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»

Виконала:

Студентка групи КБ-41
Кононченко А. В.

Перевірив:

Доцент
Нарєжній О. П.

Харків – 2020

Лабораторна робота №1

на тему:

«Приховування даних у просторовій області нерухомих 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).

Хід роботи

Реалізація алгоритму приховування і вилучення повідомлень методом заміни найменш значущого біта даних

Рисунок 1 – Початкове зображення

Рисунок 2 – Область, у яку буде проведено запис бітів повідомлення:

Повідомлення для приховування:

```
Steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection; the secret data is then extracted at its destination. The use of steganography can be combined with encryption as an extra step for hiding or protecting data.
```

Біти повідомлення будуть приховуватися у каналі червоного кольору зображення.

Вилучена інформація:

```
Steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection; the secret data is then extracted at its destination. The use of steganography can be combined with encryption as an extra step for hiding or protecting data.øªwuRõXÆð³Á¯PPe>=Þ™á□G‡ø¤Çbî*—□9´¥>□nö□©öUÐÃiþ-Kñ¤³8-wp÷8‘^Y˜xÏ‡08————————————————À0ÿ?>Þw;8˜
```

Програмна реалізація

Функція конвертації тексту в бінарний вигляд:

```
function convertTextToBinary(input) {
    let output = [];
    for (let i = 0; i < input.length; i++) {
        let tmp = input[i].charCodeAt(0).toString(2);
        output.push(make8bit(tmp));
    }
    return output.join('');
}
```

Функція конвертації бінарного тексту в символьний вигляд:

```
function convertBinaryToText(str) {
    str = str.match(/.{1,8}/g).toString().replace(/,/g, ' ');
    str = str.split(' ');
    let binCode = [];
    for (let i = 0; i < str.length; i++) {
        binCode.push(String.fromCharCode(parseInt(str[i], 2)));
    }
    return binCode.join('');
}
```

Функція перетворення масиву даних зображення у двійковий вид:

```
function arrayToBinary(data) {
    let array = [];
    for (let i = 0; i < data.length; i++) {
        array[i] = make8bit(data[i].toString(2));
    }
    return array;
}
```

Функція перетворення масиву даних зображення з двійкового виду до цілочисельного:

```
function parseFromBinary(arr) {
    for (let i = 0; i < arr.length; i++) {
        arr[i] = parseInt(arr[i], 2);
    }
    return arr;
}
```

Функція приведення символів до 8-бітного виду:

```
function make8bit(str) {
    let result = [];
    for (let i = 0; i < 8 - str.length; i++) {
        result.push('0');
    }
    result.push(str);
    return result.join('');
}
```

Функція заміни бітів:

```
function replaceChars2(string, index, replacement) {
    let strStart = string.slice(0, index);
    let strEnd = string.slice(index+1);
    return strStart + replacement + strEnd;
}
```

Функція кодування зображення:

```
let encode = function () {
    let imageData = ctx.getImageData(0, 0, width, height);
    let encodeImageData = arrayToBinary(imageData.data);
    let input = document.getElementById("message").value;
    input = convertTextToBinary(input);
    let array = '';
    for (let i = 0, index = 0; i < input.length * 4; i++) {
        if (i % 4 === 0) {
            encodeImageData[i] = replaceChars(encodeImageData[i],
            LSB, input[index]);
            array += input[index];
            index++;
        }
    }
    encodeImageData = parseFromBinary(encodeImageData);
    imageData.data.set(encodeImageData);
    ctx.putImageData(imageData, 0, 0);
    alert("Your data was successfully encoded!");
};
```

Функція декодування зображення:

```
let decode = function () {
    let output;
    let decodeImageData = arrayToBinary(ctx.getImageData(0, 0,
width, height).data);
    let decodedText = '';
    for (let i = 0; i < decodeImageData.length; i += 4) {
        decodedText += getChar(decodeImageData[i], LSB);

    }
    output.innerText += convertBinaryToText(decodedText);
};
```

Експериментальні дослідження зорового порогу чутливості людини до змін яскравості зображень

