МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

**НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №4

по курсу «Защита информации»

**Выполнил:** студент 4-го курса гр. 17208

Гафиятуллин А.Р

Новосибирск, 2020

Оглавление

[1. ЦЕЛИ РАБОТЫ 3](#_Toc58783816)

[2. ХОД РАБОТЫ 3](#_Toc58783817)

[2.1 LSB внедрение и извлечение в WAV-аудиофайл 3](#_Toc58783818)

[2.2 Сторонние средства для стеганографического внедрения 15](#_Toc58783819)

[3. ВЫВОДЫ 16](#_Toc58783820)

[4. СПИСОК ЛИТЕРАТУРЫ 16](#_Toc58783821)

[5. ПРИЛОЖЕНИЕ 17](#_Toc58783822)

# ЦЕЛИ РАБОТЫ

1. Написать программу, реализующую LSB внедрение и извлечение в WAV-аудиофайл;
2. Воспользоваться сторонними средствами для стеганографического внедрения.

# ХОД РАБОТЫ

## LSB внедрение и извлечение в WAV-аудиофайл

WAV-аудиофайл имеет определенную структуру, которая описана Си-структурами, приведенными на Рисунке 1. В начале файле расположен глобальный заголовок(**wave\_header**), который определяет формат файла(строка RIFF в поле id), его тип(строка WAVE в поле file\_type), его размерв байтах, без учета первых 8 байт, а также другую необходимую для воспроизведения информацию. После этого заголовка идут все остальные сегменты с заголовками, тип которых обозначен в поле **chunk\_marker**, а размер сегмента в поле **chunk\_length**.

Нами будет использоваться сегмент с заголовком, имеющим тип **data**. Данный сегмент содержит информацию об амплитудах волны аудиосигнала в каждый отдельно взятый момент времени и для каждого канала. Эти моменты времени и кол-во каналов зависят от параметров, указанных в глобальном заголовке, но нам они не важны, кроме параметра **bits\_per\_sample**, который несет в себе информацию о кол-ве битов, выделенных на представление одного значения амплитуды(8 бит, 16 бит, 32 бита и т. д.).

Идея внедрения состоит в том, что незначительное изменение амплитуды волны не должно сильно искажать аудиосигнал. То есть, перезаписывание младших битов каждой амплитуды незначительно изменит качество звучания. Чем больше битов перезаписывается, тем заметнее будет ухудшение звучания. Причем, реализация такова, что информация внедряется только в младший байт каждой амплитуды, то есть, чем большим количеством байтов представляется одна амплитуда, тем меньше максимальный процент заполнения аудиофайла внедряемой информацией. Это сделано ради простоты реализации с учетом того, что аудиофайлы сами по себе в среднем довольно большие по размеру в байтах.

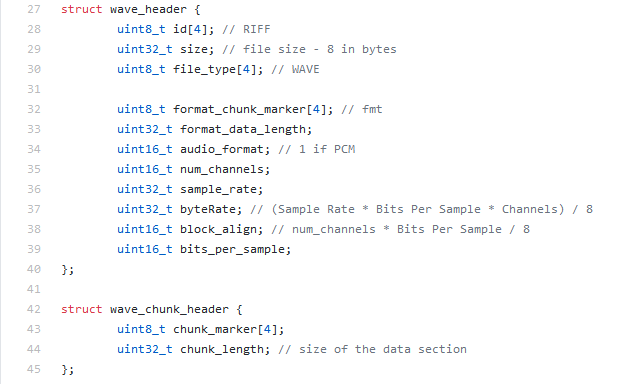


Рисунок 1. – Структура WAV-аудиофайла

Алгоритм внедрения реализован в функции **int hide\_src\_in\_dst(char \*src\_file\_name, char \*dst\_file\_name)**:

