Міністерство освіти і науки України

Харківський національний університет ім. В. Н. Каразіна

Факультет комп’ютерних наук

Кафедра безпеки інформаційних систем і технологій

Лабораторна робота №1

*з навчальної дисципліни*

«Стеганографія»

Виконав:

Студент групи КБ-41

Кравченко Є.М.

Перевірив:

доцент

Нарєжній О.П.

Харків – 2020 р.

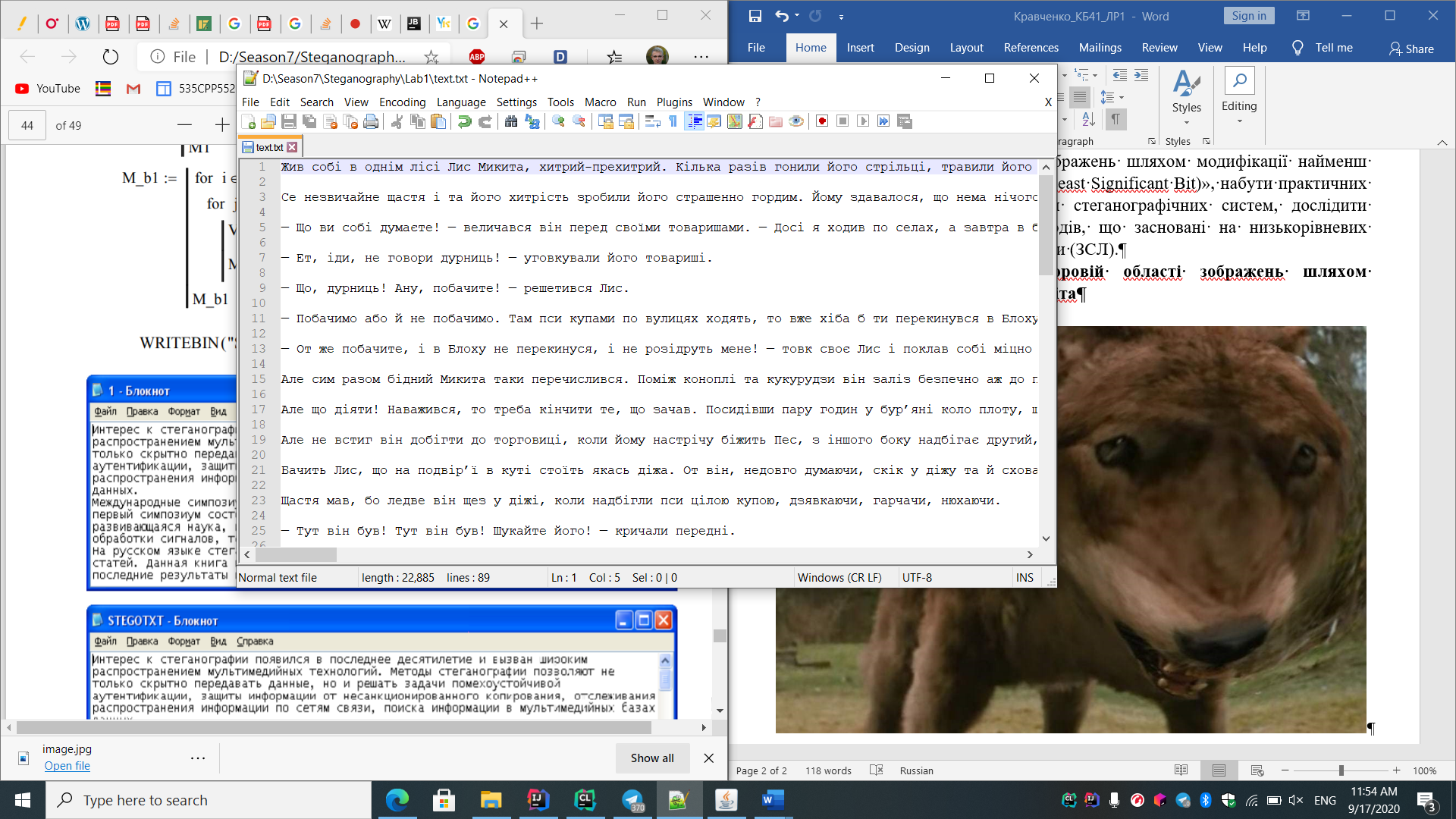
**Тема:** Приховування даних у просторовій області зображень шляхом модифікації найменш значущого біта.