Порівняння значень пікселів

Оригінальні пікселі зображення:

```
Uint8ClampedArray(1518000) [72, 67, 71, 255, 71, 67, 71, 255, 70, 67, 70, 255, 68, 66, 69, 255,
67, 65, 69, 255, 66, 64, 69, 255, 65, 63, 68, 255, 64, 62, 67, 255, 64, 63, 68, 255, 64, 63, 68,
255, 63, 62, 68, 255, 63, 62, 68, 255, 63, 62, 68, 255, 63, 62, 68, 255, 63, 62, 68, 255, 62, 6
1, 67, 255, 62, 61, 67, 255, 61, 60, 66, 255, 61, 60, 66, 255, 61, 60, 66, 255, 61, 60, 66, 255,
61, 60, 66, 255, 62, 61, 67, 255, 62, 61, 67, 255, 63, 62, 68, 255, …]
```

Пікселі зображення після запису повідомлення у наймолодший біт:

```
(1518000) [72, 67, 71, 255, 73, 67, 71, 255, 68, 67, 70, 255, 68, 66, 69, 255, 67, 65, 70, 25
5, 66, 64, 69, 255, 64, 63, 68, 255, 64, 62, 67, 255, 64, 63, 68, 255, 65, 63, 68, 255, 63, 6
2, 68, 255, 62, 62, 68, 255, 62, 62, 68, 255, 62, 62, 68, 255, 62, 62, 68, 255, 63, 61, 67, 2
55, 62, 61, 67, 255, 61, 60, 66, 255, 61, 60, 66, 255, 60, 60, 66, 255, 60, 60, 66, 255, 61,
60, 66, 255, 63, 61, 67, 255, 63, 61, 67, 255, 62, 62, 68, 255, …]
```

Пікселі зображення після запису повідомлення у 2 молодших біти:

```
(1518000) [74, 67, 71, 255, 72, 67, 71, 255, 69, 67, 70, 255, 68, 66, 69, 255, 66, 65, 70, 25
5, 65, 64, 69, 255, 64, 63, 68, 255, 66, 62, 67, 255, 66, 63, 68, 255, 65, 63, 68, 255, 62, 6
2, 68, 255, 63, 62, 68, 255, 62, 62, 68, 255, 63, 62, 68, 255, 60, 62, 68, 255, 61, 61, 67, 2
55, 62, 61, 67, 255, 61, 60, 66, 255, 61, 60, 66, 255, 62, 60, 66, 255, 62, 60, 66, 255, 61,
60, 66, 255, 62, 61, 67, 255, 60, 61, 67, 255, 60, 62, 68, 255, …]
```

Пікселі зображення після запису повідомлення у 3 молодших біти:

```
(1518000) [74, 67, 71, 255, 74, 67, 71, 255, 64, 67, 70, 255, 67, 66, 69, 255, 64, 65, 70, 25
5, 69, 64, 69, 255, 65, 63, 68, 255, 71, 62, 67, 255, 70, 63, 68, 255, 65, 63, 68, 255, 57, 6
2, 68, 255, 59, 62, 68, 255, 57, 62, 68, 255, 61, 62, 68, 255, 57, 62, 68, 255, 57, 61, 67, 2
55, 60, 61, 67, 255, 56, 60, 66, 255, 56, 60, 66, 255, 59, 60, 66, 255, 59, 60, 66, 255, 60,
60, 66, 255, 61, 61, 67, 255, 63, 61, 67, 255, 62, 62, 68, 255, …]
```

Пікселі зображення після запису повідомлення у 4 молодших біти:

```
(1518000) [66, 67, 71, 255, 65, 67, 71, 255, 70, 67, 70, 255, 72, 66, 69, 255, 70, 65, 70, 25
5, 78, 64, 69, 255, 78, 63, 68, 255, 68, 62, 67, 255, 70, 63, 68, 255, 73, 63, 68, 255, 54, 6
2, 68, 255, 50, 62, 68, 255, 52, 62, 68, 255, 48, 62, 68, 255, 54, 62, 68, 255, 51, 61, 67, 2
55, 54, 61, 67, 255, 63, 60, 66, 255, 54, 60, 66, 255, 63, 60, 66, 255, 54, 60, 66, 255, 61,
60, 66, 255, 54, 61, 67, 255, 58, 61, 67, 255, 54, 62, 68, 255, …]
```

Запис у 4 молодші біти зображення привів до помітних спотворень (рис. 3).

