца строки удаляются все пробельный символы (пробелы, знаки табуляции, символы возврата каретки и новой строки), а затем в зависимости от значения текущего бита стеганосообщения, представленного в двоичном виде, принимается решение о дописывании в конец строки одного пробела. Преобразованная таким образом строка записывается в файл-результат.

Добавление служебных символов же предполагает встраивание в текст символов, которые мы используем в повседневной жизни, например запятые, тире, двоеточия и так далее, а также некоторых непечатных символов, которые невозможно увидеть.

Каждый из данных методов имеет свои преимущества и недостатки.

1. **Практика**

Выпонение вышеописанные методы, написано 3 программы, листинг которых можно найти в приложении к данному отчету, которое приведено после списка литературы.

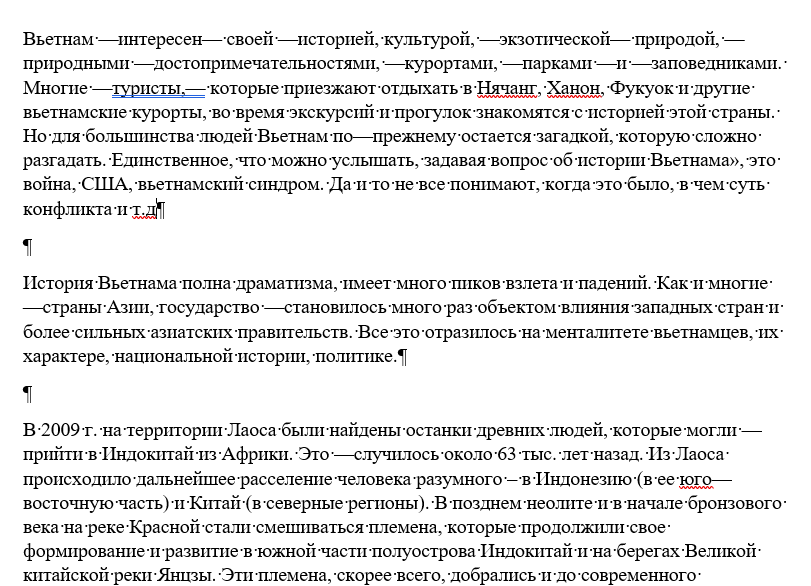
Для написания программ использовался язык Python на Visual Code

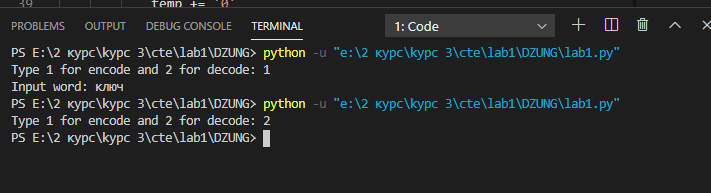
Схема принципу: в начале мы переводим наше сообщение, которое хотим спрятать, в двоичный код, и сохраняем его в строке, после чего начинаем посимвольно копировать исходный текст в новый файл, который станет стегоконтейнером. Как только в исходном тексте встречается символ, который следует заменить, программа вставляет вместо него в стегоконтейнер нужный нам символ, означающий единичный или нулевой бит, после чего продолжает свою работу. Так, в первой программе русская «о» заменяется английской, и это означает 1. Русская «е» заменяется английской, и это означает 0. Во второй программе один пробел после знака абзаца равняется 0, 2 пробела равняются 1. В третьей программе знак короткого тире «–» перед длинным «—» означает 0, наоборот – 1.

После того, как наше сообщение полностью помещено в текст, программа прекращает посимвольное сравнение и просто вставляет оставшуюся часть текста в стегоконтейнер. Так, в тексте я изменяю все длинные тире «—», добавляя к ним обычное тире «–», а не только те, которые нужны для кодирования сообщения . Я использовал функцию random для определения порядка вставки (вставить короткий до длинного или же после). Сделано это для того, чтобы при визуальном осмотре стегоконтейнера не бросалось в глаза то, что в начале текста все тире гораздо длиннее тире в конце текста. Для того, чтобы программа понимала, где заканчивается сообщение и начинается случайная абракадабра, я использовал символ длинного пробела, вставляя его после последнего измененного знака тире. Таким образом, программа извлекает из стегоконтейнера стего верно. Далее мы можем видеть примеры работы моих программ:

* **Метод знаков одинакового начертания.**

Рисунок моего текста input.txt





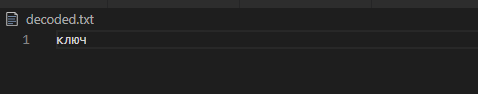
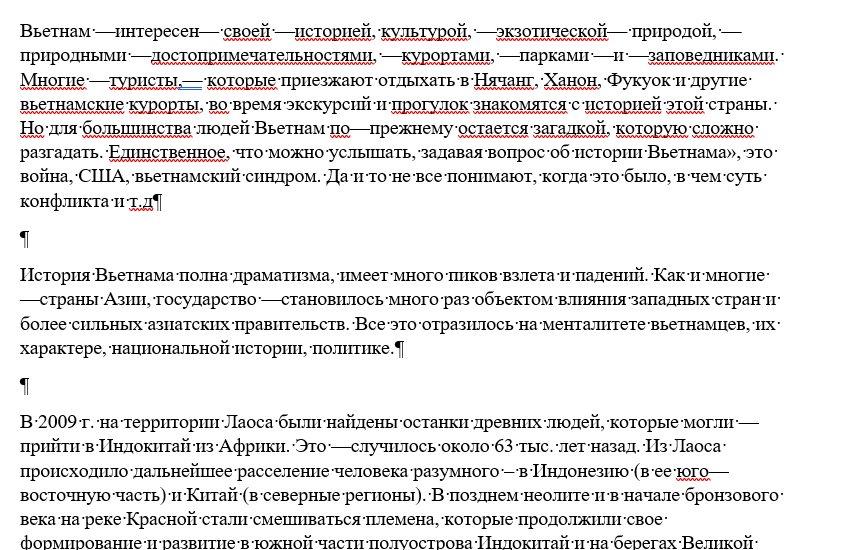
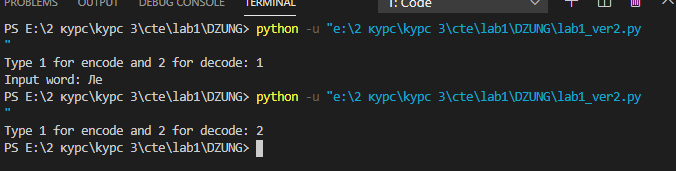


Рисунок моего текста output.txt



* **Метод хвостовых пробелов.**



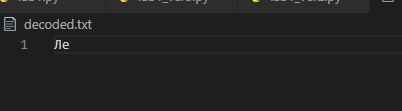
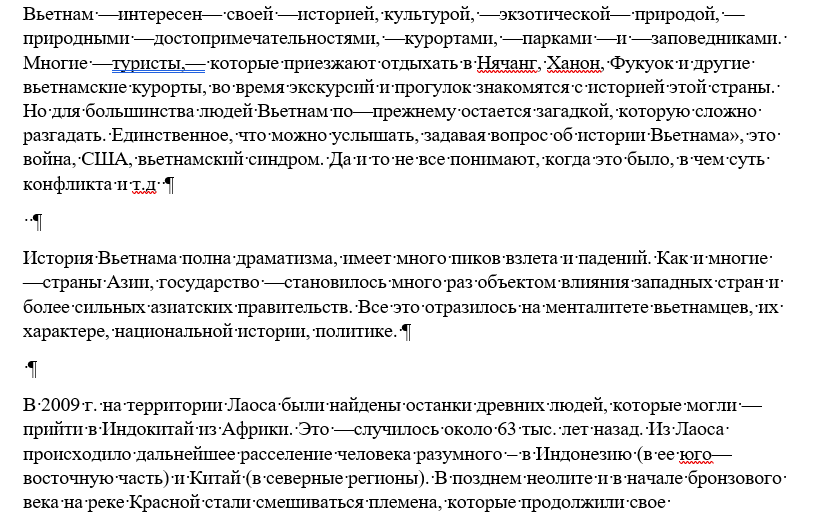
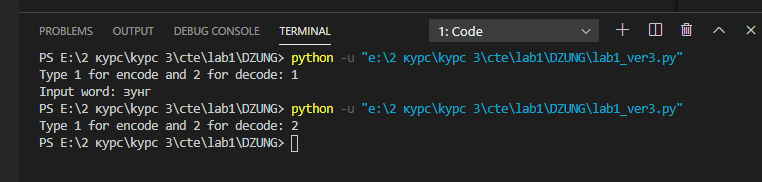


