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

Институт интеллектуальных кибернетических систем

Кафедра №42 «Криптология и кибербезопасность»

Отчёт о курсовой работе

по дисциплине "Стеганография"

«Метод изменения количества пробелов между словами

выровненного по ширине текста»

Преподаватель:  Лаврентьев Н.П.

Группа: Б17-565

Выполнил: Чыонг Тхи Ан Хай

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва 2020г

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РЕФЕРАТ**

Отчёт 12 с., 5 рис., 5 источников, 2 прил.

СТЕГАНОГРАФИЯ, СКРЫТИЕ ДАННЫХ В ТЕКСТЕ, МЕТОД ИЗМЕНЕНИЯ КОЛИЧЕСТВА ПРОБЕЛОВ, ИНТЕРВАЛ МЕЖДУ СЛОВАМИ, ПРОИЗВОЛЬНЫЙ ИНТЕРВАЛ, ВЫРОВНЕННЫЙ ПО ШИРИНЕ ТЕКСТ.

Объект исследования – метод изменения количества пробелов между словами выровненного по ширине текста.

Цель данной работы заключается в изучении метода изменения количества пробелов между словами выровненного по ширине текста, применении его на практике, а также выявлении его достоинств и недостатков.

В работе был изучен и применён на практике алгоритм сокрытия данных в тексте. Для реализации этого алгоритма использовался язык программирования С++.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc40731015)

[1 Скрытие данных в тексте 5](#_Toc40731016)

[1.1 Методы произвольного интервала 6](#_Toc40731017)

[1.1.1 Метод изменения количества пробелов между словами 6](#_Toc40731018)

[выровненного по ширине текста 6](#_Toc40731019)

[2 Алгоритм метода 7](#_Toc40731020)

[3 Пример работы программы 9](#_Toc40731021)

[4 Достоинства и недостатки 10](#_Toc40731022)

[ЗАКЛЮЧЕНИЕ 11](#_Toc40731023)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 12](#_Toc40731024)

[ПРИЛОЖЕНИЕ A: Встраивание сообщения в текст 13](#_Toc40731025)

[ПРИЛОЖЕНИЕ Б: Извлечение сообщения из текста 16](#_Toc40731026)

ВВЕДЕНИЕ

В связи с быстрым ростом информационных технологий появляется всё больше проблем, связанных с информационной безопасностью. Задача защиты информации от несанкционированного доступа решается с давних пор. Выделилось два подхода к решению данной задачи – криптография и стеганография.

Если криптография – это наука о методах сокрытия информации путем её шифрования, то стеганография изучает методы, связанные с сокрытием самого факта передачи информации [1].

Стеганография – это методы, основанные на различных правилах, которые обеспечивают скрытие самого факта нахождения тайной информации в той или иной среде. Всю стеганографию как науку можно разделить на несколько ветвей: классическую стеганографию, компьютерную стеганографию и цифровую стеганографию [2].

К настоящему времени разработано много методов сокрытия факта передачи информации. В целом их можно подразделить на несколько больших групп: передача информации в неподвижном изображении, в тексте, в аудио-файле, видео-изображении (в данном случае разговор идет о цифровой стеганографии). Предметом изучения данного работы стал метод, называемый методом изменения количества пробелов между словами выровненного по ширине текста.

1. Скрытие данных в тексте

Для скрытия конфиденциальных сообщений в тексте (так называемая лингвистическая стеганография) используется или обычная избыточность письменной речи, или же форматы представления текста.

Наиболее сложным местом для скрытия данных по многим причинам является электронная (файловая) версия текста [3]. В отличие от текстового файла его «жесткая» копия (например, бумажная) может быть обработана как высокоструктурированное изображение и поэтому является относительно легко поддающейся разнообразным методам скрытия, таким как незначительные изменения формата текстовых шаблонов, регулирование расстояния между определенными парами символов (кернинг), расстояния между строками и т.п. В значительной степени такая ситуация вызвана относительным дефицитом в текстовом файле избыточной информации, особенно в сравнении с графическими или, например, звуковыми файлами. В то время как в большинстве случаев в изображение и звук существует возможность внести незаметные глазу и неощутимые на слух модификации, даже дополнительная буква или знак пунктуации в тексте могут быть легко распознаны случайным читателем.