1. Происходит открытие файлов с именем **src\_file\_name**(информация, которую мы хотим внедрить), **dst\_file\_name**(целевой аудиофайл), причем формируется файл с именем **dst\_file\_name.out**(аудиофайл с уже скрытой информацией). Так же происходит получение длины файла src\_file\_name, которая потом будет записана в dst\_file\_name.out. Код этого этапа приведен на Рисунке 2;
2. Происходит считывание глобального заголовка из файла dst\_file\_name, его проверка на то, что это действительно заголовок WAV-файла и запись этого заголовка в файл dst\_file\_name.out. Функция **read\_wave\_header** считывает и сразу проверят заголовок. Функция **safe\_write** производит гарантированную запись указанного кол-ва байтов. Код этого этапа приведен на Рисунке 3;
3. Происходит поиск сегмента с заголовком типа data в файле dst\_file\_name, все остальные сегменты пропускаются и сразу записываются в файл dst\_file\_name.out. Функция **read\_wave\_chunk\_header** считывает заголовок в переданную структуру. Функция **safe\_read** производит гарантированное считывание в буфер указанного кол-ва байтов. Функция **is\_data\_wave\_chunk\_header** проверяет, что считанный заголовок является заголовком типа data. Код этого этапа приведен на Рисунке 4;
4. На основе размера внедряемого файла(src\_file\_name), размера сегмента файла с амплитудами(dst\_file\_name) и количества байтов на одну амплитуду вычисляется количество битов внедряемого файла, которые будут записаны в младшие биты каждой амплитуды. Это число округляется до значения, которое делит 8, чтобы упростить распределение битов внедряемого файла по битам амплитуд. Код этого этапа приведен на Рисунке 5;
5. Происходит запись размера внедряемого файла в первые значения амплитуды в сегменте с заголовком типа data и вывод модифицированной информации об амплитудах в выходной файл dst\_file\_name.out. Так как количество битов на амплитуду вычисляется на основе размера внедряемого файла, а при извлечении размера внедряемого файла этот размер еще не будет известен, то распределение идет по 1 биту на каждую амплитуду. Распределение битов внедряемого файла по битам амплитуд происходит с помощью функции **assign\_bytes\_to\_dst\_data**. Эта функция принимает буфер данных аудиофайла(амплитуды), буфер данных внедряемого файла, количество внедряемых байтов, количество битов внедряемого байта на амплитуду и количество байтов на амплитуду. С помощью битовых операций выделения нужных битов, сдвига и инвертирования биты внедряемого байта распределяются по младшим битам каждой амплитуды. Код этой функции приведен на Рисунке 6, а код всего этапа на Рисунке 7;
6. Происходит считывание очередного куска внедряемого файла и очередного куска амплитуд, внедрения с помощью assign\_bytes\_to\_dst\_dataи вывода результата в файл dst\_file\_name.out.Этот цикл длится, пока весь внедряемый файл не будет распределен по битам амплитуд. Код этого этапа приведен на Рисунке 8;
7. В конце происходит перекопированние оставшейся информации в файле dst\_file\_name в файл dst\_file\_name.out и выход из функции с очисткой ресурсов. Код этого этапа приведен на Рисунке 9.