Рисунок output.txt



* **Добавление служебных символов.**



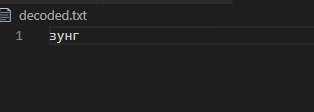
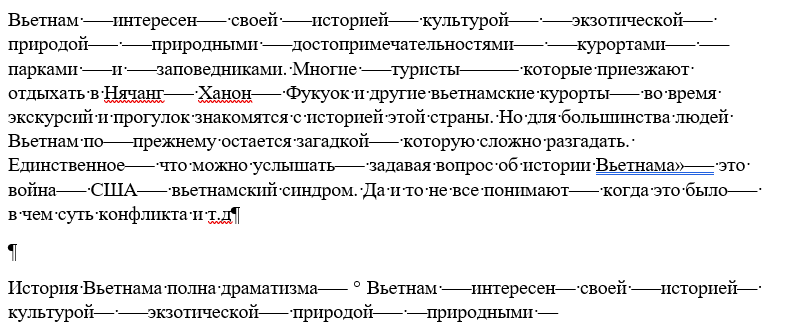
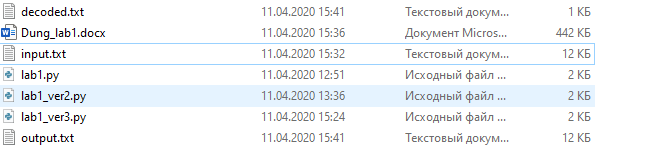


Рисунок output.txt

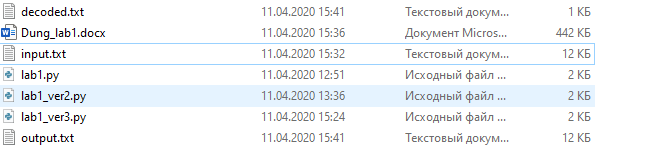


Программы работают корректно, обеспечивая встраивание стего в текст. Теперь необходимо выяснить целесообразность данного встраивания. Изначальный объем текста равен 12 КБ.

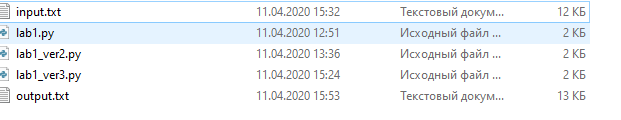
В первом случае файл размера не изменился – 12 КБ



Во втором случае файл размера не изменился – 12КБ



В третьем случае размер файла увеличился на 1КБ



В первых двух случаях встраивание целесообразно, из-за чего сообщение скрыто, а вот в третьем случае – нет хотя размер изменился 1kb .Если сначала файл большой поэтому размер много изменился .

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.

**Вывод**

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. Алгоритмы и практическая реализация [Электронный ресурс]. – Электрон. дан. – URL: [https://www.nestego.ru/2012/04/blog-post\_28.html](https://www.nestego.ru/2012/04/blog-post_28.html?m=0) (дата обращения: 08.04.2020) <http://ru-steganography.narod.ru/>
2. Стеганография & путешествия (http://www.nestego.ru/2012/05/blog-post\_03.html)
3. Стеганография & путешествия (<http://www.nestego.ru/2012/05/blog-post_05.html>)
4. Текстовая стеганография

([https://ru.wikipedia.org/wiki](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%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)

**Приложение**

* Метод замены символов одинакового начертания.

import codecs

#=============Программа 1:Метод замены символов одинакового начертания============

import codecs # модуль для открыта файла

#-----------------

def changesym(s, word): #encode symbol in file

    temp1 = ''

    temp2 = ''

    k = 0

    for i in range(len(word)): #range(start, stop[, step])

        temp1 += bin(ord(word[i])-848)[2::] # alphabet-> binany

    #русская о = 1, русская е = 0

    i = 0

    while k < len(temp1):

        if s[i] == 'о' and temp1[k] == '1':

            temp2 += 'o'

            k += 1

        elif s[i] == 'е' and temp1[k] == '0':

            temp2 += 'e'

            k += 1

        else:

            temp2 += s[i]

        i += 1

    temp2 += s[i::]

    f = codecs.open('output.txt', 'w', 'utf-8')

    f.write(temp2)

    f.close()

