МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Інститут комп'ютерних наук та інформаційних технологій

Кафедра інформаційних систем та мереж



ЗВІТ

про виконання лабораторної роботи № 4

«Шифрування даних методом стеганографії»

з дисципліни

«Технології захисту інформації»

Виконала:

студентка групи ІТ-31

Щербак Л. В. Прийняв викладач:

Досин Д.Г.

Львів 2022

**Мета роботи:** Навчитися опрацьовувати (шифрувати та дешифрувати)

файли з прихованими інформаційними повідомленнями.

**Завдання роботи:**

індивідуальне завдання з детальним формулюванням розв’язуваної

задачі, використовувані (власні) теоретичні відомості;

1. Написати програму на мові java яка виконує криптографічні перетворення (шифрування та дешифрування) над файлами за стеганограмним способом (файл повідомлення, файл - "контейнер" і файл – результат) відповідно до заданого варіанту, таблиця 1. Провести порівняння розподілу ймовірностей наявних символів файлу до та після шифрування (приховування). На вибір вибрати розподіл по 10-15

символах.

Розміщення однобайтних елементів повідомлення в файл

графічного формату 32-біти на колір

2. Здійснити компіляцію програми.

**Відповіді на контрольні питання:**

1. Що таке стеганографія і які її базові принципи?

Стеганогра́фія — (з грец. στεγανός — прихований + γράφω — пишу) — тайнопис, при якому повідомлення, закодоване таким чином, що не виглядає як повідомлення — на відміну від криптографії. Таким чином непосвячена людина принципово не може розшифрувати повідомлення — бо не знає про факт його існування. Якщо криптографія приховує зміст повідомлення, то стеганографія приховує сам факт існування повідомлення.

Щодо впровадження засобів програмно-технічного захисту в ІС, розрізняють два основні його способи:

додатковий захист — засоби захисту є доповненням до основних програмних і апаратних засобів комп'ютерної системи;

вбудований захист — механізми захисту реалізуються у вигляді окремих компонентів ІС або розподілені за іншими компонентами системи.

Перший спосіб є гнучкішим, його механізми можна додавати і вилучати за потребою, але під час його реалізації можуть постати проблеми забезпечення сумісності засобів захисту між собою та з програмно-технічним комплексом ІС. Вмонтований захист вважається більш надійним і оптимальним, але є жорстким, оскільки в нього важко внести зміни. Таким доповненням характеристик способів захисту зумовлюється те, що в реальній системі їх комбінують.

2. Поняття контейнера, їх види?

Контейнер – це будь-який файл-зображення, що містить в собі зашифроване повідомлення або буде його містити. Контейнер в процесі шифрування повідомлення – змінюється майже непомітно для людського ока.

Всього є 2 види контейнерів – контейнер-оригінал, що не містить повідомлення, та контейнер-результат, що має зашифроване повідомлення. Саме повідомлення – теж можна реалізувати, як контейнер, але вже текстовий.

3. Опишіть відомі алгоритми стеганографії?

Існуючі алгоритми вбудовування таємної інформації можна поділити на декілька підгруп:

Працюючі з самим цифровим сигналом. Наприклад, метод LSB (Least Significant Bit);

«Впаювання» прихованої інформації. В даному випадку відбувається накладення приховуваного зображення (звуку, іноді тексту) поверх оригіналу. Часто використовується для вбудовування ЦВЗ (цифровий водяний знак).

Використання особливостей форматів файлів. Сюди можна віднести запис інформації в метадані або в різні інші не використовувані зарезервовані поля файлу.

За способом вбудовування інформації стегоалгоритми можна розділити на лінійні (адитивні: А17, А18, Ь18Б, А21, А25), нелінійні та інші.

**Текст програми реалізації:**

package lab4;

import javax.imageio.ImageIO;

import java.io.File;

import java.io.IOException;

import java.awt.\*;

import java.awt.image.BufferedImage;

import java.math.BigInteger;

public class ImageSteganography {

//Get a bit from the byte on the pos.

//@param val - byte to get bit of

//@param pos - pos of a bit in the byte

//@return bit from the byte on the pos

public static byte getBit(byte val, int pos) {

return (byte) ((val & (1 << pos)) >> pos);

}

//Get a byte array representation of the string.

//@param val - string to get bytes of

//@return byte array representation of the string

public static byte[] getBytes(String val) {

return val.getBytes();

}

//Get a string representation of the byte array.

//@param byte - byte array

//@return string representation of the byte array

public static String getString(byte[] array) {

return new String(array);

}

//Get a byte array representation of the integer number.

//@param integer - integer number

//@return byte - array representation of the integer number

//@throws IOException thrown if errors while converting are occurred

public static byte[] intToByteArray(final int integer) throws IOException {

return BigInteger.valueOf(integer).toByteArray();

}

//Prepare text to encode.<br> Make a byte array containing bytes of source text and a length of source text.

//@param text - source text

//@return byte array of a source text and a length of it

public static byte[] prepareTextToEncode(String text) {

String textToEncode = String.format("%1$s %2$s", text.length(), text);

//System.out.println(textToEncode);

byte[] encode = getBytes(textToEncode);

return encode;

}

//Make a copy of the image.

