А. С. Борщ

АНАЛИЗ ФОРМАТНЫХ ЧАСТЕЙ МУЛЬТИМЕДИЙНЫХ ФОРМАТОВ НА ВОЗМОЖНОСТЬ ПОТЕНЦИАЛЬНОГО СОКРЫТИЯ ИНФОРМАЦИИ

1. Анализ форматных частей MP3-файлов

MP3 является потоковым форматом, имеющим кадровую структуру. Все кадры (frame) построены по одинаковой структуре [1, 2]:

- *Header* (32),
- *CRC* (0, 16),
- *Side Information* (136, 256),
- *Main Data*,
- *Ancillary Data*.

В начале идет заголовок фрейма. За ним располагается контрольная сумма. После это так называемая *Side Information* — информация в потоке, необходимая для управления декодером.

Затем следуют основные данные. Фрейм завершается *Ancillary Data*.

Не все из этих частей обязательно присутствуют во фрейме. Так, наличие контрольной суммы будет определяться значением соответствующего бита в заголовке фрейма. Рассмотрим подробно первые три части фрейма.

Структура заголовка фрейма:

AAAAAAAA AAABBCCD EEEEFFGH IIJJKLMM

Описание элементов структуры приведено в таблице 1.

Таблица 1. Структура заголовка фрейма

Обозначение	Длина (биты)	Позиция (биты)	Описание
A	12	(31-20)	Синхропоследовательность
B	1	(19)	Биты описания алгоритма
C	2	(18,17)	Биты описания уровня кодирования
D	1	(16)	Бит защиты
E	4	(15,12)	Индекс битрейта

F	2	(11,10)	Используемая частота дискретизации
G	1	(9)	Флаг дополнения (*padding bit*)
H	1	(8)	*Private bit* (зарезервировано)
I	2	(7,6)	Вид сигнала
J	2	(5,4)	Расширение типа стереорежима (только для *Joint stereo*)
K	1	(3)	Авторские права
L	1	(2)	Флаг оригинала
M	2	(1,0)	*Emphasis*

Рассмотрим поля, потенциальное изменение которых не ведет к изменению звучания. Поля авторских прав, флага оригинала и private bit никаким образом не влияют на звук, однако изменение их значений будет являться сильным демаскирующим признаком, поскольку они должны оставаться константными на протяжении всего битового потока.

Наиболее сильным методом стеганографического сокрытия является вставка данных в контрольную сумму, которая находится сразу за заголовком фрейма, при условии выставки соответствующего флага (D). Если флаг равен 0, то избыточность была добавлена. Рассмотрим подробнее контрольную сумму (CRC).

Как было сказано выше, CRC вносит некоторую избыточность в MP3-файлы. Это дает один из вариантов сокрытия информации.

Устройство CRC:

CRC представляет собой линейный регистр сдвига [2]:

CRC-Check diagram

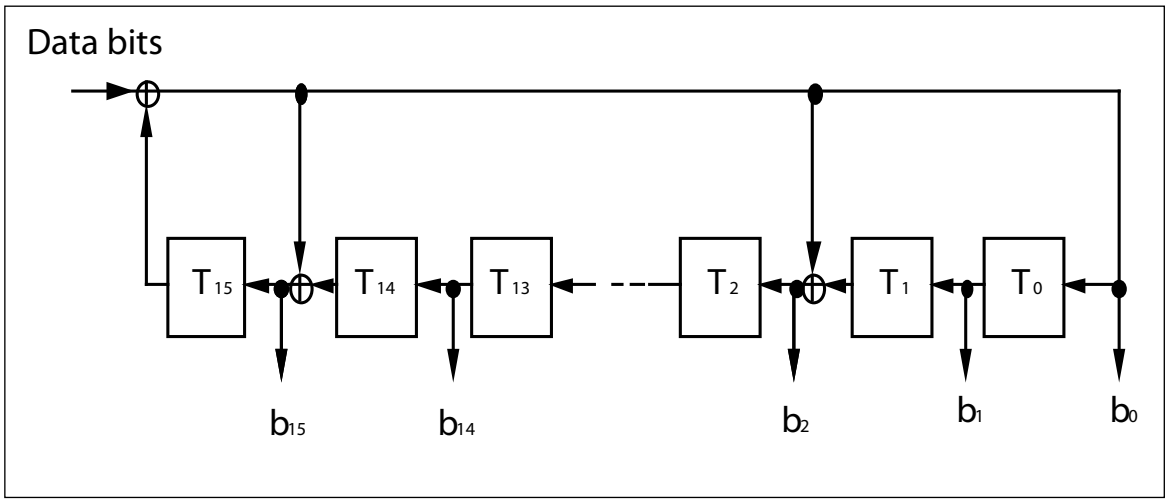


Рис. 1. Описание CRC-суммы

Начальное заполнение представляет собой 16 единиц, а данные, поступающие на вход, берутся из Header-а и Side information.

