**Министерство науки и высшего образования Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет безопасности информационных технологий**

**Дисциплина:**

**ОСНОВЫ СТЕГАНОГРАФИИ**

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1:

“Основы текстовой стеганографии ”

**Выполнил:**

Студент группы N3351

Нгуен Туан Ань

Подпись:

**Проверил:**

Давыдов Вадим Валерьевич

Санкт-Петербург, 2020 г.

**Цель работы**

Целью данной лабораторной работы :

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

**Теория**

Стеганография - это метод организации связи, который собственно скрывает само наличие связи. В отличие от криптографии, где неприятель точно может определить является ли передаваемое сообщение зашифрованным текстом, методы стеганографии позволяют встраивать секретные сообщения в безобидные послания так, чтобы невозможно было заподозрить существование встроенного тайного послания.

Слово "стеганография" в переводе с греческого буквально означает "тайнопись" (steganos - секрет, тайна; graphy - запись). К ней относится огромное множество секретных средств связи, таких как невидимые чернила, микрофотоснимки, условное расположение знаков, тайные каналы и средства связи на плавающих частотах и т. д.

Стеганография занимает свою нишу в обеспечении безопасности: она не заменяет, а дополняет криптографию. Сокрытие сообщения методами стеганографии значительно снижает вероятность обнаружения самого факта передачи сообщения. А если это сообщение к тому же зашифровано, то оно имеет еще один, дополнительный, уровень защиты.

В настоящее время в связи с бурным развитием вычислительной техники и новых каналов передачи информации появились новые стеганографические методы, в основе которых лежат особенности представления информации в компьютерных файлах, вычислительных сетях и т. п. Это дает нам возможность говорить о становлении нового направления - компьютерной стеганографии.

В лабораторной работе рассматриваются три метода сокрытия информации:

1. Метод знаков одинакового начертания
2. Метод хвостовых пробелов
3. Добавление служебных символов
4. 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.
5. 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.

В целом можно признать метод возможным для применения, особенно если использовать предварительное сжатие и шифрование стеганосообщения.

1. 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.

**Практика**

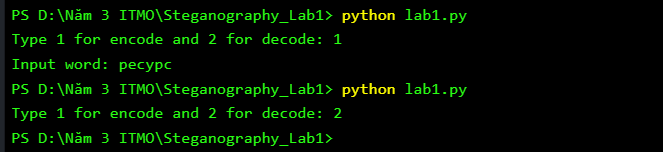
Для реализования таких методов, написал 3 программы, которые приведены после списка литературы.

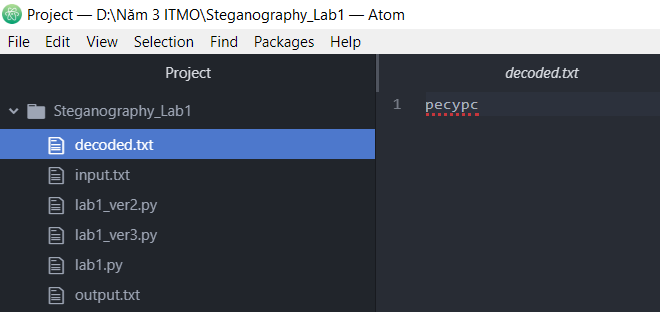
Для написания программ использовался язык Python 3, кодировался на Atom.

Использовал 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 «р» заменяется английской, и это означает 1. Русская «с» 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.