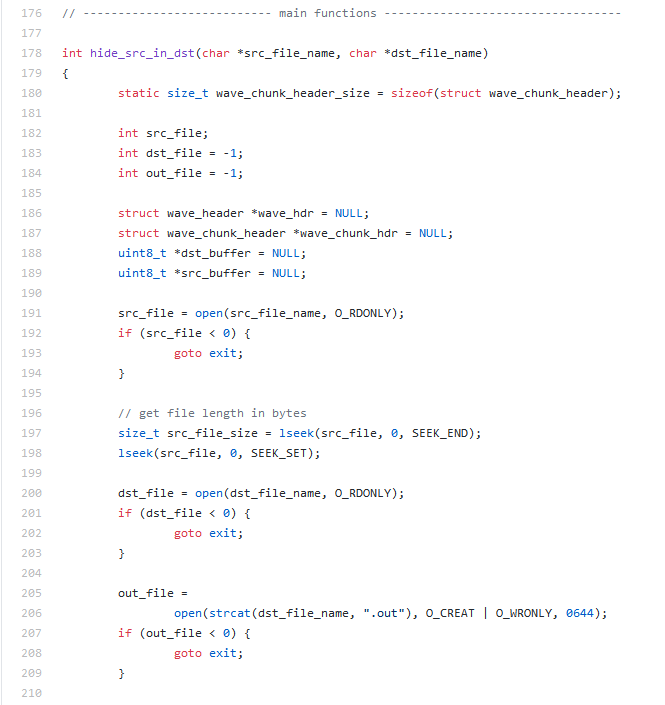


Рисунок 2. – Открытие и создание файлов

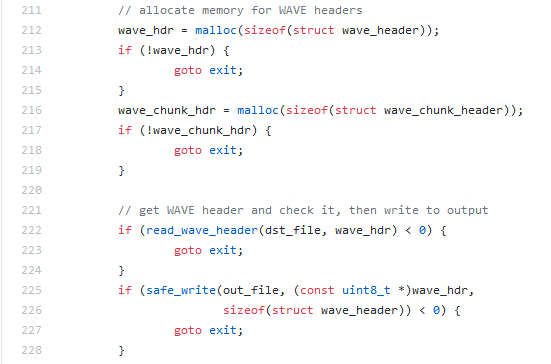


Рисунок 3. – Считывание, проверка и запись глобального заголовка



Рисунок 4. – Поиск заголовка с типом data

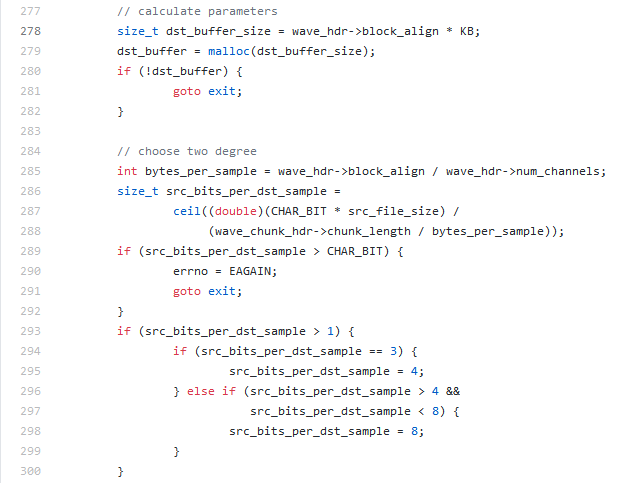


Рисунок 5. – Выбор количества битов внедряемого файла на амплитуду

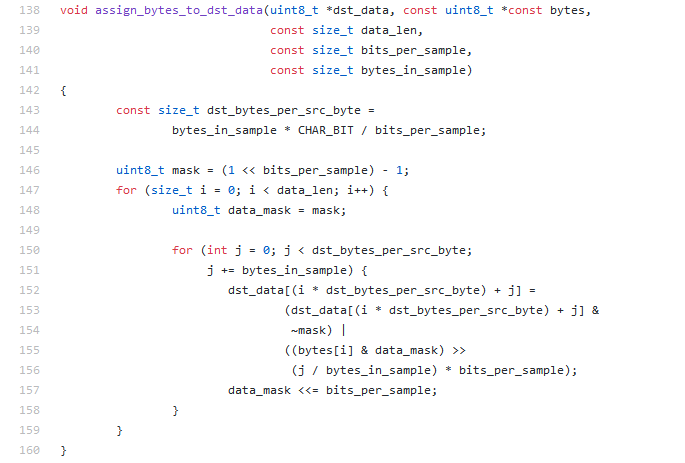


Рисунок 6. – Распределение битов внедряемых байтов по младшим битам амплитуд

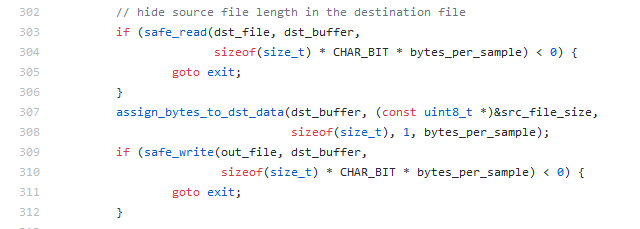


Рисунок 7. – Запись в выходной файл информации о размере внедряемого файла



Рисунок 8. – Распределение битов по амплитудам для всего внедряемого файла и вывод результата в выходной файл

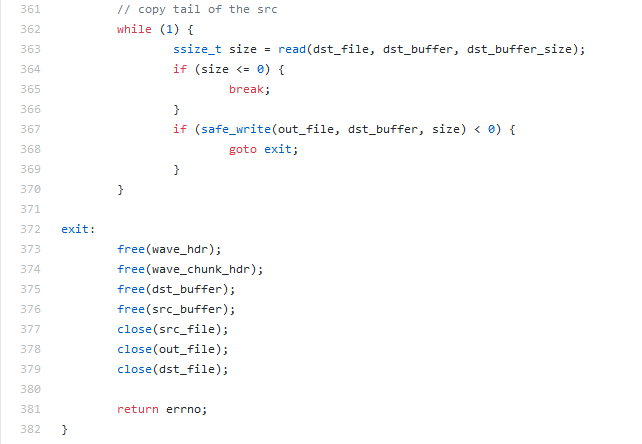


Рисунок 9. – Копирование остатка файла dst\_file\_name в файл dst\_file\_name.out

Алгоритм извлечения информации из аудиофайла реализован в **int get\_src\_from\_dst(char \*dst\_file\_name)**:

1. Открытие файлов, анализ глобального заголовка и поиск сегмента с заголовком типа data происходят аналогично функции внедрения информации, за исключением того, что считанная информация никуда не перезаписывается, а просто пропускается;
2. Из первых значений амплитуд сегмента с заголовком типа data извлекается информация о размере внедренного файла в байтах. Как уже было обозначено ранее, информация о размере файла распределена по 1 биту на каждый младший бит первых значений амплитуд. Значение их этих битов собираются с помощью функции **gather\_bits\_to\_bytes**. Эта функция принимает все те же аргументы, что и ее зеркальная функция assign\_bytes\_to\_dst\_data и с помощью обратных операций собирает байт обратно. Код этой функции приведен на Рисунке 10;
3. Вычисление количества битов внедряемых байтов на каждое значение амплитуды вычисляется так же, как и в функции внедрение информации;
4. Извлечение всей остальной информации из сегмента с заголовком типа data происходит практически так же, как и ее внедрение, за исключением того, что вместо assign\_bytes\_to\_dst\_data используется gather\_bits\_to\_bytes, а критерием остановки является извлечение всего файла по известному размеру;
5. Оставшаяся информация в файле никак не обрабатывается;
6. Освобождение ресурсов происходит аналогично функции внедрения информации.

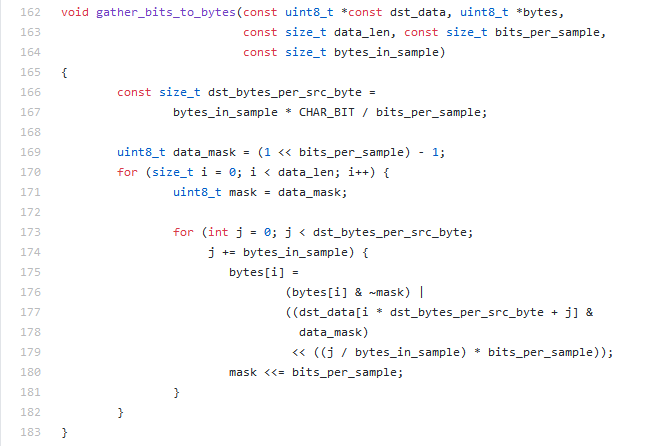


Рисунок 10. – Сборка байта внедренной информации из младших битов амплитуд

## Сторонние средства для стеганографического внедрения

Воспользуемся свободно распространяемым средством для стеганографического внедрения, которое называется **steghide**. Выбор пал на это ПО, так как оно обладает возможностью внедрения в WAV-аудиофайлы:

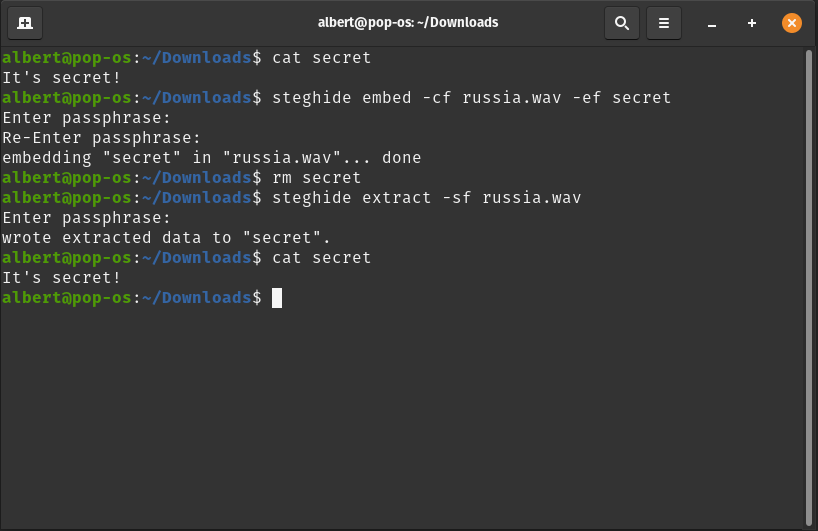


Рисунок 11. – Использование **steghide** для внедрения информации в WAV-аудиофайл

# ВЫВОДЫ

1. Написали программу, реализующую LSB внедрение и извлечение в WAV-аудиофайл;
2. Воспользовались сторонними средствами для стеганографического внедрения.

# СПИСОК ЛИТЕРАТУРЫ

1. Формат WAVE файлов. — URL: http://alexei-s1.narod.ru/WAVE.htm (дата обращения: 12.13.2020).
2. RIFF.— URL: https://ru.wikipedia.org/wiki/RIFF (дата обращения: 12.13.2020).
3. Steghide - a steganography program.— URL: http://manpages.ubuntu.com/manpages/focal/man1/steghide.1.html (дата обращения: 12.13.2020).

# ПРИЛОЖЕНИЕ

Исходный код и примеры использования(в README.md): <https://github.com/gafiyatullin-a/NSU-FIT/tree/master/4th-year/informaition-security/laboratory-work-4>