Міністерство освіти і науки України
Харківський національний університет ім. В. Н. Каразіна
Факультет комп’ютерних наук
Кафедра безпеки інформаційних систем і технологій

Лабораторна робота №3

з навчальної дисципліни

«Стеганографія»

Виконала:

Студентка групи КБ-41
Кононченко А. В.

Перевірив:

Доцент
Нарєжній О. П.

Харків – 2020

Лабораторна робота №3

на тему:

«Приховування даних у просторовій області нерухомих зображень на основі прямого розширення спектру»

Мета роботи: закріпити теоретичні знання за темою «Приховування даних у просторовій області нерухомих зображень на основі прямого розширення спектру», набути практичних вмінь та навичок щодо розробки стеганографічних систем, дослідити властивості стеганографічних методів, що засновані на низькорівневих властивостях зорової системи людини (ЗСЛ).

Хід роботи

1 Реалізація алгоритму формування ансамблів ортогональних дискретних сигналів Уолша-Адамара та кодування інформаційних бітів даних складними дискретними сигналами

Формування матриці Адамара відбувається наступним чином:

```
function generateHadamardMatrix(n) {
    let size = parseInt(Math.pow(n, 2).toString());
    let hadamard = matrix(size, size);
    hadamard[0][0] = 1;
    for (let k = 1; k < size; k += k) {
        for (let i = 0; i < k; i++) {
            for (let j = 0; j < k; j++) {
                hadamard[i + k][j] = hadamard[i][j];
                hadamard[i][j + k] = hadamard[i][j];
                hadamard[i + k][j + k] = -hadamard[i][j];
            }
        }
    }
    return hadamard;
}
```

На виході отримуємо:

(i…	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1
2	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1
3	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1
4	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1
5	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1
6	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1
7	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1
8	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1
9	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1
10	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1
11	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1
12	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1
13	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1
14	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1
15	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1

Рисунок 1.1 – Сформована матриця Адамара

Дискретні сигнали Уолша-Адамара формуються з рядків матриці Адамара.

Інформаційні біти представляються у вигляді полярному вигляді:

```
function transformToSignal(str) {
    let inputArr = [];
    for (let i = 0; i < str.length; i++) {
        inputArr.push(parseInt((str[i] === '0' ? -1 : 1).toString()));
    }
    return inputArr;
}
```

На виході отримуємо:

(index)	Value
0	"0"
1	"1"
2	"0"
3	"1"
4	"0"
5	"0"
6	"1"
7	"1"
8	"0"
9	"1"
10	"1"
11	"1"
12	"0"
13	"1"
14	"0"
15	"0"
16	"0"

Рисунок 1.2 – Бінарний вигляд повідомлення

(index)	Value
0	-1
1	1
2	-1
3	1
4	-1
5	-1
6	1
7	1
8	-1
9	1
10	1
11	1
12	-1
13	1
14	-1
15	-1
16	-1

Рисунок 1.3 – Полярний вигляд повідомлення

Кодування складними дискретними сигналами відбувається таким чином:

```
function sum(m, k, hadamard) {
    let sum = new matrix(hadamard.length, hadamard.length);
    for (let i = 0; i < hadamard.length; i++) {
        let a = new myArray(hadamard.length);
        for (let j = 0; j < k; j++) {
            let arr = [];
            arr.length = hadamard.length;
            arrayCopy(hadamard[j + 1], 0, arr, 0, hadamard.length);
            for (let l = 0; l < arr.length; l++) {
                arr[l] *= (((i * k + j) < m.length)?m[i * k + j]*g : 0);
                a[l] += arr[l];
            }
        }
        arrayCopy(a, 0, sum[i], 0, a.length);
    }
    return sum;
}
```

На виході:

(i…	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	12	-4	8	0	12	-4	-8	-4	0	0	4	-4	0	0	-12	0
1	0	4	12	0	8	-4	4	-8	-4	8	0	-4	4	0	-8	-12	0
2	0	12	-12	0	-8	4	-4	-8	4	8	0	4	-4	0	8	-4	0
3	0	4	4	-8	0	4	-12	-8	4	0	16	-4	4	0	0	-4	0
4	4	0	-16	-4	-4	8	-8	4	8	-4	-4	0	0	4	4	8	4
5	8	-4	12	0	0	-12	4	8	4	0	0	-4	-4	-8	-8	4	8
6	4	8	-8	12	-4	0	0	4	8	4	-12	0	0	-4	-4	-8	4
7	-4	0	8	-4	-4	0	-8	-4	0	-4	20	0	0	-4	4	0	-4
8	-4	8	-8	4	4	0	-16	-4	0	4	4	8	8	-4	-4	0	-4
9	0	-4	4	0	16	-4	4	0	-12	-8	0	4	4	-8	0	4	0
10	-8	-4	4	8	-8	-4	4	-8	4	0	8	4	4	0	8	-12	-8
11	-4	0	0	4	-12	8	-8	-4	0	-4	12	8	-8	4	4	0	-4
12	0	4	-4	16	0	4	-4	0	4	0	-8	4	4	0	-8	-12	0
13	-4	0	8	-4	12	0	8	-4	-16	-4	4	0	0	-4	4	0	-4
14	-4	0	-8	-4	-4	0	8	-4	0	12	-12	0	0	12	4	0	-4
15	-4	8	8	4	-4	8	-8	4	-8	-4	12	0	-8	-4	-4	0	-4
16	-12	0	-8	4	4	0	-8	4	-8	-4	4	8	8	-4	4	8	-12