Скрытие данных в тексте требует поиска таких модификаций, которые были бы незаметными подавляющему большинству читателей. Авторы [4] рассматривают три группы методов, которые получили наибольшее распространение при встраивании скрываемых данных в текст:

* методы произвольного интервала, которые осуществляют встраивание путем манипуляции с пробельными символами (свободным местом на печатной полосе);
* синтаксические методы, которые работают с пунктуацией;
* семантические методы, в основу алгоритмов которых положено манипулирование словами, зависимое от скрываемых бит данных.
  1. Методы произвольного интервала

Существует, по меньшей мере, две причины, по которым манипулирование свободным местом в определенных случаях показывает довольно неплохие результаты. Во-первых, изменение количества пробелов в конце текстовой строки не вызывает существенных изменений в значении фразы или предложения. Во-вторых, среднестатистический читатель вряд ли заметит незначительные модификации свободного места страницы текста.

В [4] предложено три метода, которые для скрытия данных используют свободное место в тексте. Указанные методы оперируют с интервалами между предложениями, пропусками в конце текстовых строк и интервалами между словами в тексте, выровненном по ширине.

* + 1. Метод изменения количества пробелов между словами

выровненного по ширине текста

Данный метод позволяет скрывать данные в свободных местах текста, выровненного по ширине. При этом биты данных встраиваются путем управляемого выбора позиций, в которых будут размещены дополнительные пробелы. Один пробел между словами интерпретируется как «0». Два пробела — как «1». В среднем метод позволяет встраивать по несколько бит в одну строку [5].

Из-за ограничений, которые накладываются выравниванием текста по ширине, не каждый пробел между словами может использоваться для встраивания данных. Для возможности принятия однозначного решения при определении принимающей стороной, какие же именно из пробелов между словами скрывают встроенную информацию, а какие являются частью оригинального текста, в предложено использовать метод встраивания, аналогичный манчестерскому кодированию.

В результате упомянутого кодирования биты группируются попарно, причем последовательность «01» интерпретируется как «1», «10» — как «0», а пары «00» и «11» считаются пустыми. Например, извлеченное сообщение «1010000111» сводится к «001», тогда как сообщение «0011110011» представляет пустую строку.

1. Алгоритм метода

На рисунке 1 представлена блок-схема, описывающая встраивание скрытого сообщения в текстовый контейнер методом изменения количества пробелов между словами выровненного по ширине текста.

- Получение вектора двоичного представления скрыт.сооб.

- Подсчет количество символов в r-й строке Lr

Lmax = max (Lr)

r=1

К=1

Количество пробелов в r-й строке = t

**<** Lmax - Lr?

Для встраивания испол. floor() – 1 пар пробелов

остальные будут использоваться для удлинения строки

до Lmax путем создания пустых пар

Для встраивания испол. Lmax - Lr пар пробелов

все биты?

Послед. строка?

r = r +1

Нетт

Да

Да

Нет

Нет

Да

Рис. 1 – Блок - схема встраивания скрытого сообщения в текстовый контейнер

На рисунке 2 представлена блок-схема, описывающая извлечение скрытого сообщения из текстового контейнера.

Получение количества строк

r=1

,

Получение вектора Lr

-|Lr| = колич. Пробелов в r-й строке

-Элементы Lr – позиций пробелов

i=0

b [i] = Lr [i+1]-Lr[i]

,

b [i] = 1?

a [i] = 1

i+=1

,

a [i] = 0

i+=1

,

Все пробелы?

r = r +1

Из а выделяются подмассивы, состоящие из трех элементов:

‘100’ ‘0’; ‘010’ ‘1’; остальные нет

,

Преобразование двоичного кода в символы

,

Послед. строка?

