Дерканосов Александр Сергеевич, 1маг, МКН

Лабораторная работа № 1

Контроль аутентичности содержимого цифровых изображений с использованием технологии цифровых водяных знаков и перцептивных хеш

Цель работы

Реализовать схему контроля аутентичности содержимого цифровых изображений на основе стеганографического встраивания информативной низкочастотной составляющей блоков изображения в само изображение.

Задание

Схема должна обеспечивать не только контроль аутентичности содержимого исходного изображения в целом, но также возможность локализации областей, подвергшихся искажениям низкочастотного характера (например, редактирование отдельных цифр/символов отсканированного текста, удаление информации о правообладателе в растровом представлении и т.п.). Смоделировать работу злоумышленника – реализовать различные типы негативных воздействий по отношению к маркированным контейнерам (фильтрацию, компрессию, дорисовку/удаление объектов и т.п.). Оценить вероятности ошибок первого и второго рода при проверке аутентичности содержимого блоков контейнера для различных типов маркируемых изображений (сканированные текстовые документы, фотографические изображения), различных типов внесенных негативных воздействий и различных значений порогов τ.

Результаты выполнения задания

Была реализована схема контроля аутентичности содержимого цифровых изображений на основе стеганографического встраивания информативной низкочастотной составляющей блоков изображения в само изображение.

Разработка проводилась в среде MATLAB версии R2020b. Для удобства работы были использованы стандартные средства и методы по работе с изображениями из библиотеки Image Processing Toolbox.

Рабочая программа представляет собой скрипт, разделенный на 2 секции. В первой секции производится стеганографическое встраивание, на вход передается название файла с изображением:

```
img = imread('csf.png'); % загрузка изображения из папки с кодом
```

Запустив первую секцию, на выходе получаем зашифрованное изображение, которое можно сохранить в эту же папку, воспользовавшись командой imwrite:

```
imwrite(newImage, 'shifredcsf.png');
```

Теперь это изображение можно изменить, имитируя действия злоумышленника, и подать его во вторую секцию, аналогично используя команду imread:

```
badImage = imread('badcsf.png');
```

Запуская отдельно вторую секцию, на выходе получаем маркированное изображение с указанием участков, где были внесены изменения (подсвечены красным цветом).

Скрипт можно запускать весь, тогда изображение из первой части попадет сразу во вторую секцию, при этом работу злоумышленника можно имитировать непосредственно в коде.

Все результаты и промежуточные шаги выводятся в виде изображений с помощью команды imshow. Всего их 4:

1) изначальное изображение,
2) зашифрованное,
3) изменённое (или неизменённое, если злоумышленник не перехватил нашу картинку),
4) маркированное с указанием измененных блоков.

Для всех значимых частей приведены комментарии в коде, здесь отмечу наиболее важные моменты:

1. Для расчёта перцептивного хеш использовался алгоритм pHash на основе дискретного косинусного преобразования. Для этого использовалась стандартная функция dct2, возвращающая матрицу коэффициентов ДКП.
2. Алгоритм шифрования хеша применять не стал.
3. Стеганографическое встраивание производилось по методу Куттера-Джордана-Боссена:

```
% стеганографическое встраивание по методу Куттера-Джордана-Боссена:
        B = blockImage(:,:,3); % выделение синего цветового канала
        % проходим по пикселям синего канала изображения:
        p = 1;      % номер встраиваемого бита сообщения
        for a = w + 1 : n - w
            for b = w + 1 : n - w
                % модифицируем пиксель в синем канале (прибавляем или
вычитаем яркость в этом месте)
                B(a, b) = B(a,b)+(2*s(p)-1)*q*grayBlockImage(a, b);
                p = p + 1;
            end
        end

        % копируем исходное изображение и вставляем измененный синий
канал обратно в контейнер:
        newBlockImage = blockImage;
        newBlockImage(:,:,3) = B;  % модифицированный блок изображения
        newImage{I, J} = newBlockImage; % заполняем массив блоков
нового изображения
```

Здесь w это ширина окрестности (число соседей слева, сверху, справа и снизу от модифицируемого пикселя), а q – энергия сигнала (чем меньше q, тем менее заметно изменение пикселей, но больше вероятность ошибки при извлечении).