**Мета роботи:** закріпити теоретичні знання за темою «Приховування даних у просторовій області нерухомих зображень шляхом модифікації найменш значущого біта даних (НЗБ, LSB – Least Significant Bit)», набути практичних вмінь та навичок щодо розробки стеганографічних систем, дослідити властивості стеганографічних методів, що засновані на низькорівневих властивостях зорової системи людини (ЗСЛ).

**Приховування даних у просторовій області зображень шляхом модифікації найменш значущого біта**

Вихідні дані:





**Програмна реалізація алгоритмів приховування та вилучення повідомлень методом LSB:**

Приховання:

public BufferedImage encode(BufferedImage bufferedImage, String text) throws ThroughputException {

String mb = toBinaryString(text);

if (mb.length() > bufferedImage.getWidth() \* bufferedImage.getHeight()) {

throw new ThroughputException();

}

for (int k = 0; k < mb.length(); k++) {

int i = k / bufferedImage.getHeight();

int j = k - i \* bufferedImage.getHeight();

Color color = new Color(bufferedImage.getRGB(i, j)); // get color from image

char[] changeRedColor = toBinaryString(color.getRed()).toCharArray();

changeRedColor[LSB] = mb.charAt(k);

bufferedImage.setRGB(i, j, new Color(Integer.parseInt(new String(changeRedColor), 2), color.getGreen(), color.getBlue()).getRGB());

}

return bufferedImage;

}

Вилучення:

public String decode(BufferedImage bufferedImage) {

StringBuilder binaryString = new StringBuilder();

for (int i = 0; i < bufferedImage.getWidth(); i++) {

for (int j = 0; j < bufferedImage.getHeight(); j++) {

binaryString.append(toBinaryString(new Color(bufferedImage.getRGB(i, j)).getRed()).charAt(LSB));

}

}

byte[] result = new byte[bufferedImage.getWidth() \* bufferedImage.getWidth() / 8];

for (int i = 0, k = 0; i + 8 < binaryString.length(); i += 8) {

byte b = (byte) (int) Integer.valueOf(binaryString.substring(i, i + 8), 2);

result[k++] = b;

}

return new String(result, StandardCharsets.UTF\_8);

}

Візуальне порівняння порожньою та заповненого контейнерів:



**Програмна реалізація алгоритмів приховування та вилучення повідомлень методом ПВП:**

Приховання:

public BufferedImage encode(BufferedImage bufferedImage, String message) throws ThroughputException {

String mb = toBinaryString(message);

if (messageIsLargestContainer(bufferedImage, mb)) {

throw new ThroughputException();

}

mb = encode(mb);

for (int k = 0; k < mb.length(); k++) {

int j = k / bufferedImage.getWidth();

int i = k - j \* bufferedImage.getWidth();

Color color = new Color(bufferedImage.getRGB(i, j));

int red = color.getRed();

char[] bRed = toBinaryString(red).toCharArray();

bRed[LSB] = mb.charAt(k);

bufferedImage.setRGB(i, j, new Color(Integer.parseInt(new String(bRed), 2), color.getGreen(), color.getBlue()).getRGB());

}

return bufferedImage;

}

Вилучення:

public String decode(BufferedImage bufferedImage) {

StringBuilder binaryString = new StringBuilder();

for (int j = 0; j < bufferedImage.getHeight(); j++) {

for (int i = 0; i < bufferedImage.getWidth(); i++) {

binaryString.append(toBinaryString(new Color(bufferedImage.getRGB(i, j)).getRed()).charAt(LSB));