Рисунок 3 – Спотворення зображення при записі повідомлення у 4 молодші біти каналу червоного кольору

У випадку запису бітів повідомлення у найстарший біт значення червоного кольору пікселя зображення набуває помітних спотворень:

Рисунок 4 – Спотворення зображення при записі повідомлення у найстарший біт

Таким чином, експериментально доведено, що допустимим є використання трьох молодших бітів для запису інформаційних бітів повідомлення. Починаючи з четвертого біту, вбудовані дані помітно спотворюють пікселі зображення.

Реалізація алгоритму приховування і вилучення повідомлень методом псевдовипадкової перестановки

Використовується те саме зображення і та сама просторова область (адже після вбудовування інформації відбувається методом заміни найменш значущого біта). Для реалізації алгоритму необхідно зашифрувати вхідний текст переставним шифром. Для цього використовується наступна квадратна матриця:

```
let keyMatrix = [
    [1, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 0, 0, 0],
    [0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 1, 0],
    [0, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 1]
];
```

При множенні повідомлення, представленого у двійковому вигляді, на матрицю відбувається перестановка інформаційних бітів символів повідомлення. Приклад такої перестановки представлений нижче.

Таблиця 1 – Приклад перестановки символів слова “Steganograpghy”, представленого у двійковому вигляді

Оригінальний символ		Символ після перестановки	
S	01010011	00110101	5
t	01110100	01111000	x
e	01100101	01011001	Y
g	01100111	01011101	]
a	01100001	00011001	
n	01101110	01011110	^
o	01101111	01011111	_
g	01100111	01011101	]
r	01110010	00111100	<
a	01100001	00011001	
p	01110000	00111000	8
h	01101000	00011010	
y	01111001	00111011	;

З допомогою такого попереднього шифрування повідомлення підвищується стійкість методу до детектування. Далі виконується вбудовування та вилучення повідомлення методом заміни НЗБ.

Програмна реалізація

Функція конвертації тексту в бінарний вигляд:

```
function convertTextToBinaryArray(input) {
    let output = [];
    for (let i = 0; i < input.length; i++) {
        let tmp = input[i].charCodeAt(0).toString(2);
        tmp = make8bit(tmp);
        tmp = tmp.split('');
        for (let j = 0; j < tmp.length; j++) {
            tmp[j] = parseInt(tmp[j], 2);
        }
        output.push(tmp);
    }
  return output;
}
```

Функція конвертації бінарного тексту в символьний вигляд:

```
function convertBinaryToTextArray(str) {
   str = str.match(/.{1,8}/g).toString().replace(/,/g, '
').split(' ');
    let tmp = [];
    for (let i = 0; i < str.length; i++) {
        tmp.push((str[i].split('')).map(item => parseInt(item)));
    }
    let output = shiftInputBits(tmp, TransMatrix(keyMatrix));
    output = output.match(/.{1,8}/g).toString().replace(/,/g, '
').split(' ');
```

```
    let binCode = [];
    for (let i = 0; i < output.length; i++) {
        binCode.push(String.fromCharCode(parseInt(output[i],
2)));
    }
  return binCode.join('');
}
```

Функція перестановки символів повідомлення:

```
function shiftInputBits(input, matrix) {
    let res = [];
    for (let i = 0; i < input.length; i++) {
        let tmp = vectorOnMatrix(input[i], matrix);
        res.push(tmp);
    }
    return res.join('').replace(/,/g, '');
}
```

Функція множення вектору на матрицю:

```
function vectorOnMatrix(vector, matrix) {
    let result = [];
    for (let i = 0; i < matrix.length; i++) {
        let tmp = 0;
        for (let j = 0; j < matrix[i].length; j++) {
            tmp += vector[j] * matrix[i][j];
        }
        result.push(tmp);
    }
    return result;
}
```

Функція транспонування матриці:

```
function TransMatrix(matrix) {
    let m = matrix.length, n = matrix[0].length, matrixTrans =
[];
    for (let i = 0; i < n; i++) {
        matrixTrans[i] = [];
        for (let j = 0; j < m; j++)
            matrixTrans[i][j] = matrix[j][i];
    }
    return matrixTrans;
}
```