Извлечение:

```
% извлечение информации
        s_ = s * 0; % сюда будем записывать извлеченные биты сообщения
        B = blockImage(:,:,3); % выделяем синий цветовой канал
```

```
        % проходим по пикселям синего канала изображения:
        p = 1;      % номер считываемого бита
        for a = w + 1 : n - w
            for b = w + 1 : n - w
                hs = B(a - w : a + w, b);    % пиксели сверху и снизу
от пикселя (a,b)
                vs = B(a, b - w : b + w);    % пиксели слева и справа
от пикселя (a,b)
                b_ = (sum(hs) + sum(vs) - 2 * B(a, b)) / (4 * w);    %
среднее
                % сравниваем значение пикселя с величиной среднего
соседей
                % если больше, то бит сообщения = 1, если меньше - 0
                s_(p) = B(a, b) > b_;
                p = p + 1;
            end
        end
```

4. Оценка близости записанного ранее хеша с рассчитанным сейчас при проверке аутентичности производится с помощью вычисления расстояния по Хэммингу:

```
D = pdist([s; s_], 'hamming'); % вычисление расстояния по Хэммингу
```

Если оно было больше определенного значения τ, которое устанавливается пользователем, то такой блок помечается красным цветом:

```
if(D>tau)
     block{I, J}(:, :, 1) = 1; % окрашивание блока в красный цвет,
если расстояние больше tau, т.е. блок скорее всего был изменен
злоумышленником
end
```

Параметр τ можно не только устанавливать вручную, но и автоматизировать его подбор:

```
tau = mean2(isAuthentic); % среднее значение расстояний D в массиве
for i=1:size(block, 1)
    for j=1:size(block, 2)
       if(isAuthentic(i, j)>3*tau)
           block{i, j}(:, :, 1) = 1; % если значение расстояния для
блока втрое выше среднего, этот блок окрашивается красным
       end
    end
end
```