Нет

Да

Нет

Да

Да

Нет

Рис. 2 – Блок - схема извлечения скрытого сообщения из текстовый контейнера

1. Пример работы программы

С помощью программы encode.cpp мы встраиваем скрываемое сообщение *Hi* в тексте test.txt (Рис.3).

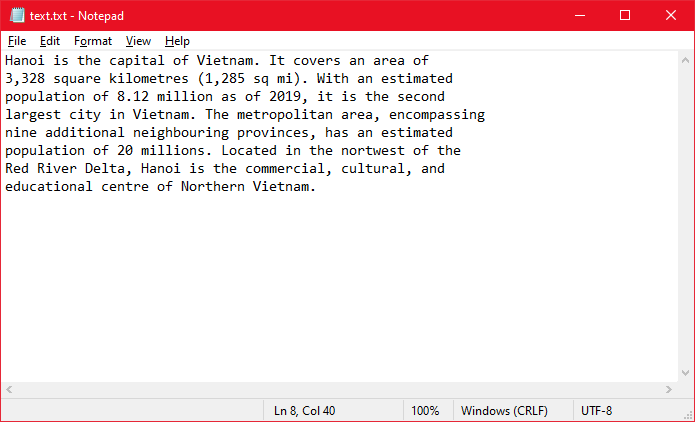


Рис. 3 - текстовый контейнер test.txt

Результат выполнено встраивание сообщения *Hi* в контейнер test.txt методом изменения количества пробелов между словами выровненного по ширине текста приведен на рисунке 4.

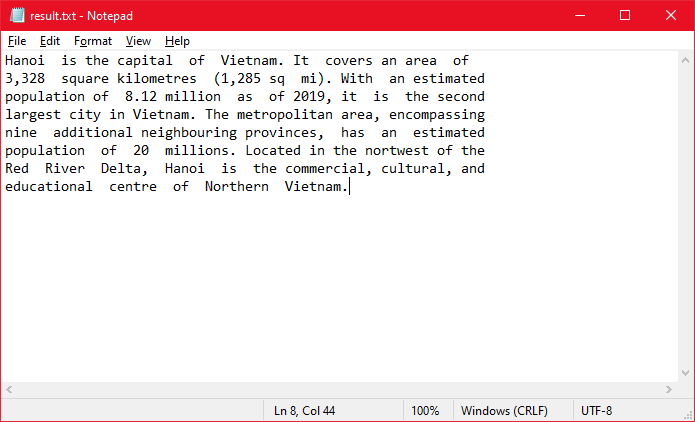
****

Рис. 4 - Текст со скрытым сообщением

Далее, мы используем программy decode.cpp, чтобы извлечить скрытое сообщение из контейнера result.txt. Получили скрытое сообщение: *Hi* (Рис.5).

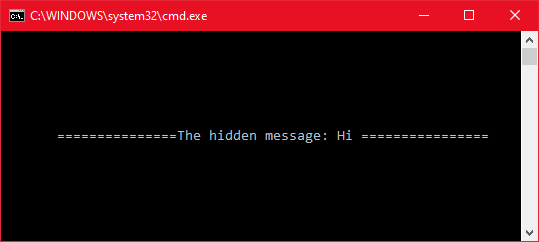


Рис. 5 - Извлечения скрытого сообщения из текстовый контейнера

1. Достоинства и недостатки

Легко увидеть, что преимущество этого метода - это его простая реализация. Трудно обнаружить непоследовательное и противоречивое использование свободных мест между словами на бумажной копии тескта, потому что текст - выровненный по ширине.

Но кроме несомненной простоты, данный метод имеет и ряд недостатков:

* метод можно использовать только в том случае, если ширина строк текста отличается;
* для встраивания незначительного количества бит требуется текст значительного объема;
* и наконец, некоторые программы обработки текста могут непреднамеренно удалять дополнительно внесенные пробелы.

ЗАКЛЮЧЕНИЕ

В настоящей работе были проведены исследование метода изменения количества пробелов между словами выровненного по ширине текста, рассмотрение примеры данного метода на практике, обсуждение достоинств и недостатков метода.