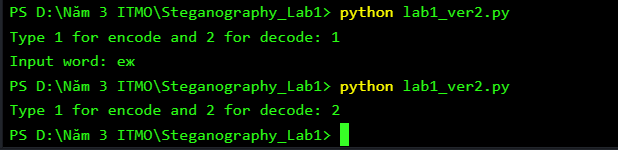
После того, как наше сообщение полностью помещено в текст, программа прекращает посимвольное сравнение и просто вставляет оставшуюся часть текста в стегоконтейнер. Так, в тексте я изменил все длинные тире «—», добавляя к ним обычное тире «–», а не только те, которые нужны для кодирования сообщения . 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:

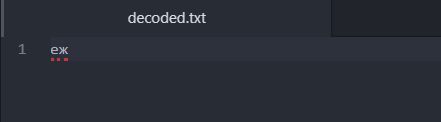
**Метод 1: Метод знаков одинакового начертания**

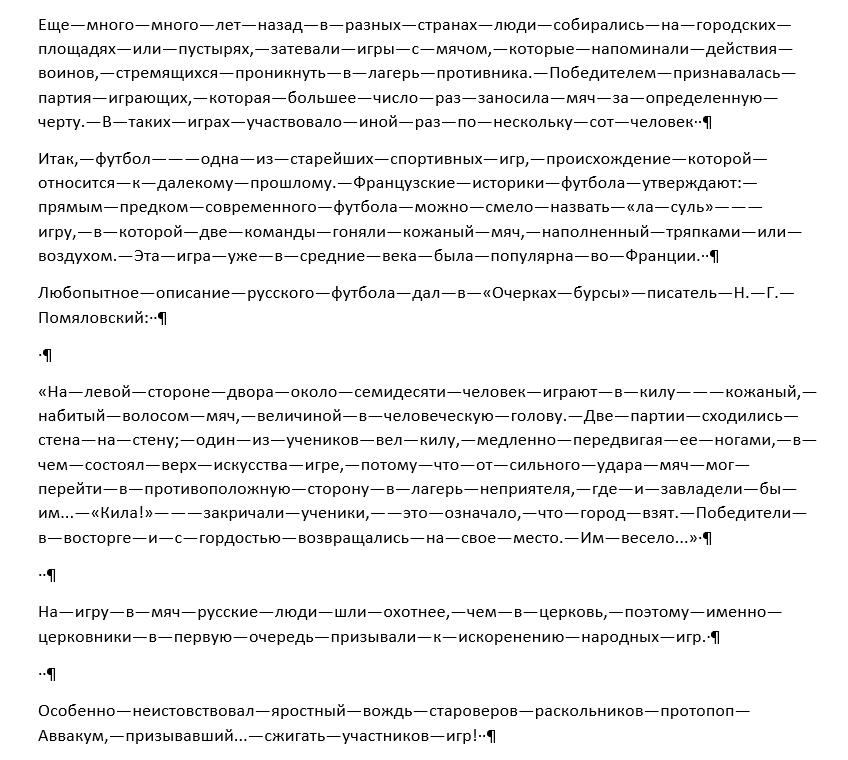




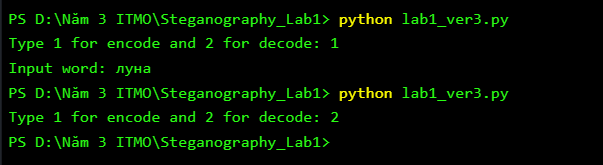
**Метод 2: Метод хвостовых пробелов**

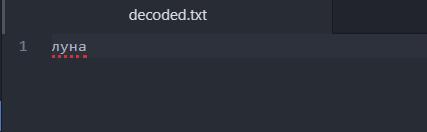






**Метод 3: Добавление служебных символов**

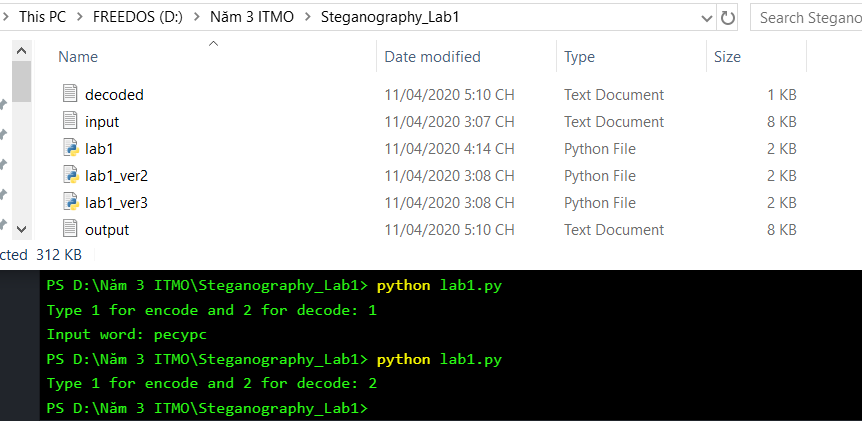




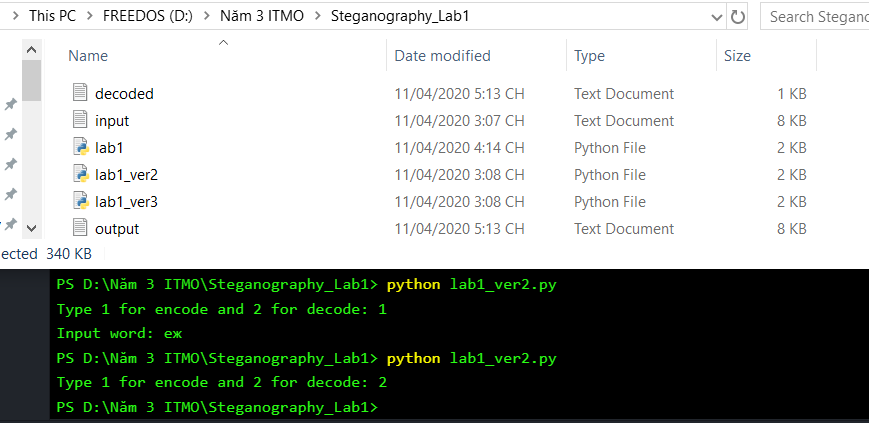


Мы видим, что программы работают корректно, обеспечивая встраивание стего в текст. Теперь необходимо выяснить целесообразность данного встраивания. Изначальный объем текста равен 8 КБ.

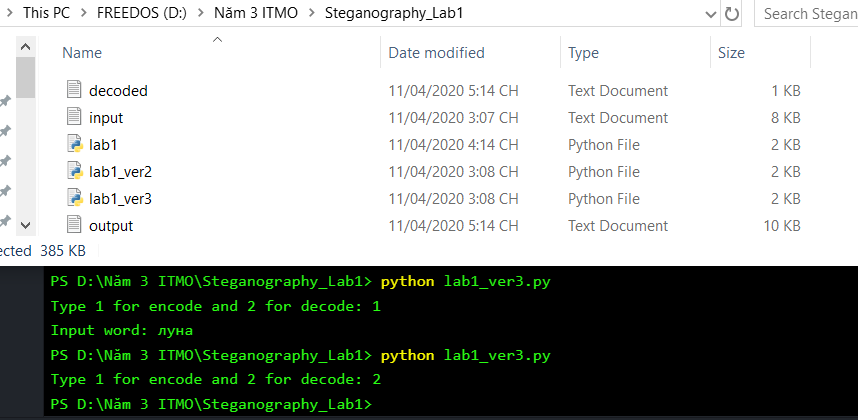
В первом случае размер не изменился – 8 КБ



Во втором случае размер также не изменился – 8 КБ



В третьем случае размер файла увеличился на 2 КБ.



Можно сделать вывод, что в первых двух случаях встраивание целесообразно, из-за чего сообщение скрыто, а вот в третьем случае – нет хотя размер изменился 2 Kb. Для использования данного метода следует либо отказываться от случайного распределения служебных символов после помещения сообщения в текст, либо использовать текст с меньшим количеством тире.

Если смотреть сообщения пользованием программой, которая не поддерживает отображение всех знаков (например, Блокнот), 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. 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 следует что, при проведении экспертной оценки – данный метод скрытия я не могу назвать успешным.

**Вывод**

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

Свойства контейнера должны быть модифицированы, чтобы изменение невозможно было выявить при визуальном контроле. Это требование определяет качество сокрытия внедряемого сообщения: для обеспечения беспрепятственного прохождения стегосообщения по каналу связи оно никоим образом не должно привлечь внимание атакующего.

