ГУАП

Институт киберфизических систем
Кафедра ИБ

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

доц., канд. техн. наук		А.В. Афанасьева
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

Внедрение информации в пространственную область

по курсу: Технологии стеганографии в системах инфокоммуникаций

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 3931		В.С. Кулешов
	подпись, дата	инициалы, фамилия

Санкт-Петербург, 2022 г.

1 Цель работы

Целью данной лабораторной работы является разработка стегосистемы на основе внедрения информации в изображение в формате *.bmp* методом блочного скрытия.

2 Ход выполнения работы

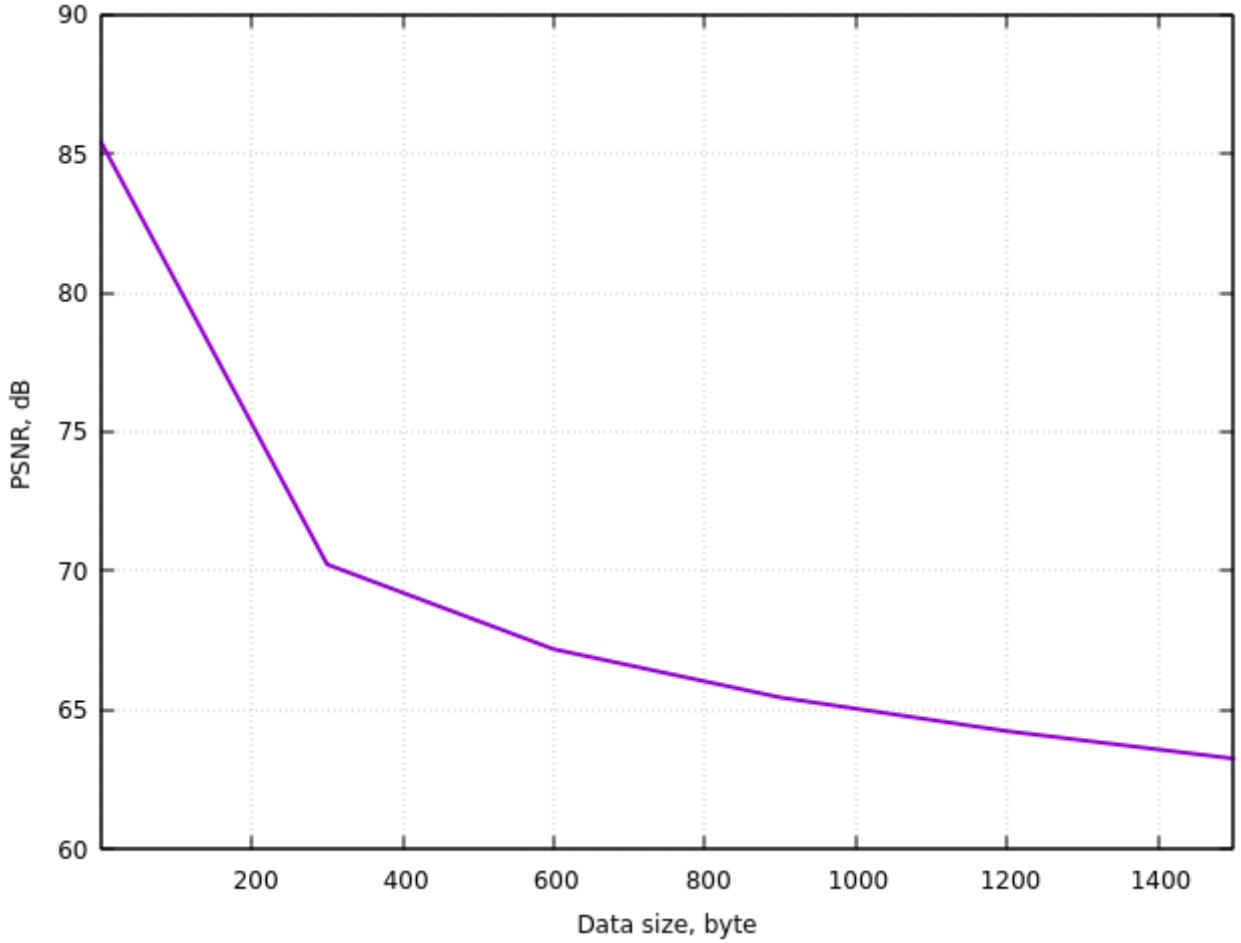
Алгоритм закрытия информации:

1. Данные представляются в виде бинарной последовательности;
2. Изображение-контейнер разбивается на блоки 8x8;
3. Последовательно проходя по блокам, а внутри их по компонентам цвета, высчитывается бит четности. Если бит четности не совпадает с очередным битом бинарной последовательности входных данных, изменяется четность случайного значения в блоке.

Алгоритм раскрытия информации:

1. Изображение-контейнер разбивается на блоки 8x8;
2. Для каждого очередного блока изображения высчитывается бит четности, который помещается в контейнер типа FIFO;
3. По достижении размера контейнера длины закрытого сообщения обработка прекращается.

2. 1 Влияние на качество изображения



Санкт-Петербург, 2022 г.

3 Вывод

В результате выполнения лабораторной работы была реализована стегосистема в соответствии с предъявленными требованиями.

Санкт-Петербург, 2022 г.

4 Листинг кода

```
void
BMPContainer::Cell::set(bool v)
{
        bool sum = false;

        for (int i = 0; i < BLOCK_SIZE; ++i)
            for (int j = 0; j < BLOCK_SIZE; ++j)
                switch (color_) {
                case 0:
                    sum ^= pic.GetPixel(y_ + j, x_ + i).Red & 1;
                    break;

                case 1:
                    sum ^= pic.GetPixel(y_ + j, x_ + i).Green & 1;
                    break;

                case 2:
                    sum ^= pic.GetPixel(y_ + j, x_ + i).Blue & 1;
                }

        if (sum != v)
        {
            struct {
                int x, y;
            } coord = {(rand() % BLOCK_SIZE), (rand() % BLOCK_SIZE)};

            auto pxl = pic.GetPixel(y_ + coord.y,
                                    x_ + coord.x);

            switch (color_) {
            case 0:
                pxl.Red ^= 1;
                break;

            case 1:
                pxl.Green ^= 1;
                break;

            case 2:
                pxl.Blue ^= 1;
                break;
            }

            pic.SetPixel(y_ + coord.y,
                         x_ + coord.x,
                         pxl);
        }
}
```

```
bool
BMPContainer::Cell::get()
const
{
        bool sum = false;

        for (int i = 0; i < BLOCK_SIZE; ++i)
            for (int j = 0; j < BLOCK_SIZE; ++j)
                switch (color_) {
                case 0:
                    sum ^= pic.GetPixel(y_ + j, x_ + i).Red & 1;
                    break;

                case 1:
                    sum ^= pic.GetPixel(y_ + j, x_ + i).Green & 1;
                    break;

                case 2:
                    sum ^= pic.GetPixel(y_ + j, x_ + i).Blue & 1;
                }

        return sum;
}
```