Рисунок 1.4 – Результат кодування складними дискретними сигналами

2 Реалізація алгоритмів приховування та вилучення даних шляхом прямого розширення спектрів із використанням ортогональних дискретних сигналів

2.1 Реалізація алгоритму вбудовування інформаційних даних в просторову область зображення на основі прямого розширення спектру із використанням ортогональних дискретних сигналів Уолша-Адамара

Вбудовування полягає в підсумовуванні даних контейнера з модульованими складними дискретними сигналами інформаційного повідомлення:

```
function encode() {
    let input = document.getElementById("message").value;
    input = convertTextToBinary(input);
    let imageData = ctx.getImageData(0, 0, width, height);
    let n = parseInt(log2(Math.min(height, width)).toString());
    let hadamard = generateHadamardMatrix(n);
    input = transformToSignal(input)
    let tmp = sum(input, k, hadamard);
    let imagePixels = dividePixels(imageData.data);
    for (let i = 0; i < hadamard.length; i++) {
        for (let j = 0; j < hadamard.length; j++) {
            let color = getRGB(imagePixels, i * hadamard.length + j);
            let newRed = color[0] + tmp[i][j];
            if (newRed > 255) newRed = 255;
            if (newRed < 0) newRed = 0;
            imagePixels = setRGB(imagePixels, i * hadamard.length + j,
newRed, color[1], color[2]);
        }    }
    imagePixels = mergePixels(imagePixels);
    for (let i = 0; i < imageData.data.length; i++) {
        imageData.data[i] = imagePixels[i];
    }    ctx.putImageData(imageData, 0, 0);
}
```

Порівняння зображення до і після вбудовування інформаційних даних:

Рисунок 2.1.1 – Зображення до вбудовування

Рисунок 2.1.2 – Зображення після вбудовування

2.2 Реалізація алгоритму вилучення даних з просторової області зображень на основі прямого розширення спектру

Функція отримання масиву рядків з вбудованими інформаційними бітами:

```
let arrayString = []; //array of divided RED colors
   for (let i = 0; i < hadamard.length; i++) {
      let arrRed = [];//array of RED
      for (let j = 0; j < hadamard.length; j++) {
         arrRed.push(getRGB(decodeImageDataArr,i*hadamard.length+j)[0]);
      }
      arrayString.push(arrRed);
   }
```

На виході:

```
▶0: (64) [133, 143, 124, 134, 124, 135, 120, 117, 128, 137, 138, 138, 130, …
▶1: (64) [124, 129, 137, 124, 132, 123, 132, 121, 119, 133, 126, 121, 127, …
▶2: (64) [113, 128, 104, 114, 106, 118, 111, 110, 120, 123, 118, 120, 116, …
▶3: (64) [114, 119, 114, 101, 111, 113, 93, 98, 111, 106, 120, 99, 108, 105…
▶4: (64) [114, 110, 94, 106, 106, 118, 101, 113, 118, 106, 103, 106, 110, 1…
▶5: (64) [110, 99, 116, 103, 103, 93, 108, 110, 109, 102, 107, 106, 100, 93…
▶6: (64) [103, 105, 86, 107, 93, 94, 94, 104, 101, 94, 83, 96, 95, 87, 91, …
▶7: (64) [82, 91, 103, 90, 89, 93, 85, 88, 93, 90, 109, 85, 87, 82, 88, 88,…
▶8: (64) [82, 96, 80, 91, 91, 88, 70, 78, 84, 89, 87, 94, 91, 82, 80, 85, 8…
▶9: (64) [91, 88, 97, 92, 108, 89, 98, 94, 82, 83, 93, 95, 99, 85, 95, 98, …
▶10: (64) [122, 126, 134, 139, 124, 125, 134, 121, 132, 129, 138, 133, 131,…
▶11: (64) [114, 116, 118, 121, 108, 124, 111, 110, 117, 109, 121, 118, 104,…
▶12: (64) [108, 112, 107, 126, 113, 120, 109, 111, 119, 114, 105, 118, 111,…
▶13: (64) [106, 111, 119, 108, 124, 109, 118, 104, 92, 106, 113, 109, 113, …
```

Рисунок 2.2.1 – Отриманий масив зі вбудованими бітами повідомлення

Функція вилучення полягає в зіставленні результату обчислення коефіцієнта кореляції з граничним значенням «0»:

```
let outputSignal = [];
for (let i = 0; i < hadamard.length; i++) {
  for (let j = 0; j < k; j++) {
   outputSignal.push(multString(arrayString[i],hadamard[j+1]) > 0
? '1' : '0');
  }
}
```