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 методом 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.**

#Программа 1:Метод замены символов одинакового начертания

import codecs # модуль для открыта файла

def change\_sym(sym, word): #encode symbol in file

tmp1 = ''

tmp2 = ''

j = 0

for i in range(len(word)):

tmp1 += bin(ord(word[i])-848)[2::]

#русская р = 1, русская c = 0

i = 0

while j < len(tmp1):

if sym[i] == 'р' and tmp1[j] == '1':

tmp2 += 'p'

j += 1

elif sym[i] == 'с' and tmp1[j] == '0':

tmp2 += 'c'

j += 1

else:

tmp2 += sym[i]

i += 1

tmp2 += sym[i::]

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

f.write(tmp2)

f.close()

def decode():

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

sym = f.read()

f.close()

tmp = ''

res = ''

for i in range(len(sym)):

if sym[i] == 'p':

tmp += '1'

if sym[i] == 'c':

tmp += '0'

sym1 = [tmp[x:x+8] for x in range(0, len(tmp), 8)]

for i in sym1:

res += chr(int(i, 2)+848)

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

f.write(res)

f.close()

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

sym = f.read()

f.close()

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

if opt == '1':

word = input('Input word: ')

change\_sym(sym, word)

elif opt == '2':

decode()

**Программа 2. Метод хвостовых пробелов.**

import codecs

def probel(sym, word):

tmp1 = ''

tmp2 = ''

for i in range(len(word)):

tmp1 += bin(ord(word[i])-848)[2::]

i = 0

k = 0

#2 пробела = 1, 1 пробел = 0

while k < len(tmp1):

if ord(sym[i]) == 10 and tmp1[k] == '1':

tmp2 += ' \n'

k += 1

elif ord(sym[i]) == 10 and tmp1[k] == '0':

tmp2 += ' \n'

k += 1

elif ord(sym[i]) != 13:

tmp2 += sym[i]

i += 1

tmp2 += sym[i::]

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

f.write(tmp2)

f.close()

def decode():

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

sym = f.read()

f.close()

temp = ''

result = ''

for i in range(len(sym)):

if sym[i] == '\n':

if sym[i-1] == ' ':

if sym[i-2] == ' ':

temp += '1'

else:

temp += '0'

sym1 = [temp[x:x+8] for x in range(0, len(temp), 8)]

for i in sym1:

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')

sym = f.read()

f.close()

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

if opt == '1':

word = input('Input word: ')

probel(sym, word)

elif opt == '2':

decode()

**Программа 3. Добавление служебных символов**.

import codecs

import random

def sym\_spec(sym, word):

tmp1 = ''

tmp2 = ''

for i in range(len(word)):

tmp1 += bin(ord(word[i])-848)[2::]

i = 0

k = 0

#long dash = 1, short dash = 0

while k < len(tmp1):

if sym[i] == '—' and tmp1[k] == '1':

tmp2 += '—–'

k += 1

elif sym[i] == '—' and tmp1[k] == '0':

tmp2 += '–—'

k += 1

elif ord(sym[i]) != 13:

tmp2 += sym[i]

i += 1

tmp2 += chr(8195)

j = i

for j in range(len(sym)):

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

tmp2 += '—–'

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

tmp2 += '–—'

else:

tmp2 += sym[j]

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

f.write(tmp2)

f.close()

def decode():

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

sym = f.read()

f.close()

tmp = ''

result = ''

i = 0

k = sym.find(chr(8195))

while i < k:

if sym[i] == '—':

tmp += '1'

i += 2

elif sym[i] == '–':

tmp += '0'

i += 2

else:

i += 1

sym1 = [tmp[x:x+8] for x in range(0, len(tmp), 8)]

for i in sym1:

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')

sym = f.read()

f.close()

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

if opt == '1':

word = input('Input word: ')

sym\_spec(sym, word)

elif opt == '2':

decode()