ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»

Факультет безопасности информационных технологий

Дисциплина:

«Основы стеганографии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

«Основы текстовой стеганографии»

Выполнил:

Студент гр. N3352

Шарипов Ф.Р.

Проверил:

Давыдов В. В.

Санкт-Петербург

2020г.

Цель работы: Реализация простых методов встраивания информации в стегоконтейнер.

Теоретическая часть

Стеганография – это наука о способах передачи или хранения информации при условии сохранения в тайне самого факта наличия скрытой информации.

Основными стеганографическими понятиями являются сообщение и контейнер.

Сообщение – это секретная информация, наличие которой необходимо скрыть.

Контейнер - некоторая информация, в частности файл, в которую можно внедрить другую информацию, не предназначенную для посторонних глаз.

Тестовая стеганография – стеганография, использующая текстовые контейнеры для скрытия данных.

Алгоритм встраивания-извлечения сообщения в простейшем случае состоит из двух основных этапов:

- Встраивание в стеганокодере секретного сообщения в контейнер-оригинал.
- Обнаружение (выделение) в стеганодетекторе (декодере) скрытого зашифрованного сообщения из контейнера-результата .

Процесс встраивания:

1. Необходимо привести сообщение к двоичному виду.
2. Выбрав метод встраивания, решить, как будут кодироваться 1 и 0.

    2.1 Методе замены русских букв на похожие английские «аеосрхуАЕОСРХТ»;
    2.2 Метод со встраиванием пробелов;
    2.3 Метод встраивания специальных;

3. Встроить секретное сообщение в стегоконтейнер.

Процесс извлечения:

1. Считывание символов из контейнера-результата и запись двоичной последовательности.
2. Преобразовать двоичную последовательность в секретное сообщение.

Практическая часть

Часть 1

Программная часть стеганокодера реализовано на языке C++.

Метод прямой замены символов:

Русские буквы «е» и «с», заменяются на аналогичные английские буквы, и будут кодироваться 1 и 0 соответственно. Для реализации этого метода была создана функция **embed(string &text, string s)**;

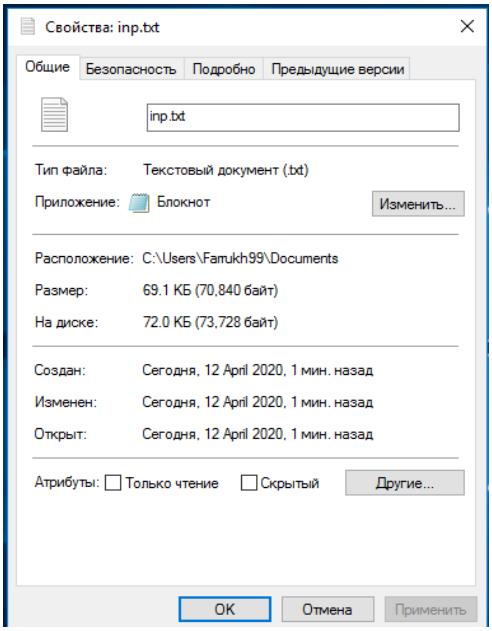
Метод с использованием дополнительных пробелов:

В этом методе будут добавляется пробелы в конец строки, один пробел будет кодироваться 1, а два пробела будет кодироваться 0. для реализации этого метода было создано функция **string addSpace(string s);**