Примеры работы программы

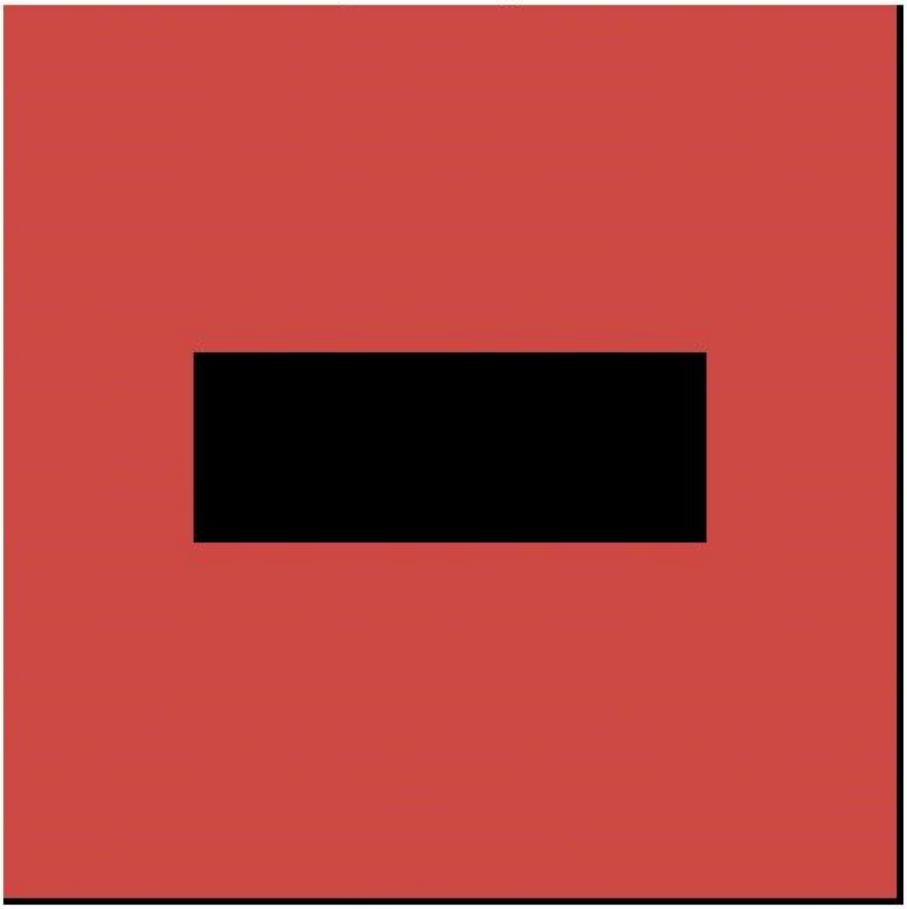
Пример №1

Удалим логотип ФКН с картинки:

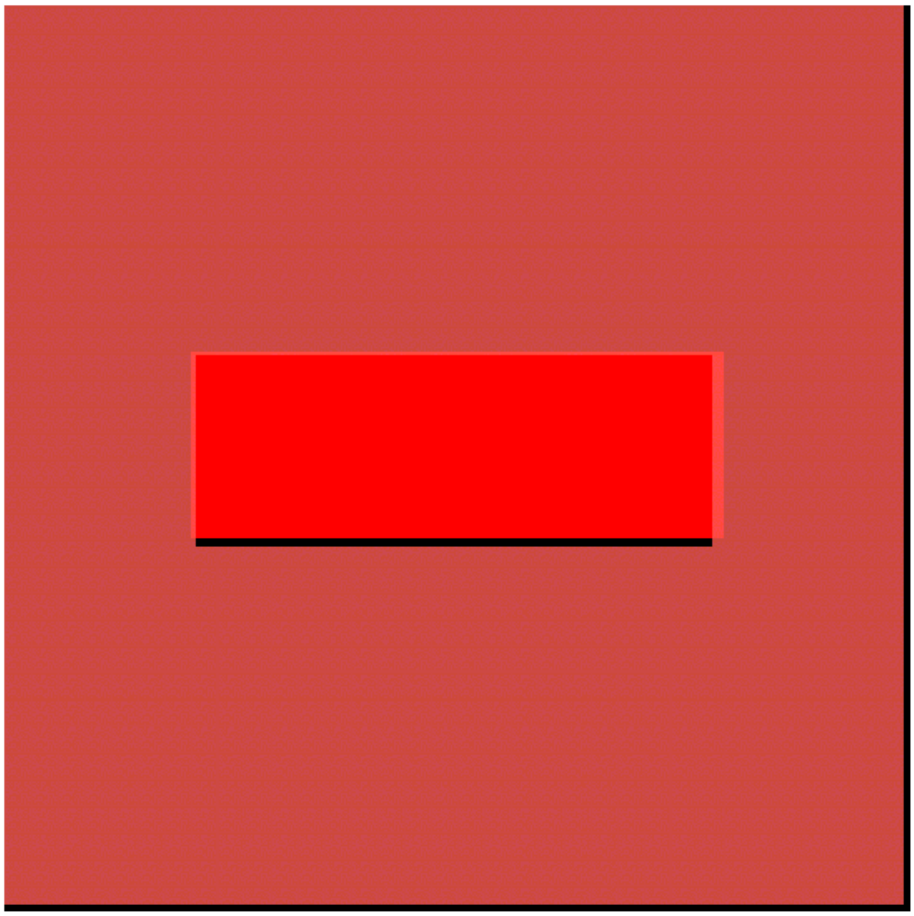
Исходное изображение

Зашифрованное изображение

Полученное изображение



Изображение с помеченными изменениями



Пример №2

Допустим, злоумышленник хочет подделать автомобильный номер на фотографии:

Как видим, программа обнаружила, где были внесены изменения.