1) Из Header-а на вход регистра поступают его последние 2 байта (поступают побитно, начиная с максимального разряда).

2) Затем на вход подаются данные из Side information, в зависимости от параметров файла число байт из SI меняется от 9 до 32.

Таким образом, CRC может зависеть минимум от 11 и максимум от 34 байт. Однако алгоритм не меняется, что позволяет реализовать его в программном виде и делать предположения о местах, где можно спрятать данные.

Заметим, что на вход подается n-мерный вектор (n лежит от 88(11*8) до 272(34*8)), а на выходе имеется 16-мерный вектор. Основным стеганографическим «плюсом» CRC является линейная функция обратной связи, которая позволяет составить следующее соотношение, имеющее очень большое значение:

$C(Vn+e_i)= C(Vn) + C(e_i)$,

где $C(Vn)$ — выходной вектор CRC при входе равен Vn, а $C(e_i)$ соответственно выходной вектор CRC при входе равен e_i. e_i — входной вектор, у которого все элементы равны нулю, а i-й элемент равен 1. Таким образом, можно сказать, что если какое-то значение j выходного вектора равно 1 (на входном e_i), то значение $C(Vn+e_i)$ в j-й точке будет инвертироваться относительно $C(Vn)$, что позволяет утверждать: добавляя (если нужно) или не добавляя (если не нужно) e_i к Vn, получаем на j-м месте CRC нужный бит. Теперь несложно понять и сам алгоритм сокрытия в стеганоконтейнер, представленный в виде MP3-файла. Идея проста — на j-е место CRC ставим бит, совпадающий с соответствующим битом открытого текста, причем если изначально биты не совпадают, то следует изменить часть Side information, т. е. найти такое e_i, которое инвертирует $C(Vn+e_i)$ относительно $C(Vn)$. Как показали исследования, таких подходящих e_i может быть достаточно много и остается только выбрать более приемлемое. Заметим, что знак сложения в данном случае означает сложение по модулю 2.

Варианты вставки с использованием CRC: После анализа MP3-файлов было найдено несколько возможностей сокрытия информации в форматной части. Далее приводятся методы, позволяющие внедрять данные с разной степенью надежности. Проведен их анализ и указаны алгоритмы встраивания.

Из возможных вариантов данных фрейма, которые можно изменить, нужно выделить следующие части: поле Start_of_main_data, показывающее отрицательное смещение начала поля данных относительно Header-а данного фрейма, т. е. определяющее местоположение начального бита основных данных; поле copyright (авторские права), определяющее, защищен ли поток авторскими правами, и содержащееся в каждом фрейме поле original/home, определяющее копия это или оригинал. Как показала практика, изменение последнего бита поля Start_of_main_data не производит значительного изменения звука, а в большинстве кадров не производит вообще никакого изменения.

Звук не меняется абсолютно при изменении полей copyright и original/home, которые не участвуют в обработке данных, а служат как служебная вставка. Однако эти поля менять опасно с точки зрения того, что Header находится всегда на «виду», а изменение полей, которые должны на протяжении всего файла оставаться постоянными, может привести к подозрениям.

Алгоритм сокрытия представляет собой следующий процесс: сначала следует найти файл MP3-формата, содержащий CRC, сразу следует указать, что простой поиск может не дать положительного результата, так как файлы, содержащие CRC, — большая редкость. Поэтому предпочтительнее действовать следующим образом: найти файл формата WAV и перекодировать его в MP3 с добавлением контрольной суммы. Затем надо, изменяя, если необходимо, определенные параметры, от которых зависит CRC, получать в каждом кадре биты сообщения, если оно изначально не шифровано, и биты шифртекста в противном случае, которые будут располагаться на определенных местах CRC. Например, в случае с сокрытием при использовании поля Start_of_main_data можно использовать последний бит второго байта контрольной суммы, а в случае использования поля Original — последний бит первого байта CRC.