Функція кодування зображення:

```
let encode = function () {
    let imageData = ctx.getImageData(0, 0, width, height);
    let encodeImageData = arrayToBinary(imageData.data);
    let input = document.getElementById("message").value;
    input = shiftInputBits(convertTextToBinaryArray(input),
keyMatrix);
    let array = '';
    for (let i = 0, index = 0; i < input.length * 4; i++) {
```

```
        if (i % 4 === 0) {
            encodeImageData[i] = replaceChars(encodeImageData[i],
LSB, input[index]);
            array += input[index];
            index++;
        }
    }
    encodeImageData = parseFromBinary(encodeImageData);
    imageData.data.set(encodeImageData);
    ctx.putImageData(imageData, 0, 0);
    alert("Your data was successfully encoded!");
};
```

Функція декодування зображення:

```
let decode = function () {
    let tmp = document.getElementsByClassName('input-wrapper');
    console.log(tmp[0].children.length);
    let output;
    let decodeImageData = arrayToBinary(ctx.getImageData(0, 0,
width, height).data);
    let decodedText = '';
    for (let i = 0; i < decodeImageData.length; i += 4) {
        decodedText += getChar(decodeImageData[i], LSB);
    }
    output.innerText += convertBinaryToTextArray(decodedText);

};
```

Реалізація алгоритмів вбудовування та вилучення повідомлень методом псевдовипадкового інтервалу

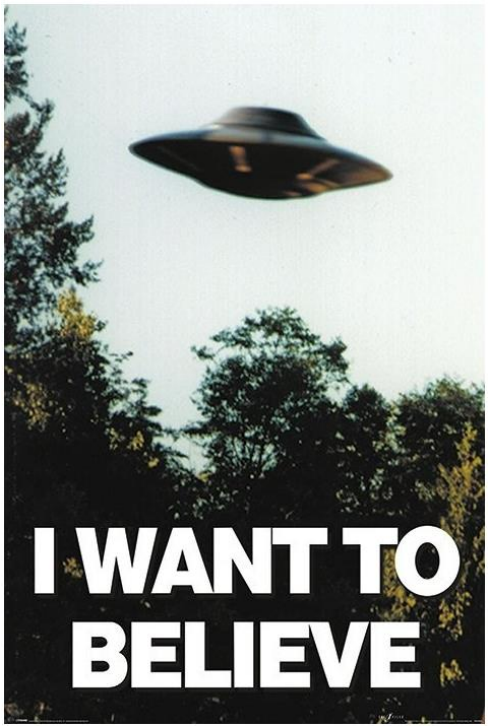


Рисунок 5 – Початкове зображення

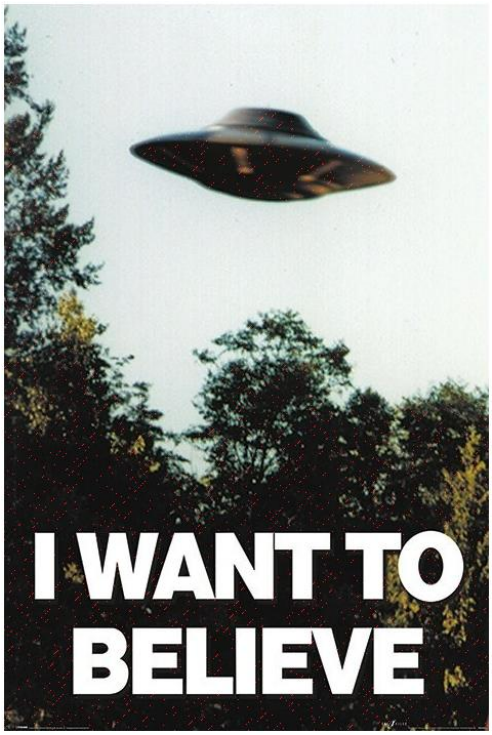
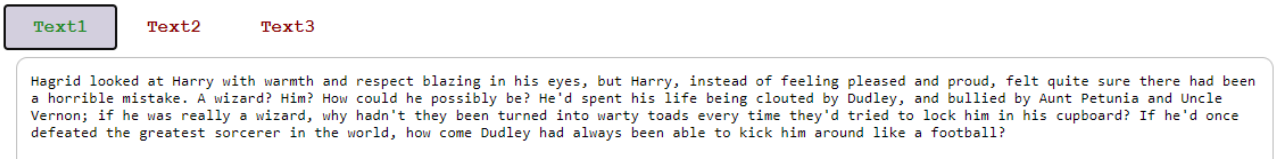


Рисунок 6 – Область, у яку буде проведено запис бітів повідомлення:

Повідомлення для приховування:



Для вбудовування необхідно ввести секретний ключ (рис. 7).