//@param imageToCopy - image to make copy of

//@return copy of the image

public static BufferedImage makeImageCopy(BufferedImage imageToCopy) {

BufferedImage result = new BufferedImage(imageToCopy.getWidth(),

imageToCopy.getHeight(),

imageToCopy.getType());

Graphics g = result.getGraphics();

g.drawImage(imageToCopy, 0, 0, null);

return result;

}

//Encode text into a buffered image using Least Significant Bit algorithm.<br>

//@param text - text to encode

//@param bitmap - buffered image

//@return encoded buffered image

public static BufferedImage encode(String text, BufferedImage bitmap) throws Exception {

int i = 0;

int j = 8;

//Creating a result image

BufferedImage result = makeImageCopy(bitmap);

//Preparing the text to encode

byte[] encode = prepareTextToEncode(text);

for(int y = 0; y < result.getHeight(); y++) {

for(int x = 0; x < result.getWidth(); x++) {

Color pixel = new Color(result.getRGB(x, y));

int R = (j == 8) ? pixel.getRed() & 254 : pixel.getRed() & 254 | +getBit(encode[i], --j);

int G = pixel.getGreen() & 254 | +getBit(encode[i], --j);

int B = pixel.getBlue() & 254 | +getBit(encode[i], --j);

Color newPixel = new Color(R, G, B);

result.setRGB(x, y, newPixel.getRGB());

if(j == 0) {

j = 8;

i++;

}

if(i == encode.length) { break; }

}

if(i == encode.length) { break; }

}

return result;

}

//Decode text from a buffered image using Least Significant Bit algorithm.

//@param bitmap - encoded buffered image

//@return decoded text

public static String decode(BufferedImage bitmap) {

int textLength = 0;

boolean containsMessage = false;

StringBuilder information = new StringBuilder();

StringBuilder tmp = new StringBuilder();

for(int y = 0; y < bitmap.getHeight(); y++) {

for(int x = 0; x < bitmap.getWidth(); x++) {

Color pixel = new Color(bitmap.getRGB( x, y ));

tmp.append(pixel.getRed() % 2);

tmp.append(pixel.getGreen() % 2);

tmp.append(pixel.getBlue() % 2);

if(tmp.length() == 9) {

try {

Integer tmpInt = Integer.parseInt(tmp.toString(), 2);

information.append(getString(intToByteArray(tmpInt)));

if(!containsMessage && tmpInt == 32) {

containsMessage = true;

information.deleteCharAt(information.length() - 1);

textLength = Integer.parseInt(information.toString());

information.setLength(0);

}

if(!containsMessage && tmpInt == 0) {

throw new Exception();

}

tmp.setLength(0);

} catch(Exception e) {

containsMessage = true;

information.setLength(0);

information.append( "Decode error!" );

}

}

if(containsMessage && information.length() >= textLength) { break; }

}

if(containsMessage && information.length() >= textLength) { break; }

}

return information.toString();

}

//Read Image from file

//@param fName - file name

//@return encoded buffered image

public static BufferedImage readFile(String fName) throws IOException {

BufferedImage bi = null;

try {

bi = ImageIO.read(new File(fName));

} catch (IOException e) {}

return bi;

}

//Save Image to file

//@param fName - file name

//@param bi - encoded buffered image

public static void saveFile(BufferedImage bi, String fName) throws IOException {

try {

File outputfile = new File(fName);

ImageIO.write(bi, "PNG", outputfile);

} catch (IOException e) {}

}

public static void main(String[] args) throws Exception {

String fNameOrig = "one-dollar.png";

String fNameEncrypted = "one-dollar\_encrypted.png";

BufferedImage img1, img2, img3 = null;

String msg = "Encryption of data by the method of steganography";

System.out.println("Original Message : "+msg);

System.out.println("Original Image file : "+fNameOrig);

System.out.println();

//Read Image, Encrypt Message in Image, Save Image

System.out.println("Reading Image from file : "+fNameOrig);

img1 = readFile(fNameOrig);

try {

System.out.println("Encoding...");

img2 = encode(msg, img1);

} catch( Exception e ) {

return;

}

System.out.println("Save encrypted Image to file : "+fNameEncrypted);

saveFile(img2, fNameEncrypted);

System.out.println();

//Read Image, Decode Message from encrypted Image, Return Message

System.out.println("Reading encrypted Image from file : "+fNameEncrypted);

img3 = readFile(fNameEncrypted);

System.out.println("Decoding...");

msg = decode(img3);

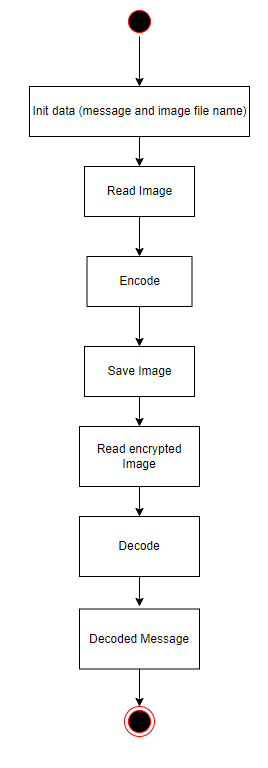
System.out.println("Decoded Message : "+msg);

System.out.println();

}

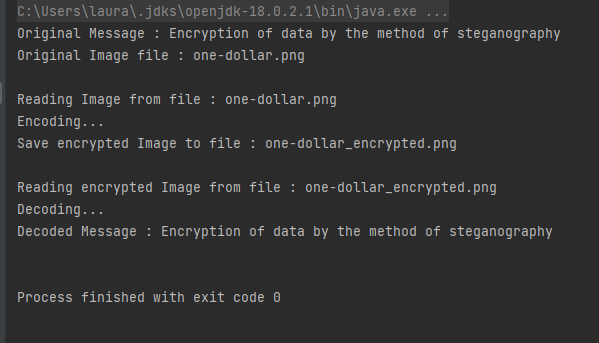
}

**Блок-схема:**



**Результати роботи програми:**





**Висновки:** Навчилася опрацьовувати (шифрувати та дешифрувати) файли з прихованими інформаційними повідомленнями.