На виході:

```
(640) ["0", "1", "0", "1", "0", "0", "1", "1", "0", "1", "1", "1", "0", "1",
"0", "0", "0", "1", "1", "0", "0", "1", "0", "1", "0", "1", "1", "0", "0",
"1", "1", "1", "0", "1", "1", "0", "0", "0", "0", "1", "0", "1", "1", "0",
"1", "1", "1", "0", "0", "1", "1", "0", "1", "1", "1", "1", "0", "1", "1",
"0", "0", "1", "1", "1", "0", "1", "1", "1", "0", "0", "1", "0", "0", "1",
"1", "0", "0", "0", "0", "1", "0", "1", "1", "1", "0", "0", "0", "0", "0",
"1", "1", "0", "1", "0", "0", "0", "0", "0", "1", "1", …]
```

Рисунок 2.2.2 – Вилучені інформаційні біти

Оригінальне повідомлення:	`Steganography is the technique of hiding secret data`
Вилучене повідомлення:	`Steganograph9 is the technique of hhding secret data`

3 Експериментальні дослідження ймовірнісних властивостей реалізованого методу

Для дослідження ймовірнісних властивостей реалізованого методу було проведено експериментальні дослідження зі зміною параметрів k – кількість вбудованих інформаційних бітів в один рядок контейнера, та g – коефіцієнту посилення, який задає «енергію» біта, що вбудовується:

- розраховано відносну похибку вилучення інформаційних бітів при фіксованому $g=1$

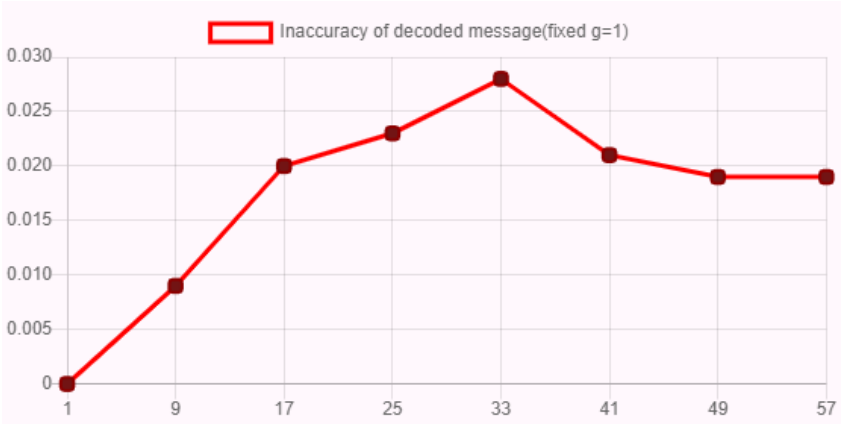


Рисунок 3.1 – Графік залежності помилкового вилучення інформаційних бітів при зміні значення k

- розраховано відносну частку внесених у контейнер спотворень при фіксованому $g=1$

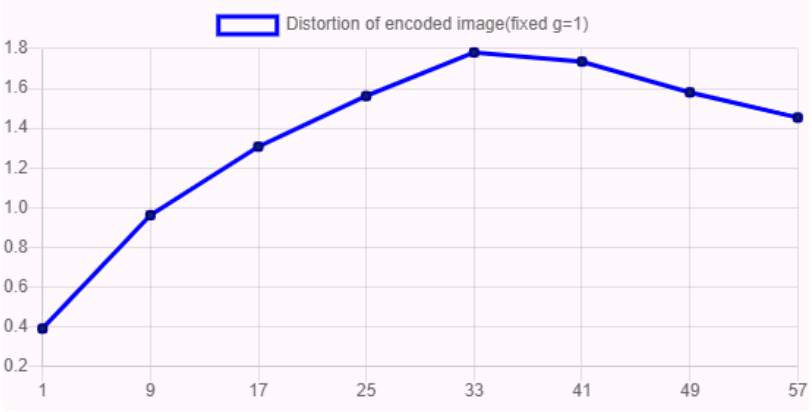


Рисунок 3.2 – Графік залежності частки внесених у зображення спотворень при зміні значення k

- розраховано відносну похибку вилучення інформаційних бітів при фіксованому $k=4$

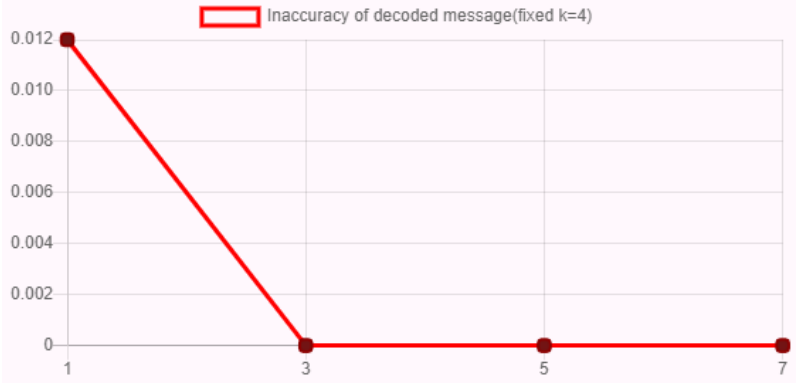


Рисунок 3.3 – Графік залежності помилкового вилучення інформаційних бітів при зміні значення g