В ходе работы получены следующие результаты:

* описание метода изменения количества пробелов между словами выровненного по ширине текста;
* блок-схемы алгоритмов встраивания и извлечения информации в/из текст(а), приложения, реализующие этих алгоритмов использовались язык программирования С++;
* результаты применения метода на примере;
* список достоинств и недостатков метода.

Практическая ценность работы заключается в разработанном стеганографическом методе и реализующем его программном средстве для встраивания скрытого сообщения в тескт и извлечения из текста.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Грибунин В. Г., В. Г. Грибунин, И. Н. Оков, И. В. Туринцев Цифровая стеганография - М.: СОЛОН-Пресс, 2002. - 272 с.
2. Конахович Г.Ф., Пузыренко А.Ю. Компьютерная стеганография. Теория и практика М.: МК-Пресс, 2006. — 288 с. — ISBN: 966-8806-06-9.
3. Методы сокрытия информации в текстовых данных [Электронный ресурс] / Режим доступа <https://studopedia.org/4-95485.html> (дата обращения 15.03.2020).
4. W. Bender, D. Gruhl, N. Morimoto, A. Lu, Techniques for Data Hiding. IBM Systems Journal, 35(3&4): pp. 313-336, 1996.
5. Методы произвольного интервала [Электронный ресурс] / Режим доступа <https://helpiks.org/6-82910.html> (дата обращения 12.03.2020).

ПРИЛОЖЕНИЕ A: Встраивание сообщения в текст

// Encode.cpp

#include <iostream>

#include <string>

#include <fstream>

#include <windows.h>

#include <map>

#include <math.h>

using namespace std;

int main()

{

fstream fi;

fi.open("text.txt", ios::in);////открыли

if (fi.good()) //если файл вообще нашли

{ //Посчитать количество символов каждой строки и найти строку -max длиннa

string a;

string rows[100];

int k = 0;

while (getline(fi, a)) {

rows[k] = a;

k++;

}

size\_t s[100];

for (int i = 0; i < k; ++i) {

s[i] = rows[i].size();

}

size\_t max = 0;

for (int i = 0; i < k; ++i) {

if (s[i] > max) max = s[i];

}

//вставляемая строка

string insertString;

std::cout << "Enter the insertion string: ";

cin >> insertString;

string decTo2;

//преобразование ascii кода вставляемой строки в двоичный вид

for (size\_t ind = 0; ind < insertString.size(); ++ind) {

int ascii = (int)insertString[ind];

string binary;

do {

binary = to\_string(ascii % 2).c\_str() + binary;

ascii = (int)ascii / 2;

} while (ascii > 0);

if (binary.size() < 8) binary = '0' + binary;

decTo2 += binary;

}

std::cout << decTo2 << endl;

string resultText;

int index = 0;

ofstream out("result.txt");

//Разработать каждую строку

for (int i = 0; i < k; ++i)

{ //если строка имеет max символ -> записать в result

if (s[i] == max) {

resultText += rows[i] + "\n";

continue;

}

size\_t blankPos = 0;

int vector[100];

string a = rows[i];

//считать количество пробелов и позиций в строке

for (int j = 0; j < s[i]; ++j)

{

if (a[j] == ' ') {

vector[blankPos] = j;

blankPos++;

}

}

size\_t t = blankPos / 2;

size\_t space;

// если (blankPos/2 >= Lmax - Lr) для встраивания испол. Lmax - Lr пар пробелов

if ((t >= (max - s[i])) && (index < decTo2.size()))

{

space = (max - s[i]);

size\_t x = 0;

while (x <= space \* 2)

{

if (decTo2[index] == '0')

{

if (x >= blankPos) break;

a.insert(vector[x], 1, ' ');

for (int j = x + 1; j < blankPos; ++j)

{

vector[j] += 1;

}

x += 2;

}

if (decTo2[index] == '1')

{

x += 1;

if (x >= blankPos) break;

a.insert(vector[x], 1, ' ');

for (int j = x + 1; j < blankPos; ++j)

