Выполнение декомпрессии изображения формата JPEG для стегоанализа

Трушин Н.А., Толмачев В.С.

Научный руководитель И.В. Горбунов, с.н.с. каф. КИБЭВС

г. Томск, ТУСУР

Проект ГПО КИБЭВС- 1519 – Моделирование системы защиты информации

Постановка задачи

Сегодня, когда цифровые способы передачи информации наиболее актуальны, злоумышленники могут воспользоваться стеганографическими методами для скрытой передачи сообщений. Одним из самых популярных стегоконтейнеров являются изображений в формате JPEG. Для выявления фактов передачи информации злоумышленниками существуют методы стегоанализа, но их реализация требует декомпрессии фалов JPEG формата.

Для проведения стеганографического анализа изображения, методиками блочной бинарной верстки и группировки коэффициентов ДКП необходимо произвести декомпрессию и получить битовую плоскость изображения (YCrCb, CMYK, RGB и т.д.). При этом, необходимо проанализировать различные хранящиеся мета данных и передать их для дальнейшего стегоанализа.

Описание структуры JPEG

Графический файл подделен на две основные части: заголовок и тело. Заголовок в свою очередь поделен на сегменты, каждый из которых предваряем специальным маркером. Тело же содержит сами кодовые слова, необходимые для декомпрессии. В свою очередь, заголовок JPEG-файла можно формально разделить на две части: это метаданные (расширение изображение, его размеры, прореживание и т.д.) и данные необходимые для декомпрессии.

Описание процесса декомпрессии JPEG

Предварительным этапом проведения декомпрессии является извлечение таблицы Хаффмана и построение кодового дерева, а также извлечение таблиц квантования. Для этого необходимо просканировать сегменты DHT (Data Huffman’s Table) и DQT (Data Quantization Table), с соответствующими маркерами [FF C4] и [FF DB]. Каждый из этих сегментов, после маркера, имеет двухбайтовый фрагмент определяющий его длину и одной байтовый идентификатор, следующий после неё. [2]

В смысловой части DHT содержится количество кодовых слов, необходимых для декодирования Хаффмана. При этом первые 16 байт тело сегмента отражают количество кодовых слов определённой длины, а остальные уже сами кодовые слова. Для более наглядный пример изображен на рисунке 1, где количество кодовых слов, имеющих длину 1 бит равно нулю, 2 бита – 1 слово, 3 бита – 5 слов и т.д. [1]

```
000000E0 | 00 11 08 04 38 07 80 03 01 22 00 02 11 01 03 11 | ....8....".....
000000F6 | 01 FF C4 00 1F 00 00 01 05 01 01 01 01 01 01 00 | ................
00000105 | 00 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09 | ................
00000110 | 0A 0B FF C4 00 B5 10 00 02 01 03 03 02 04 03 05 | ................
```

Рисунок 1 – Секция DHT

На основе полученной информации строим бинарное дерево с кодовыми словами. При этом стоит учитывать, что полученные битовые последовательности должны иметь

префиксный код. Для этого использовался следующим алгоритм построение дерева: в каком узле мы бы не находились, всегда пытаемся добавить значение в левую ветвь, если она занята, то в правую. Если и та занята – поднимаемся на уровень выше. Остановить нужно на уровне равном длине кодового слова.

После приступаем к чтению секции SOS и SOF, в которых хранятся указатели на идентификаторы составных компонент изображения. И переходим к составлению MCU блоков для каждой компоненты, на основе AC&DC коэффициентов. Следующих после секции SOS [1].

Для нахождения DC коэффициента читаем последовательность битов, синхронно вдвигаясь по дереву Хаффмана, по ветвям 0 или 1, в зависимость от прочитанного бита. После попадания в конечный узел, берем его значение. Если оно равно 0, то коэффициент равен 0, если он отличен от нуля, то его значение является длинный коэффициентов в битах, читаем следующие биты и получаем следующие значение, которое подлежит следующему преобразованию. Если первая цифра в двоичном представлении – 1, то записываем выражение в таблицу без преобразования, иначе преобразуем, и записываем в таблицу. Также стоит обратить внимание, что таблица в зигзагообразном виде.

Для нахождения AC коэффициента, продолжаем читать последовательность битов. Если значение узла равно 0, то заполняем оставшиеся значения матрицы нулями и далее следует следующая матрица. Иначе, берем старшую часть полученного выражения и добавляем столько нулей в матрицу, а младшая часть является длинной коэффициента в битах.[3]

Таким образом находим для каждой компоненты соответствующее количество MCU матриц, которое определяется форматом дискретизации. При этом стоит учитывать, что DC коэффициенты представлены их разностям и их нужно преобразовать:

На следующем шаге необходимо произвести квантование, перемножаем каждый элемент MCU блока с соответствующим элементом матрицы квантования, полученной в DHT. После чего производим дискретно-косинусное преобразование.

На данном этапе мы произвели все необходимо для статистического стегоанализа, однако, необходимо выполнить сканирование мета данных. Например, секции с комментарием, под маркером [FF FE], ведь именно туда можно без особых усилий встроить информацию, а также извлечь оттуда. Также стоит обращать внимание на маркеры, не предусмотренные стандартом кодирования JPEG, поскольку для значительной части ПО для просмотра изображений, они не представляют никакого интереса, в них также может скрывать стеговложение.

Заключение

В результате проделанной работы был изучена структура формата файлов JPEG, реализована декомпрессия файлов этого формата,

Реализация декомпрессии позволит произвести разработать и реализовать новые алгоритмы статического стегоанализа изображений. Результаты работы могут быть использованы для сканирования различных сегментов потенциально содержащие скрытое вложение и не только выявить факт вложения в изображение.

Список литературы

1. http://www.impulseadventure.com/photo/jpeg-decoder.html
2. https://en.wikipedia.org/wiki/JPEG
3. https://habrahabr.ru/post/102521/