- розраховано відносну частку внесених у контейнер спотворень при фіксованому $k=4$

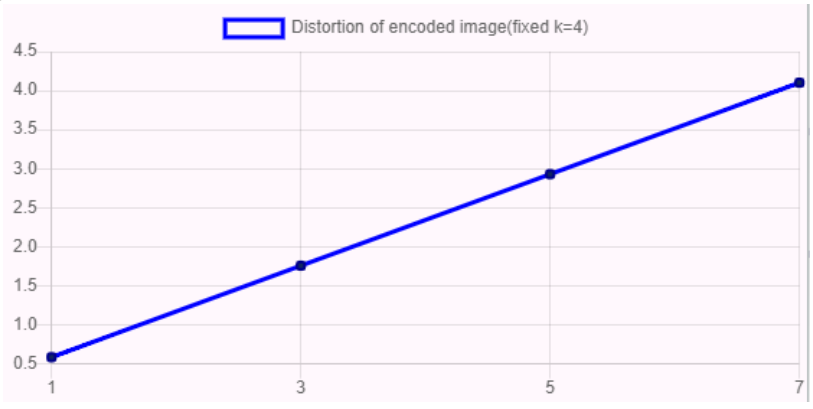


Рисунок 3.4 – Графік залежності частки внесених у зображення спотворень при зміні значення g

4 Реалізація алгоритму формування квазіортогональних дискретних сигналів

4.1 Реалізація алгоритму приховування даних в просторовій області зображень із використанням квазіортогональних дискретних сигналів

Ансамбль квазіортогональних дискретних сигналів формується за допомогою наступної функції:

```
function arrayFunction() {
    let resArr = [];
    for (let i = 0; i < ansamblHeight; i++) {
        let a = [];
        for (let j = 0; j < ansamblWidth; j++) {
            let b = getRndInteger(0, 1);
            a.push(b === 0 ? -1 : 1)
        }
        resArr.push(a);
    }
    return resArr;
}
```

На виході:

(i…	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1	1	1	1	-1	-1	1	1	1	1	1	-1	1	-1	-1	-1	1	-1	-1	-1
1	1	1	-1	1	1	1	-1	1	1	-1	1	1	1	-1	1	1	-1	1	1	1
2	-1	1	-1	-1	1	1	1	-1	1	-1	-1	-1	-1	1	1	1	-1	1	-1	-1
3	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	1	1	-1	1	-1	1	1	-1
4	1	-1	1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	1	1	-1	1
5	-1	-1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1
6	1	-1	1	1	-1	1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	1
7	1	-1	1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	1	-1	-1
8	-1	1	1	-1	-1	-1	1	-1	1	-1	1	1	1	-1	1	-1	-1	1	-1	-1
9	-1	1	1	1	-1	1	-1	1	1	-1	-1	-1	1	-1	1	1	1	-1	1	1
10	-1	-1	1	-1	-1	1	-1	1	1	1	1	1	1	1	-1	-1	1	-1	1	-1
11	-1	1	-1	1	-1	1	1	1	-1	1	-1	-1	-1	-1	1	-1	1	-1	-1	1
12	1	1	1	1	1	1	-1	1	1	-1	1	1	-1	-1	1	-1	-1	-1	-1	-1
13	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	-1	-1	-1	1	1	1	-1	1
14	1	1	-1	1	1	-1	1	1	-1	-1	1	1	1	-1	1	1	1	-1	1	1
15	-1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	-1	-1	-1	1
16	-1	1	1	1	1	-1	-1	1	-1	-1	-1	-1	1	-1	-1	1	-1	-1	1	-1

Рисунок 4.1.1 – Сформований ансамбль квазіортогональних дискретних сигналів

Для вбудовування інформаційних повідомлень з використанням сформованого ансамблю квазіортогональних дискретних сигналів бітовий масив секретного повідомлення розбивається на підблоки, які перетворюються у десяткові числа (рис. 4.1.2). Функція формування масиву:

```
function split10blockMessage(arr) {
    let resDesArr = [];
    for (let i = 0; i < arr.length / 10; i++) {
        let a = 0;
        for (let j = 0; j < 10; j++) {
            if (typeof arr[10 * i + j] === 'undefined') {
                arr[i * 10 + j] = 0;
                continue;
            }
            a += arr[10 * i + j] * parseInt((Math.pow(2,
j).toString()));
        }
        resDesArr.push(a);
    }
    return resDesArr;
}
```