{

vector[j] += 1;

}

x += 1;

}

index += 1;

if (index >= decTo2.size()) break;

}

resultText += a + "\n";

continue;

}

// если (blankPos/2 < Lmax - Lr) для встраивания испол.floor(blankPos/2)-1 пар пробелов

if ((t < (max - s[i])) && (index < decTo2.size()))

{

space = floor(blankPos/2)-1;

size\_t x = 0;

while (x <= space \* 2)

{

if (decTo2[index] == '0')

{

if (x >= blankPos) break;

a.insert(vector[x], 1, ' ');

for (int j = x + 1; j < blankPos; ++j)

{

vector[j] += 1;

}

x += 2;

}

if (decTo2[index] == '1')

{

x += 1;

if (x >= blankPos) break;

a.insert(vector[x], 1, ' ');

for (int j = x + 1; j < blankPos; ++j)

{

vector[j] += 1;

}

x += 1;

}

index += 1;

if (index >= decTo2.size()) break;

}

//Добавить пробелов для выровненного по ширине текста

size\_t bonus = max - a.size();

while (bonus !=0)

{

if (x >= blankPos) break;

a.insert(vector[x], 1, ' ');

for (int j = x + 1; j < blankPos; ++j)

{

vector[j] += 1;

}

x += 1;

bonus -= 1;

}

resultText += a + "\n";

continue;

}

// если все биты встаивили

if (index == decTo2.size() )

{

size\_t x = 0;

size\_t bonus = max - a.size();

while (bonus != 0)

{

if (x >= blankPos) break;

a.insert(vector[x], 1, ' ');

for (int j = x + 1; j < blankPos; ++j)

{

vector[j] += 1;

}

x += 1;

bonus -= 1;

}

resultText += a + "\n";

}

}

std::cout << "Results saved to file: result.txt!\n";

out << resultText;

out.close();

}

else

{

std::cout << "file test.txt not found" << endl;

}

fi.close();

system("pause");

return 0;

}

ПРИЛОЖЕНИЕ Б: Извлечение сообщения из текста

//Decode.cpp

#include <iostream>

#include <bitset>

#include <sstream>

#include <string>

#include <fstream>

#include <windows.h>

using namespace std;

int main()

{

fstream fi;

fi.open("result.txt", ios::in);////открыли

if (fi.good()) //если файл вообще нашли

{

string a;

string rows[100];

int k = 0;

while (getline(fi, a))

{

rows[k] = a;

k++;

}

string Text,HiddenText;

for (int i = 0; i < k; ++i)

{

string a = rows[i];

size\_t blankPos = 0;

int space[100];

//считать количество пробелов и позиций в строке

for (int j = 0; j < a.size(); ++j)

{

if (a[j] == ' ')

{

space[blankPos] = j;

blankPos++;

}

}

int b[100];

//вектор b, значение каждого элемента которого зависит от

//результата вычисления разности между следующим и текущим

//значениями элементов вектора space

for (int j = 0; j < blankPos; ++j)

{

if (j == blankPos - 1) b[j] = -1;

b[j] = space[j + 1] - space[j];

if (b[j] != 1) b[j] = 0;

}

//подсчитать двоичного кода сообщения

string massiv;

int three = blankPos / 3;

for (int i = 0; i < three; ++i)

{

if ((b[i \* 3] == 1) && (b[i \* 3 + 1] == 0) && (b[i \* 3 + 2] == 0))

massiv += '0';

else if ((b[i \* 3] == 0) && (b[i \* 3 + 1] == 1) && (b[i \* 3 + 2] == 0))

massiv += '1';

else

break;

}

Text += massiv;

}

// Преобразование двоичного кода в символы

stringstream sstream(Text);

while (sstream.good())

{

bitset<8> bits;

sstream >> bits;

HiddenText += char(bits.to\_ulong());

}

std::cout << "===============The hidden message: " << HiddenText << "================"<< endl;

}

else

{

std::cout << "file test.txt not found" << endl;

}

fi.close();

system("pause");

return 0;

}