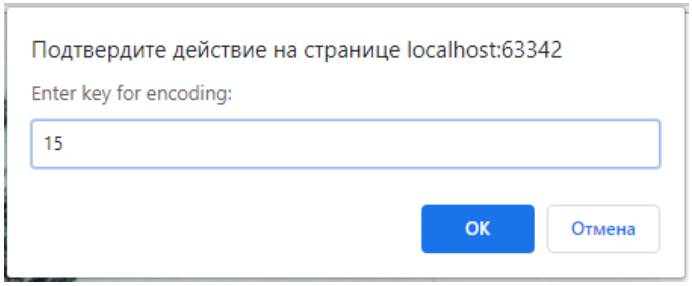


Рисунок 7 – Введення секретного ключа для кодування зображення

Дане значення бере участь в розрахунку інтервалів між пікселями, в які будуть вбудовані біти повідомлення, тому несанкціонований користувач без знання ключа не зможе вірно вилучити інформаційні біти. У даному методі реалізовано запис бітів повідомлення у пікселі, вибрані псевдовипадковим чином. На рис.6 відображені біти, в які буде виконано вбудовування інформаційних бітів.

Для вилучення вводиться секретний ключ (рис. 8).

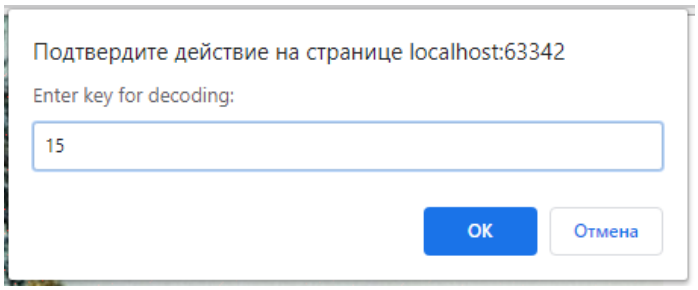


Рисунок 8 – Введення секретного ключа для вилучення повідомлення

Вилучене повідомлення:

```
Hagrid looked at Harry with warmth and respect blazing in his eyes, but Harry, instead of feeling pleased and proud, felt quite sure there had been
a horrible mistake. A wizard? Him? How could he possibly be? He'd spent his life being clouted by Dudley, and bullied by Aunt Petunia and Uncle
Vernon; if he was really a wizard, why hadn't they been turned into warty toads every time they'd tried to lock him in his cupboard? If he'd once
defeated the greatest sorcerer in the world, how come Dudley had always been able to kick him around like a football?
```

При введенні неправильного ключа вилучене повідомлення не має сенсу:

```
▯▯u▯Y▯®a!▯Ó¦íuÊ{▯$▯▯▯¾[³▯OK^-åu£Y<▯ño▯^ðv▯b▯▯Ħ▯HW▯A▯JÔ¡Ó¡▯▯▯PB+ü'Ú¼▯^+&Ñ^Ë▯▯▯iÜ▯×▯▯▯Å¿g▯;k,"Jµ
ù▯Û ´ÇÙË▯▯ÒÝ▯Ãy▯^Ú▯▯±Yk1▯▯he²TÛf¹y▯"ø▯2À±▯g▯ÿÙåÅ1▯PÂkJNÀ▯ò▯▯tBjñ▯ÿñ¼üãþ9_Í▯dáÄ▯ð▯▯▯▯▯▯▯▯Ñò▯ok¢ñ▯33íø▯▯▯È▯³▯ø▯o▯P▯Dk_$å▯▯▯ß^P¦▯▯Y▯#\ýú®▯|%ÌG\R%OzÝ§Î,Õ
Ç▯C▯▯3®5▯Ñé3    QB▯^▯r®&▯▯zs▯▯å▯KR▯æ▯)[LP(▯s▯²¾£▯^Æõ7^ÃÝ▯▯&▯▯uPcF▯Ùiky8k▯▯▯=▯ò~%'▯ý²fuR▯[!à(¦▯èw´▯hÐÅ▯o+ÚÅ▯Ü/▯u▯▯ÓfD a▯▯QV▯▯t.b~A9▯▯å▯▯ÉgO◊JVÍYbÊó-
z±▯▯▯▯▯hÆ«▯▯▯▯¬▯▯=¶wãâÎ▯▯h▯▯ê&ð*7Tì▯▯R§▯▯¬TcCÚ▯▯u▯Ì▯▯ò◊!dW▯?¢▯´ PGP▯&▯▯Êî²ì³¾▯▯l CÍd#C▯I▯µyÒú8)VJxb▯Ö(Å▯ß²▯cr▯ y)Lë▯0è^c¸Ä▯²>6▯?▯'▯▯F▯.1▯Å¢    ▯
ãÖHÖ:º▯²|w◊nÅ▯'é#AçÏ
```