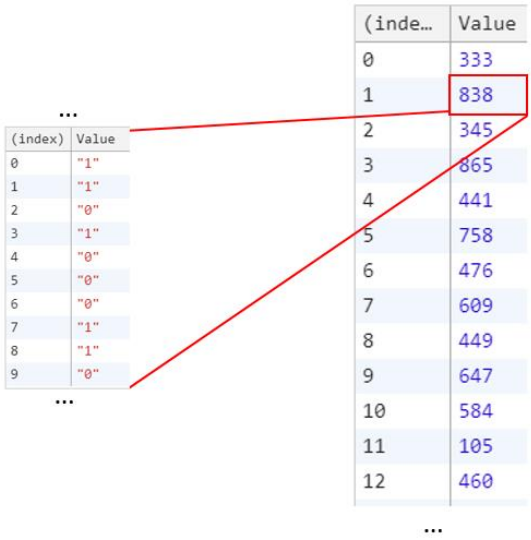


Рисунок 4.1.2 – Фрагмент масиву десяткових чисел, які відповідають блокам довжиною у 10 інформаційних бітів

Алгоритм кодування квазіортогональними дискретними сигналами:

```
function sum(md, g, arrayFunction) {
    let sum = [];
    for (let i = 0; i < height; i++) {
        if (i === md.length) {
            console.log('i === md.length');
            break;
        }
        sum.push(arrayFunction[md[i]].map(item => g * item));
    }
    return sum;
}
```

На виході:

(in…	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	-45	-45	45	45	-45	45	-45	-45	-45	-45	45	45	45	45	45	45	-45	-45	45	45
1	45	45	45	-45	-45	-45	45	-45	45	-45	45	-45	-45	-45	-45	-45	45	45	45	-45
2	45	-45	45	-45	45	-45	45	-45	45	45	-45	45	45	45	45	45	45	45	-45	-45
3	-45	-45	-45	45	-45	-45	-45	-45	45	45	-45	-45	-45	45	-45	-45	45	45	45	-45
4	-45	45	45	-45	-45	45	-45	-45	-45	-45	45	-45	-45	45	45	-45	-45	-45	-45	45
5	45	-45	-45	45	45	-45	-45	45	45	-45	-45	45	45	45	45	-45	45	-45	-45	-45
6	-45	45	45	45	45	45	-45	45	-45	45	-45	-45	45	45	-45	-45	45	45	45	45
7	-45	45	45	45	45	-45	45	-45	-45	-45	45	45	-45	45	-45	-45	45	45	-45	45
8	-45	45	45	-45	-45	45	-45	-45	-45	-45	45	-45	-45	45	45	-45	-45	-45	-45	45
9	-45	-45	-45	45	-45	-45	45	45	45	45	45	-45	-45	-45	45	45	-45	-45	45	-45
10	-45	45	-45	-45	45	45	-45	45	45	45	-45	-45	45	-45	-45	-45	45	-45	45	-45
11	-45	45	-45	-45	45	-45	45	-45	45	-45	45	45	45	45	-45	45	45	-45	45	-45
12	45	45	45	45	45	-45	45	45	45	-45	45	-45	45	45	45	-45	45	-45	45	45
13	45	45	45	-45	-45	45	45	45	45	45	45	45	45	45	45	-45	45	-45	45	45
14	-45	-45	-45	45	45	45	-45	45	45	-45	45	-45	45	-45	-45	-45	-45	45	-45	45
15	45	45	45	45	-45	-45	45	45	45	45	-45	45	45	45	-45	45	-45	-45	-45	45
16	-45	-45	45	45	45	-45	45	-45	45	45	45	-45	-45	45	-45	45	-45	-45	-45	45

Рисунок 4.1.3 – Результат кодування квазіортогональними дискретними сигналами

Алгоритм вбудовування інформаційного повідомлення в контейнер-зображення за допомогою накладення модульованого повідомлення на масив яскравостей каналу червоного кольору:

```
function encode() {
    let input = document.getElementById("message").value;
    input = convertTextToBinary(input);
    let transInput = split10blockMessage(input);
    let sum1 = sum(transInput, g, arrayFunction1);
    let imageData = ctx.getImageData(0, 0, width, height);
    let redChannel = getRedChannel(imageData.data);
    let arrayString = [];
    for (let i = 0; i < height; i++) {
        let a = [];
        for (let j = 0; j < ansamblWidth; j++) {
            a.push(redChannel[i * ansamblWidth + j]);
        }
        arrayString.push(a);
    }
    for (let i = 0; i < height; i++) {
        for (let j = 0; j < width; j++) {
            if (i < sum1.length) {
                let newRed = arrayString[i][j] + sum1[i][j];
                if (newRed > 255) newRed = 255;
                if (newRed < 0) newRed = 0;
                arrayString[i][j] = newRed;
                c++;
            }
        }
```

```
    }
    let newRedChannel = arrayString.flat();
    for (let i = 0, index = 0; i < imageData.data.length; i += 4)
{
        imageData.data[i] = newRedChannel[index]; // red
        index++;
    }
    ctx.putImageData(imageData, 0, 0), 1000);
}
```

Зміни червоного кольору при $g = 45$:

Початкові значення каналу червоного кольору:

```
(518400) [165, 151, 122, 94, 71, 55, 74, 117, 160, 167, 159, 163, 137, 109, 81, 79, 81, 75, 68, 66, 63, 55, 6
0, 59, 59, 52, 46, 52, 67, 78, 82, 92, 92, 104, 28, 34, 36, 27, 53, 75, 27, 0, 26, 0, 35, 12, 29, 45, 59, 40,
▶ 54, 47, 30, 35, 25, 44, 44, 60, 44, 22, 19, 28, 1, 36, 0, 72, 39, 126, 48, 48, 63, 54, 47, 54, 52, 53, 57, 6
4, 69, 103, 121, 101, 182, 156, 117, 117, 159, 182, 176, 179, 182, 168, 151, 150, 120, 112, 150, 178, 176, 13
4, …]
```

Змінені значення каналу червоного кольору:

```
(518400) [120, 106, 167, 139, 26, 100, 29, 72, 115, 122, 204, 208, 182, 154, 126, 124, 36, 30, 113, 111, 108,
10, 105, 14, 104, 97, 91, 7, 22, 123, 127, 47, 47, 149, 73, 0, 81, 0, 8, 30, 0, 0, 71, 45, 80, 0, 0, 0, 14, 8
▶ 5, 9, 92, 75, 80, 70, 89, 89, 105, 89, 67, 0, 73, 0, 81, 0, 117, 84, 171, 93, 93, 108, 99, 2, 9, 7, 8, 102, 1
09, 114, 148, 76, 146, 227, 201, 72, 72, 114, 227, 221, 134, 227, 213, 106, 105, 75, 67, 105, 223, 221, 8
9, …]
```

Порівняння зображення до і після вбудовування інформаційних даних:

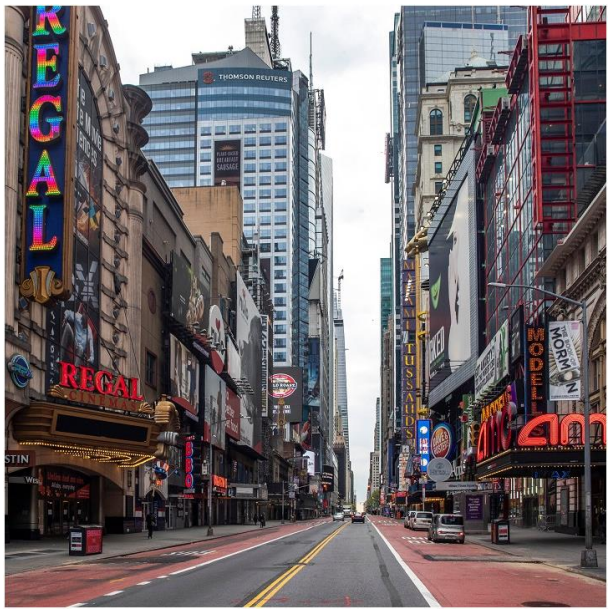


Рисунок 4.1.4 – Зображення до вбудовування

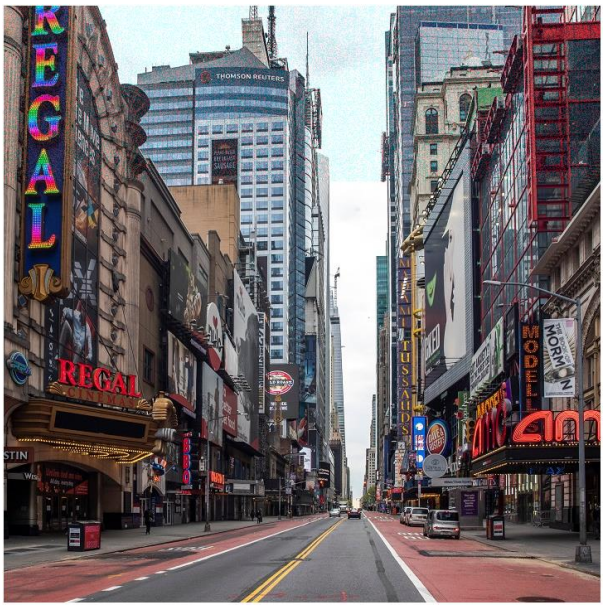


Рисунок 4.1.5 – Зображення після вбудовування

4.2 Реалізація алгоритму вилучення даних в просторовій області зображень із використанням квазіортогональних дискретних сигналів

Функція формування масиву з червоного кольору:

```
let redChannel = getRedChannel(decodeImageData.data);
    let arrayString = [];
    for (let i = 0; i < height; i++) {
        let a = [];
        for (let j = 0; j < ansamblWidth; j++) {
            a.push(redChannel[i * width + j]);
        }
        arrayString.push(a);
    }
```