2. Анализ форматных частей JPEG-файлов

В данном разделе рассматривается проблема стеганографического сокрытия информации в JPEG-файлах. Наиболее часто сообщения вставляются в графические файлы [3], что определяется несложностью формата, например, изменение последних битов определенных байтов в BMP-файлах не меняет картинку визуально, меняется только статистика «шумов». Однако для файлов формата JPEG сокрытие информации представляет куда более сложную задачу, это обусловлено особенностями формата и алгоритмами кодирования. Вариантов внедрения информации в файлы формата JPEG множество, они основаны, как правило, на разборе картинки и вставлении чего-либо в места, мало отличительные от другого окружающего фона, соответственно, человеческий глаз не воспринимает небольшую разницу.

Стандарт JPG [7], определяет, что JPEG-файл сформирован в основном из частей, названных сегментами. Сегмент — это поток байт с общей длиной <=65535. Сегментное начало определяется маркером. Маркером являются 2 байта, первый из которых всегда равен FF, а второй является указателем на тип сегмента, например FFDA, FFD8 и другие. Если в течение обработки файла встречается байт FF, а за ним байт, не равный 0 и не имеющий значения маркера, то байт FF должен игнорироваться и пропускаться.

Всякий раз, когда встречается FF, следует проверить следующий байт на значение маркера. Однако могут быть ситуации, когда действительно необходимо записать байт FF как обычный байт. В таких случаях нужно сделать следующий байт равным 0, тогда дешифратор должен будет трактовать последовательность FF00 как обычный байт FF. Другое замечание: все маркеры являются выровненными по байту в файле JPG Что случается, если после кодирования Хаффмана битовая последовательность не смогла уложиться ровно в целое число байт. Тогда следует установить остальные биты до начала следующего байта на 1 и писать маркер в начале следующего байта.

Основные используемые маркеры JPG:

Формат файла:

Заголовок — 2 байта: FFD8 — SOI (Start of image). Эти 2 байта идентифицируют JPEG-файл; любое число сегментов; конец файла — FFD9 — EOI (End of image).

Формат сегмента:

Заголовок (4 байта): FF — идентифицирует сегмент, n — 1 байт — тип сегмента, Sh, sl — размер сегмента, включая эти 2 байта.

Типы основных использующихся сегментов:

Таблица 2. Основные использующиеся сегменты JPEG-файлов

Байт маркера	Описание
C0 — SOF0	Начало кадра
C4 — DHT	Определение таблицы Хаффмана
DN — RSTn, где N = [0..7]	Сегменты используются для синхронизации
D8 — SOI	Начало изображения
D9 — EOI	Конец изображения
DA — SOS	Начало основных данных
DB — DQT	Определение таблицы квантования
DD — DRI	Определение интервала перезапуска

Рассмотрим подробнее DRI (определение интервала перезапуска):

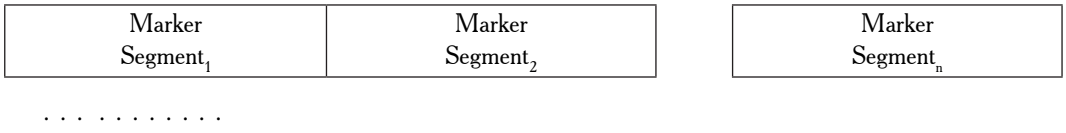
1. FFDD
2. Длина (старший байт, младший байт) должна быть равна 4.
3. Интервал перезапуска (старший байт, младший байт).
4. Первый маркер будет RST0, затем RST1 и т. д., после RST7 будет вновь RST0.

Этот сегмент используется далеко не всегда, однако именно его присутствие в файле дает интересную возможность скрыть информацию в последних битах, которые (из-за выравнивания по байту) должны быть выставлены все в 1. Использование этого сегмента многие специалисты считают нецелесообразным из-за того, что как раз и происходит прерывание нормального битового потока.

Таблица 3. Структура маркера RSTn

Маркер RSTn	Данные	Последний байт, где обычно аааа1111, а €0,1.

Свойства сегментов JPEG-файлов.



. . . . . .