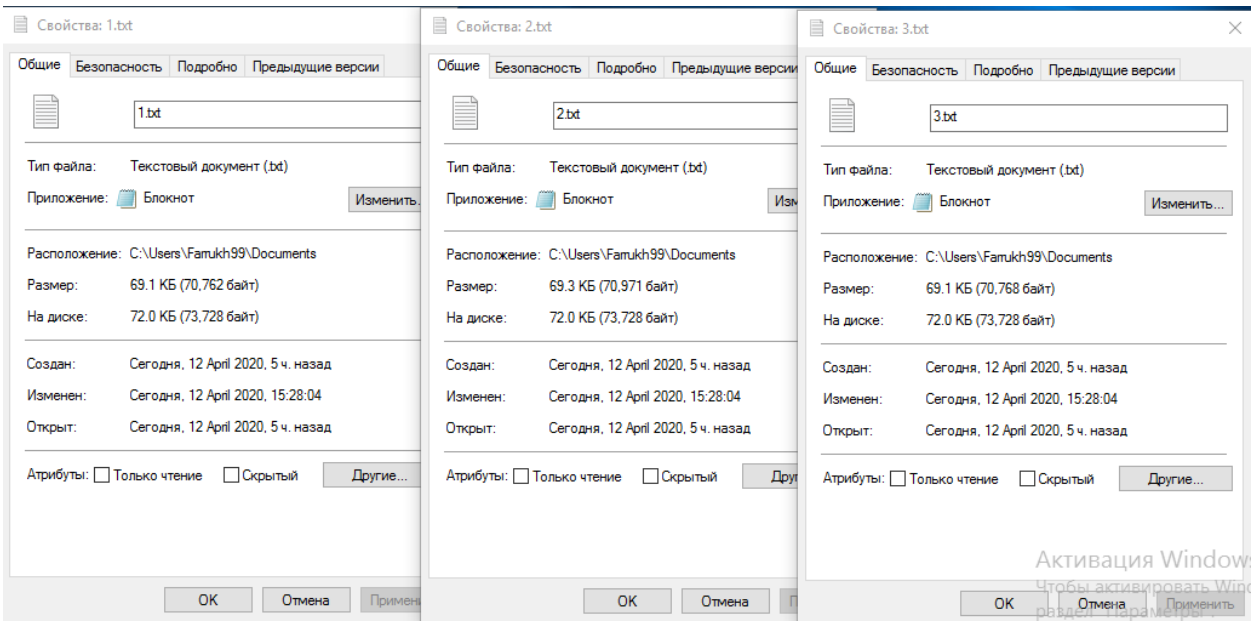
Добавление служебных символов:

Символы «–» (тире средней длины) и «…», которые будут заменять символы «-» (обычное тире) и «.» и кодировать 1 и 0 соответственно.

Программа будет встраивать сообщения «информация» в текстовый контейнер. В качестве текстового контейнера было использовано файл с размером 70840байт.



Здесь показана размер контейнера с начальным текстом



На этом скриншоте демонстрируются размеры всех контейнеров после обработки тремя методами

Часть 2

Каждая буква сообщения кодируется последовательностью из 16 символов (0 и 1), всего букв в сообщении 10, значить последовательность из 160 символа.

В первом методе 160 русских букв заменяются английскими буквами. Стегоконтейнер получился меньшим размером, так как русские буквы занимают 2 байта, а английские 1 байт. В данном методе размер стегоконтейнера составляет 70 762байт. В данном методе размер уменьшается несущественно, поэтому встраивание целесообразно.

Во втором методе размер стегоконтейнера составил 70 971байт. В этом методе размер стегоконтейнера увеличился на 131 байт, так как на каждую 0 добавляется два пробела, а на каждую 1 – пробел. В данном случае размер незначительно увеличивается, поэтому такое встраивание целесообразно. Минусом, является существование программ, автоматически исправляющих количество пробелов. Так же в случае печатной копии, невозможно извлечь скрываемое сообщение.

В методе служебных символов размер стегоконтейнера составляет 70768 байт. Здесь, размер стегоконтейнера уменьшился так как размер «.» и «-» составляет по одному баллу, а изначальные символы были по два балла.

Метод прямой замены символов внешне никак не распознается, но в случае проверки на алфавит, встраивание будет сразу обнаружено.

Добавление служебных символов также внешне распознать тяжело, тире на первый взгляд ничем не отличаются, на символ многоточия не обращаешь внимания, но если подсчитать все символы, и выяснится, что в тексте существуют тире разных длин и символ многоточия, встраивание будет обнаружено.

Проведя статистический анализ двух текстов разной величины до и после встраивания пробелов, можно сделать вывод о том, что процент пробелов до встраивания в больших текстах увеличивается, а процент пробелов после встраивания уменьшается. Можно судить о том, что в больших текстах сообщения будет заметить крайне тяжело, тем более сообщение может быть меньшей длинны, но в случае с маленьким текстом, заметить расхождение не составит труда.