На виході:

```
[120, 106, 167, 139, 26, 100, 29, 72, 115, 122, 204, 208, 182, 154, 126, 124, 36, 30, 113, 111,…
[228, 224, 217, 120, 99, 62, 116, 7, 112, 71, 213, 110, 121, 109, 85, 40, 120, 119, 116, 20, 24…
[217, 133, 225, 131, 217, 129, 197, 70, 111, 95, 35, 180, 214, 215, 196, 186, 135, 120, 31, 20,…
[137, 132, 131, 220, 126, 130, 129, 119, 194, 146, 16, 3, 58, 207, 125, 108, 190, 134, 116, 21,…
[135, 222, 220, 133, 130, 215, 123, 125, 124, 124, 177, 42, 4, 102, 193, 112, 115, 102, 37, 110…
[223, 136, 129, 214, 216, 129, 130, 222, 205, 128, 120, 196, 144, 101, 100, 67, 207, 112, 89, 3…
[137, 225, 221, 221, 219, 214, 124, 219, 134, 215, 121, 124, 206, 171, 14, 0, 143, 199, 197, 17…
[140, 229, 226, 221, 219, 134, 223, 127, 124, 128, 219, 216, 117, 206, 78, 28, 89, 128, 119, 18…
[143, 230, 226, 137, 139, 224, 128, 132, 133, 124, 216, 126, 119, 210, 209, 76, 34, 1, 42, 192,…
[140, 141, 140, 231, 142, 135, 217, 219, 216, 219, 215, 121, 120, 114, 209, 201, 75, 26, 89, 34…
```

Рисунок 4.2.1 – Сформований масив червоного кольору

Функція формування масиву десяткових чисел, елементами якого будуть номери квазіотрогональних сигналів з ансамблю, які дають найбільше значення коефіцієнта кореляції з рядками контейнера-зображення:

```
let md1 = [];
    for (let i = 0; i < height; i++) {
        let a = 0;
        for (let j = 0; j < ansamblHeight; j++) {
          if (multString(arrayString[i], arrayFunction1[j]) > a) {
              a = multString(arrayString[i], arrayFunction1[j]);
              md1[i] = j;
          }
        }
    }
```

На виході:

```
(720) [578, 939, 66, 584, 516, 941, 234, 187, 516, 937, 225, 216, 646, 421, 71, 184, 246, 897, 873, 699, 516,
389, 751, 19, 646, 397, 67, 184, 534, 933, 76, 728, 662, 947, 98, 858, 678, 285, 352, 154, 678, 289, 864, 41
1, 516, 393, 874, 985, 686, 413, 364, 474, 230, 897, 864, 664, 646, 947, 234, 665, 516, 417, 615, 16, 526, 40
3, 360, 826, 662, 413, 78, 56, 646, 421, 71, 952, 646, 307, 96, 985, 590, 285, 96, 474, 38, 129, 73, 952, 66
2, 397, 67, 920, 662, 411, 74, 632, 758, 299, 96, 18, …]
```

Рисунок 4.2.2 – Вилучений масив десяткових чисел

Функція приведення отриманого масиву десяткових чисел до двійкового виду:

```
let messBinary = [];
    for (let i = 0; i < md1.length; i++) {
        let a = md1[i];
        for (let j = 0; j < 10; j++) {
            messBinary[i * 10 + j] = Math.trunc(a % 2);
            a /= 2;
        }
    }
    console.log(messBinary.join(''));
```

На виході:

```
0100001001110101011101000010000001001001001000000110110101110101011100110111010 steganoKvaziOrthogonal.js:157
0010000001100101011110000111000001101100011000010110100101101110001000000111010001101111001000000111100101101111
1011101010010000001101000011011110111011100100000011000010110110001101100001000000111010001101000011010010111001
1100100000011011010110100101110011011101000110000101101011011000101011011100010000001101001011001000110010101100
0010010000001101111011001100010000001100100011001010110111001101111011101011011100110001100111011010010111001100110
0111001000000111000001101100011001010110000101110011011101010111001001100101001000000110000101101110011001000001
0000001110000011100100110000101101001011100110110100101101110011001110010000001110000011000010110100101011100000
1000000111011101100001011100110010000001100010011011110111001001101110001000000110000101101110011001000010000000
1001001001000000111011101101001011011000110110000100000011001110110100101110110011001010010000001111001011011111
0111010100100000011000010010000001100011011011110110101011100000110110001100101011101000110010100100000011000000
1011000110110001101101111011101010110111001110100001000000110111101100110001000000111010001101000011001010010000
0001110011011110010111001101110100011001010110110100101100001000000110000101101110011001000010000001100101011111
0000111000001101111011101010110111001100100001000000111010001101000011001010010000001100001011000011011101000111
```