}

}

byte[] result = new byte[bufferedImage.getHeight() \* bufferedImage.getWidth() / 8];

for (int i = 0, k = 0; i + 8 < binaryString.length(); i += 8) {

byte b = (byte) (int) Integer.valueOf(multiply(binaryString.substring(i, i + 8), P, true), 2);

result[k++] = b;

}

return new String(result, StandardCharsets.UTF\_8);

}

**Програмна реалізація алгоритмів приховування та вилучення повідомлень методом ПВІ:**

Приховання:

public BufferedImage encode(BufferedImage bufferedImage, String message) throws ThroughputException {

String mb = toBinaryString(message);

if (messageIsLargestContainer(bufferedImage, mb)) {

throw new ThroughputException();

}

mb = encode(mb);

for (int k = 0; k < mb.length(); k++) {

int j = k / bufferedImage.getWidth();

int i = k - j \* bufferedImage.getWidth();

Color color = new Color(bufferedImage.getRGB(i, j));

int red = color.getRed();

char[] bRed = toBinaryString(red).toCharArray();

bRed[LSB] = mb.charAt(k);

bufferedImage.setRGB(i, j, new Color(Integer.parseInt(new String(bRed), 2), color.getGreen(), color.getBlue()).getRGB());

}

return bufferedImage;

}

Вилучення:

public String decode(BufferedImage bufferedImage) {

StringBuilder binaryString = new StringBuilder();

for (int j = 0; j < bufferedImage.getHeight(); j++) {

for (int i = 0; i < bufferedImage.getWidth(); i++) {

binaryString.append(toBinaryString(new Color(bufferedImage.getRGB(i, j)).getRed()).charAt(LSB));

}

}

byte[] result = new byte[bufferedImage.getHeight() \* bufferedImage.getWidth() / 8];

for (int i = 0, k = 0; i + 8 < binaryString.length(); i += 8) {

byte b = (byte) (int) Integer.valueOf(multiply(binaryString.substring(i, i + 8), P, true), 2);

result[k++] = b;

}

return new String(result, StandardCharsets.UTF\_8);

}

Допоміжні фукції:

private boolean messageIsLargestContainer(BufferedImage bufferedImage, String mb) {

return mb.length() > bufferedImage.getWidth() \* bufferedImage.getHeight();

}

private int charToInt(char charAt) {

return charAt == '0' ? 0 : 1;

}

private String multiply(String m, int[][] P, boolean reverse) {

if (m.length() != P.length || m.length() != P[0].length) {

throw new ArithmeticException();

}

StringBuilder res = new StringBuilder();

for (int i = 0; i < m.length(); i++) {

int sm = 0;

for (int j = 0; j < m.length(); j++) {

int tmp = reverse ? charToInt(m.charAt(i)) \* P[i][j] : charToInt(m.charAt(i)) \* P[j][i];

sm += tmp % 2;

}

res.append(sm);

}

return res.toString();

}

private String encode(String mb) {

StringBuilder result = new StringBuilder();

for (int i = 0; i < mb.length(); i += 8) {

result.append(multiply(mb.substring(i, i + 8), P, false));

}

return result.toString();

}

**Експериментальні дослідження зоровогопорогу чутливості людини до зміни яскравості зображень**

Внесено зміни до реалізації алгоритму вбудовування даних у нерухомі зображення методом LSB. Для цього змінимо порядковий номер біта, який використовується для вбудовування інформації в двійковому поданні елементів контейнера (окремих байтів яскравості конкретних пікселів зображення).



Змінено біт №1



Змінено біт №2



Змінено біт №3



Змінено біт №4



Змінено біт №5



Змінено біт №6



Змінено біт №7



Змінено біт №8

**Висновок**

В цій лаботаторній роботі було закріплено теоретичні знання за темою «Приховування даних у просторовій області нерухомих зображень шляхом модифікації найменш значущого біта даних (НЗБ, LSB – Least Significant Bit)», набуто практичних вмінь та навичок щодо розробки стеганографічних систем, досліджео властивості стеганографічних методів, що засновані на низькорівневих властивостях зорової системи людини (ЗСЛ).