Пример №3

Что будет, если шпион захочет внедрить вредоносный код?

Исходное изображение

```
USE [transferMarket]
GO
/****** Object:  Table [dbo].[Players]    Script Date: 18.10.2020 23:59:04 ******/
DROP TABLE [dbo].[Players]
GO
USE [master]
GO
/****** Object:  Database [transferMarket]    Script Date: 18.10.2020 23:59:04 ******/
DROP DATABASE [transferMarket]
GO
/****** Object:  Database [transferMarket]    Script Date: 18.10.2020 23:59:04 ******/
CREATE DATABASE [transferMarket]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'transferMarket', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\DAT
 LOG ON
( NAME = N'transferMarket_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL
 WITH CATALOG_COLLATION = DATABASE_DEFAULT
GO
ALTER DATABASE [transferMarket] SET COMPATIBILITY_LEVEL = 150
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [transferMarket].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [transferMarket] SET ANSI_NULL_DEFAULT OFF
GO
```

Полученное изображение

```
USE [transferMarket]
GO
/****** Object:  Table [dbo].[Players]    Script Date: 18.10.2020 23:59:04 ******/
DROP TABLE [dbo].[Players]
GO
USE [master]
GO
/****** Object:  Database [transferMarket]    Script Date: 18.10.2020 23:59:04 ******/
DROP DATABASE [transferMarket]
GO
                                                t]    Script Date: 18.10.2020 23:59:04 ******/
/* текст, встроенный злоумышленником */
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'transferMarket', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\DAT
 LOG ON
( NAME = N'transferMarket_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL'
 WITH CATALOG_COLLATION = DATABASE_DEFAULT
GO
ALTER DATABASE [transferMarket] SET COMPATIBILITY_LEVEL = 150
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [transferMarket].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [transferMarket] SET ANSI_NULL_DEFAULT OFF
GO
```

Изображение с помеченными изменениями

```
USE [transferMarket]
GO
/****** Object:  Table [dbo].[Players]    Script Date: 18.10.2020 23:59:04 ******/
DROP TABLE [dbo].[Players]
GO
USE [master]
GO
/****** Object:  Database [transferMarket]    Script Date: 18.10.2020 23:59:04 ******/
DROP DATABASE [transferMarket]
GO
/* текст, встроенный злоумышленником */t]    Script Date: 18.10.2020 23:59:04 ******/
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'transferMarket', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\DAT
 LOG ON
( NAME = N'transferMarket_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL'
 WITH CATALOG_COLLATION = DATABASE_DEFAULT
GO
ALTER DATABASE [transferMarket] SET COMPATIBILITY_LEVEL = 150
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [transferMarket].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [transferMarket] SET ANSI_NULL_DEFAULT OFF
GO
```

Выводы

Полученная схема обеспечивает не только контроль аутентичности содержимого исходного изображения в целом, но локализует совершенные изменения. Была смоделирована работа злоумышленника – вносились различные изменения (удаление/добавление/изменение объектов) в фотографии и скриншоты текста.

При правильном подборе параметра τ ошибки определения стремятся к нулю. Путем многочисленных опытов над различными изображениями было выяснено,

что оптимальным значением является $\tau = 0{,}3$. Для неизменённого изображения обычно расстояние Хэмминга не превышает 0.1 для каждого блока. Подвергшиеся изменению блоки показывают 0,4 и выше. Для автоматического подбора τ можно устанавливать его втрое бóльшим, чем среднее значение по массиву, но в этом случае возможны ошибки, например, когда дополненные нулями края изображения показываются как измененённые.

Также на качество аутентификации влияют параметры встраивания q и w. Было выяснено, что для энергии сигнала q оптимальным будет значение 0,2, т.к. при q >= 0,3 уже заметны синие встроенные пиксели, а при q <= 0,1 повышается ошибка извлечения. Для ширины окрестности w оптимальным является значение 1-2, т.к. при больших значениях видны незаполненные «рамки» вокруг каждого блока.