Результат статического анализа:

Количество символов до встраивания пробелов	Процент пробелов до %	Количество символов после встраивания пробелов	Процент пробелов после %	Разница, %
38410	15,86	38607	16,29	0,43
57615	15,86	57812	16,15	0,34

Вывод

Из методов программного комплекса с использованием текстового контейнера, самым лучшим на мой взгляд это метод замены символов, т.к. некоторые русские и английские буквы обладают идентичным написанием и их невозможно обнаружить простому читателю.

Естественно, раскрыть любой из этих методов эксперту не составит труда. Все методы просты в раскрытии.

Литература:

1. Грибунин В.Г., Оков И.Н., Туринцев И.В. Цифровая стеганография. – М.: Солон-Пресс, 2002. – 272 с.
2. Зайцева А.В. Методика построения энтропийных стеганографических систем защиты сообщений в информационных сетях: автореф.дис. … к.т.н МГТУ им. Н.Э.Баумана, Москва, 2014.
3. Изычаев А.В., Сидоренко В.Г. Стегонографические методы защиты информации. Москва, 2017.

Листинг программы:

```
#include <bits/stdc++.h>
#include <stdlib.h>

using namespace std;

string mainFile="inp.txt";
string method1="1.txt";
string method2="2.txt";
string method3="3.txt";

void pars(string &s);
string addSpace(string s);
void AtoB(string &chiperText);
void embed(string &text, string s);
void replaceSymbol(string &text, string s);
void OutToFile(string text, string fileName);
```

```
int main()
{

    string text1,text2,text3,chiperText="информация";

    AtoB(chiperText);
    pars(text1);
    pars(text3);
    embed(text1, chiperText);//первый метод
    text2=addSpace(chiperText); // второй метод
    replaceSymbol(text3, chiperText);// третий метод
    OutToFile(text1,method1);
    OutToFile(text2,method2);
    OutToFile(text3,method3);
    cout<<chiperText<<"\n"<<chiperText.size();

}

void embed(string &text, string s)
{
    int k=0;

    for (int i=0;i<s.size();i++)
    {
        if (s[i]=='1')
        {
            k=text.find("e");
           if (k==-1){cout<<"Недостаточно текста для встраивания";}
            text.replace(k,2,"e");
```

```
        }
        else
        {
            k=text.find("c");
            if (k==-1){cout<<"Недостаточно текста для встраивания";}
            text.replace(k,2,"c");
        }
    }
}
string addSpace(string s)
{
    string a,text;
    int i=0;
    ifstream file(mainFile.c_str());
    while (!file.eof())
    {
        getline(file,a);
        if (i<=s.size())
        {
            if (s[i]=='1') a+=" ";
            else a+="  ";
            i++;
        }
        a+="\n";
        text+=a;
    }
    file.close();
    return text;
}
void pars(string &s)
```

```
{
    string a,text;
    ifstream file(mainFile.c_str());
    while (!file.eof())
    {
        getline(file,a);
        a+="\n";
        text+=a;
    }
    file.close();
    s=text;
}
void replaceSymbol(string &text, string s)
{
   int k=0;

    string z="…";
    cout<<z;
        for (int i=0;i<s.size();i++)
    {
        if (s[i]=='1')
        {
            k=text.find(z);
            cout<<k<<" ";
           if (k==-1){cout<<"Недостаточно текста для встраивания"; break;}
            text.replace(k,3,".");
        }
        else
        {
            k=text.find("–");
```

```
            cout<<k<<" ";
            if (k==-1){cout<<"Недостаточно текста для встраивания"; break;}
            text.replace(k,3,"-");
        }
    }
}
void OutToFile(string text, string fileName)
{
    ofstream ff(fileName.c_str());
    ff<<text;
    ff.close();
}
void AtoB(string &chiperText)
{
    char x[16];
    string bin;
    for (int i=0;i<chiperText.size();i++)
    {
        _itoa(int(chiperText[i]),x,2);
        bin+=x;
        bin+=" ";

    }
    chiperText=bin;
}
```