Рисунок 4.2.3 – Фрагмент вилученого повідомлення у двійковому вигляді

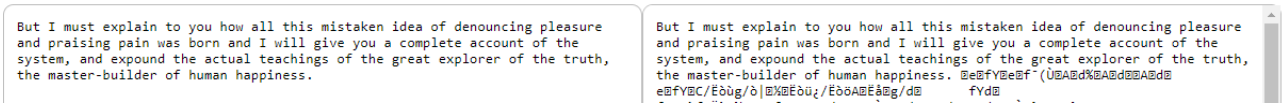
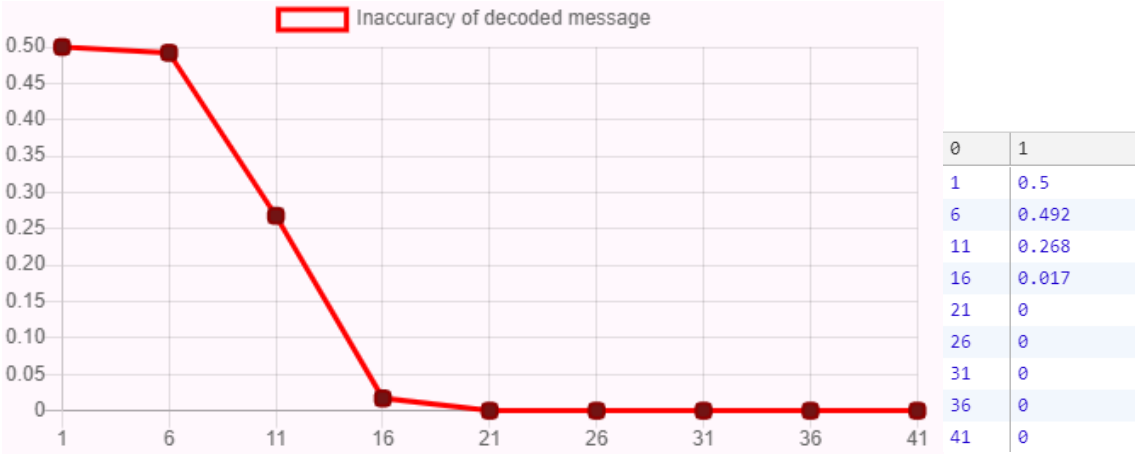


Рисунок 4.2.4 – Результат вилучення повідомлення із зображення

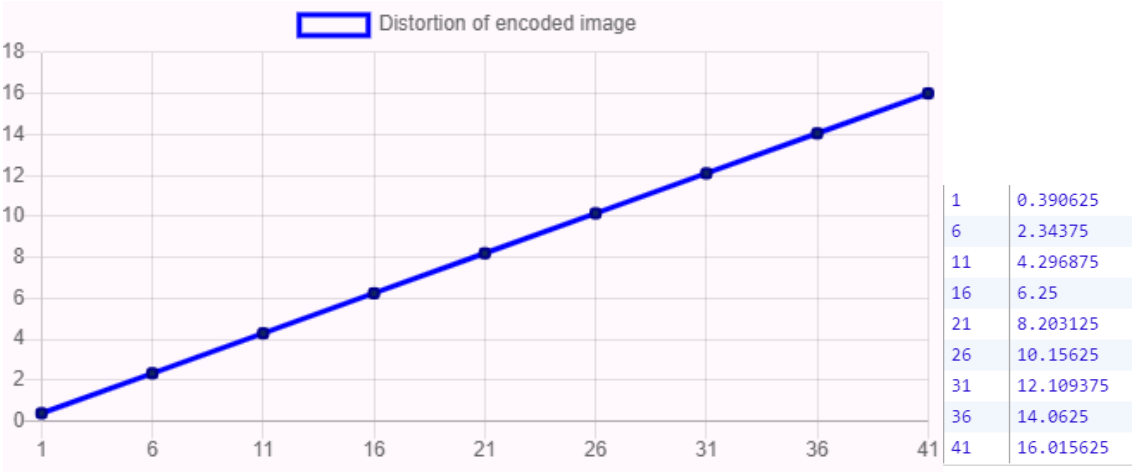
4.3 Оцінка ймовірності правильного вилучення повідомлення та обсягу внесених спотворень від коефіцієнта посилення

Для оцінки ймовірності правильного вилучення повідомлення та обсягу внесених спотворень від коефіцієнта посилення було розраховано частоту помилково отриманих інформаційних бітів і оцінку внесених спотворень в контейнер-зображення. Графіки представлені на рис. 4.3.1 - 4.3.2.



0	1
1	0.5
6	0.492
11	0.268
16	0.017
21	0
26	0
31	0
36	0
41	0

Рисунок 4.3.1 – Ймовірність помилкового вилучення повідомлення



1	0.390625
6	2.34375
11	4.296875
16	6.25
21	8.203125
26	10.15625
31	12.109375
36	14.0625
41	16.015625

Рисунок 4.3.2 – Оцінка внесених спотворень у зображення

З приведених графіків видно, що з підвищенням коефіцієнта підсилення зменшується ймовірність помилкового вилучення інформаційних бітів, але збільшується частка внесених спотворень.

Висновки

При виконанні лабораторної роботи було реалізовано стеганографічні методи приховування даних на основі прямого розширення спектру. Було розроблено алгоритми приховування та вилучення даних із просторової області нерухомих зображень шляхом прямого розширення спектру із використанням ортогональних дискретних сигналів та квазіортогональних дискретних сигналів., а також проведено експериментальні дослідження ймовірнісних характеристик реалізованих методів.