Програмна реалізація

Функція розрахунку інтервалу між пікселями:

```
function step(x, k) {
    let count = 0;
    for (let i = 0; i < x.length; i++) {
        if (x[i] === '1') {
            count++
        }
    }
    return k * count;
}
```

Функція вбудовування інформаційних бітів у пікселі зображення:

```
function putPixelsInInterval(arr, st, key, input, imageDataArr) {
    for (let i = 0, index = 0; i < imageDataArr.length; i++) {
        if (i === st) {
            arr[i] = replaceChars(arr[i], LSB, input[index]);
            st += step(i.toString(2), key);
            index++;
        }
        if (index === input.length) {
            break;
        }
    }
    return arr;
}
```

Функція вилучення інформаційних бітів з пікселів зображення:

```
function getBitsInIntervals(arr, st, key, inputLength,
imageDataLength) {
    let output = '';
    for (let i = 0, index = 0; i < imageDataLength; i++) {
        if (i === st) {
            output += getChar(arr[i], LSB);
            st += step(i.toString(2), key);
            index++;
        }
        if (i > arr.length) {
            console.log(i, arr[i]);
            break;
        }
        if (index === inputLength) {
            break;
        }
    }
  return output;
}
```

Функція кодування зображення:

```
let encode = function () {
    let key = parseInt(prompt('Enter key for encoding: ', '7'));
    let input = document.getElementById("message").value;
    input = convertTextToBinary(input);
    let binaryLength = intToBinary(input.length);
    let imageData = ctx.getImageData(0, 0, width, height);
    let encodeImageData = arrayToBinary(imageData.data);
    let redChannel =
arrayToBinary(getRedChannel(imageData.data));
    let k = encodeImageData.length / 10000 / key;
    k = parseInt(k.toString().split('.')[0]);
    redChannel = putPixelsInInterval(redChannel, startMark, k,
input, redChannel);
    for (let i = 0, index = 0; i < encodeImageData.length; i++) {
        if (i % 4 === 0) {
            encodeImageData[i] = redChannel[index];
            index++;
        }
    }
    for (let i = 0; i < binaryLength.length; i++) {
        encodeImageData[i] = replaceChars(encodeImageData[i],
LSB, binaryLength[i]);
    }
    encodeImageData = parseFromBinary(encodeImageData);
    imageData.data.set(encodeImageData);
    ctx.putImageData(imageData, 0, 0);
    alert("Your data was successfully encoded!");
};
```

Функція декодування зображення:

```
let decode = function () {
    let key = parseInt(prompt('Enter key for decoding: ', '7'));
    let tmp = document.getElementsByClassName('input-wrapper');
    let decodeImageData = arrayToBinary(ctx.getImageData(0, 0,
width, height).data);
    let redChannel = getRedChannel(decodeImageData);
    let k = decodeImageData.length / 10000 / key;
    k = parseInt(k.toString().split('.')[0]);
    let inputLength = '';
    for (let i = 0; i < 24; i++) {
        inputLength += getChar(decodeImageData[i], LSB);
    }
    inputLength = parseInt(inputLength, 2);
    let res = getBitsInIntervals(redChannel, startMark, k,
inputLength, decodeImageData.length);
    output.innerText += convertBinaryToText(res);
};
```

Висновки

При виконанні лабораторної роботи було реалізовано три методи приховування даних у просторовій області нерухомих зображень шляхом модифікації найменш значущого біта даних: класичний з заміною наймолодшого біта, метод псевдовипадкового інтервалу та метод псевдовипадкової перестановки. Було практично виведено, що такі методи можливі за рахунок низької чутливості зорової системи людини до незначних змін яскравості.