#-------------------------

def decode():

    f = codecs.open('output.txt', 'r', 'utf-8')

    s = f.read()

    f.close()

    temp = ''

    result = ''

    for i in range(len(s)):

        if s[i] == 'o':

            temp += '1'

        if s[i] == 'e':

            temp += '0'

    s1 = [temp[x:x+8] for x in range(0, len(temp), 8)]

    for i in s1:

        result += chr(int(i, 2)+848) #binany -> alphabet

    f = codecs.open('decoded.txt', 'w', 'utf-8')

    f.write(result)

    f.close()

#-------------------------------

f = codecs.open('input.txt', 'r', 'utf-8') #open file input.txt

s = f.read()

f.close()

opt = input('Type 1 for encode and 2 for decode: ')

if opt == '1':

    word = input('Input word: ')

    changesym(s, word)

elif opt == '2':

    decode()

* Метод хвостовых пробелов.

import codecs

def gap(s, word):

    temp1 = ''

    temp2 = ''

    for i in range(len(word)):

        temp1 += bin(ord(word[i])-848)[2::]

    i = 0

    k = 0

    #2 пробела = 1, 1 пробел = 0

    while k < len(temp1):

        if ord(s[i]) == 10 and temp1[k] == '1':

            temp2 += '  \n'

            k += 1

        elif ord(s[i]) == 10 and temp1[k] == '0':

            temp2 += ' \n'

            k += 1

        elif ord(s[i]) != 13: #chr(13) - \r -carriage return

            temp2 += s[i]

        i += 1

    temp2 += s[i::]

    f = codecs.open('output.txt', 'w', 'utf-8')

    f.write(temp2)

    f.close()

def decode():

    f = codecs.open('output.txt', 'r', 'utf-8')

    s = f.read()

    f.close()

    temp = ''

    result = ''

    for i in range(len(s)):

        if s[i] == '\n':

            if s[i-1] == ' ':

                if s[i-2] == ' ':

                    temp += '1'

                else:

                    temp += '0'

    s1 = [temp[x:x+8] for x in range(0, len(temp), 8)]

    for i in s1:

        result += chr(int(i, 2)+848)

    f = codecs.open('decoded.txt', 'w', 'utf-8')

    f.write(result)

    f.close()

f = codecs.open('input.txt', 'r', 'utf-8')

s = f.read()

f.close()

opt = input('Type 1 for encode and 2 for decode: ')

if opt == '1':

    word = input('Input word: ')

    gap(s, word)

elif opt == '2':

    decode()

* Добавление служебных символов.

import codecs

import random

def spec\_sym(s, word):

    temp1 = ''

    temp2 = ''

    for i in range(len(word)):

        temp1 += bin(ord(word[i])-848)[2::]

    i = 0

    k = 0

    #long dash = 1, short dash = 0

    while k < len(temp1):

        if s[i] == '—' and temp1[k] == '1':

            temp2 += '—–'

            k += 1

        elif s[i] == '—' and temp1[k] == '0':

            temp2 += '–—'

            k += 1

        elif ord(s[i]) != 13:

            temp2 += s[i]

        i += 1

    temp2 += chr(8195)

    j = i

    for j in range(len(s)):

        if s[j] == '—' and random.getrandbits(1) == 0:

            temp2 += '—–'

        elif s[j] == '—' and random.getrandbits(1) == 1:

            temp2 += '–—'

        else:

            temp2 += s[j]

    f = codecs.open('output.txt', 'w', 'utf-8')

    f.write(temp2)

    f.close()

def decode():

    f = codecs.open('output.txt', 'r', 'utf-8')

    s = f.read()

    f.close()

    tmp = ''

    result = ''

    i = 0

    k = s.find(chr(8195))

    while i < k:

        if s[i] == '—':

            tmp += '1'

            i += 2

        elif s[i] == '–':

            tmp += '0'

            i += 2

        else:

            i += 1

    s1 = [tmp[x:x+8] for x in range(0, len(tmp), 8)]

    for i in s1:

        result += chr(int(i, 2)+848)

    f = codecs.open('decoded.txt', 'w', 'utf-8')

    f.write(result)

    f.close()

f = codecs.open('input.txt', 'r', 'utf-8')

s = f.read()

f.close()

opt = input('Type 1 for encode and 2 for decode: ')

if opt == '1':

    word = input('Input word: ')

    spec\_sym(s, word)

elif opt == '2':

    decode()