Рис. 2. Разбиение JPEG-файла на сегменты

На рис. 2 представлен обычный JPEG-файл с разбиением на сегменты. Одно из свойств заключается в том, что для просмотрщика не будет абсолютно никакой разницы, в каком порядке мы запишем в файл следующие сегменты: таблицу квантования, таблицу Хаффмана, арифметическую таблицу (сейчас практически не используется), интервал перезапуска, комментарии, сегменты APPn (сейчас также уже практически не используются) [7].

Внутри многих сегментов существует множество зарезервированных полей, при попытке изменять что-то в этих полях результат был достаточно неожиданным: разные просмотрщики реагировали по-разному, т. е. либо показывали картинку без изменений, либо выдавали ошибку.

Внесение избыточности в JPEG-формат. Если в данных внутри сегмента стоит байт FF, а следующий за ним не равен ни нулю, ни маркеру сегмента, то FF игнорируется [8, 9]. Таким образом, мы можем сами вносить избыточность в JPEG-файл. Вот как это можно использовать: мы вставляем байт FF перед байтом, последний (можно необязательно последний) бит которого является очередным битом нашего сообщения. Преимущество этого метода состоит в относительной простоте, а также в возможности занесения в файлы JPEG достаточно большого сообщения. Недостатком этого метода я считаю то, что наличие в файле многих байтов FF, не имеющих впоследствии значения маркера, может привлечь внимание потенциального противника.

Использование маркера RSTn при сокрытии данных. Метод состоит в использовании избыточности при появлении маркеров RSTn в JPEG-файле. Так как все сегменты являются выровненными по байту внутри файла, то всегда возникают ситуации, когда приходится забивать последние биты последнего байта предыдущего сегмента единицами, что, естественно, вносит определенную избыточность. Причем надо заметить, что обычно последние 3–4 бита как раз заполняются единицами, т. е. делают небольшой битовый запас, идеально подходящий для стеганографического внедрения скрытого сообщения. Заметим, что интервал перезапуска, а следовательно, маркеры RSTn почти всегда возникают при получении картинки JPEG путем цифровой фотографии и практически никогда не возникают при перекодировании BMP в JPEG.

Достоинством метода является то, что противнику трудно определить факт наличия информации. Минусом является невозможность внедрения большого сообщения, например, в файл размером 170 Кб можно занести примерно 0,5 Кб данных при использовании 3 бит.

Использование перестановки сегментов. Третий метод состоит в том, что мы можем в каком угодно порядке расставлять сегменты файла JPEG [7], показанные выше. Это дает нам возможность передачи информации с помощью манипулирования порядком их расположения. Например, мы можем передавать сообщение через страничку сети Интернет следующим образом: отправитель скрытого сообщения выкладывает на данный сайт картинки в определенном порядке, а получатель, смотря на расположение сегментов, может прочитать сообщение. Этот метод наиболее тяжело определить, так как структура файла не меняется, все значения остаются постоянными, на изображение данное изменение не влияет абсолютно. Минусом этого метода является невозможность быстрой передачи большого сообщения, так как надо изменить очень много картинок и передать их получателю. Этот метод может подходить для передачи ключей по открытым каналам.

Заключение

Таким образом, в данной работе были выделены поля форматных частей MP3- и JPEG-файлов, подходящих для стеганографического сокрытия информации. Как можно заметить, таких полей довольно много и проблема написания фильтров для выявления вставки данных довольно сложна. При этом указанные форматы имеют широкое распространение, что еще более затрудняет задачи стегоаналитиков.

СПИСОК ЛИТЕРАТУРЫ:

1. Dipl.–Ing, Ralph Sperschneider. MPEG–Layer3 Bitstream Syntax and Decoding 1999.
2. ISO/IEC International Standard IS 11172–3. Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/s – Part 3: Audio. 1993.
3. National Institute of Standards and Technology (NIST), FIPS Publication 180 – 1: Secure Hash Standard (SHS), 1995.
4. MP3: The Definitive Guide by Scot Hacker. 1st Edition. March 2000.
5. *Rangachar R.* Analysis Improvement of the MPEG – 1 AUDIO LAYER III. 1998.
6. *Бухаров Д.* Основы алгоритма сжатия JPEG и других алгоритмов. 2000.
7. ISO/IEC 10918 – 1 : 1993(E).
8. *Fromme O.* Техническое описание JPEG файлового формата. 1997.
9. *Wallace G. K.* The JPEG Still Picture Compression Standard.
10. *Hamilton E.* JPEG File Interchange Format Version 1.02.