ВВЕДЕНИЕ

В переводе с греческого слово "стеганография" буквально означает "тайнопись". Она может осуществляться разными способами, которые объединены тем, что скрываемое сообщение встраивается в какой-либо объект, не привлекающий внимания и который открыто можно отправить адресату. То есть при стеганографии наличие скрытого сообщения остается незаметным для постороннего, хотя при использовании криптографии наличие шифрованного сообщения есть то, что противник заметит в первую очередь [1].

В компьютерной стеганографии в настоящее время используется огромное количество различных стегоконтейнеров для встраивания информации и существует много способов встраивания информации.

В данной работе в качестве контейнера используются аудио файлы формата mp3 и реализовано несколько методов встраивания информации.

При изучении подобных программ, было выявлено, что в большинстве из них был использован алгоритм предварительного шифрования данных DES, в них было осуществлено преобразование из сжатого формата mp3 в более подходящий, которым считается wav, чтобы внедрять информацию. В качестве алгоритма внедрения был использован алгоритм замены наименее значимого бита.

Первым этапом данной работы было исследование методов внедрения данных в mp3 файл напрямую без его преобразования, рассмотрены и реализованы программно 3 способа внедрения: в заголовок mp3 файла, в тело данных и запись в конец после всех данных, составляющих mp3 файл. Данные перед внедрением шифровались алгоритмом DES и затем встраивались в аудиофайл. Результатом данного этапа является программа, выполняющая все 3 способа следующим образом:

- в заголовок информация внедрялась в обычно пустое поле "Комментарий" версии заголовка ID3v1, но при этом проигрыватель аудио, показывающий информацию из заголовка, мог прочитать и показать зашифрованное сообщение, также минусом являлся слишком маленький максимально допустимый размер встраиваемых данных, второй вариант внедрения в заголовок - использование данных о версии ID3v2, в которой размер скрываемого сообщения больше, чем в первой версии и появилась возможность скрывать данные в пустом поле, не читаемом проигрывателями;

- при внедрении скрываемого сообщения в тело данных mp3 файла, происходит изменение структуры аудиофайла таким образом, что проигрыватель либо не может прочитать и воспроизвести весь файл (это случается чаще), либо его воспроизведении появляются очень заметные посторонние шумы;

- если использовать третий вариант и дописывать зашифрованное сообщение в конец файла после флага окончания тела данных, это никак не повлияет на воспроизведение, но и секретную информацию в этом случае будет проще найти врагу.

Для того, чтобы тщательнее спрятать необходимую информацию в mp3 файле, был исследован и реализован следующий этап, который заключается в том, что перед внедрением данных в имеющуюся аудиозапись формата mp3, преобразовывать ее в формат wav. Это позволит размещать секретное сообщение непосредственно в поле данных, используя алгоритм замены наименее значимых битов.

Основным отличием описываемого программного средства является то, что в нем есть возможность не только оценить на слух разницу между аудиозаписями до внедрения и после, которую сложно заметить, но и увидеть осциллограммы обоих аудиофайлов, наложенные друг на друга для более наглядного различия.

В работе рассматриваются основные понятия и определения, необходимые для понимания предметной области, осуществляется обзор существующих приложений, использующих стеганографические методы для внедрения информации в mp3 файлы.

Подробно описываются выбранные стеганографические методы и способы внедрения информации, реализованные в данной работе, конструируется общая схема алгоритма работы приложения, алгоритмы шифрования внедряемой информации и дешифрования, а затем ее сокрытия и извлечения, также описаны алгоритмы преобразования аудиофайлов из формата mp3 в wav.

Описана программная часть приложения, включающая в себя основные модули программы, их функции, входные и выходные параметры и связь между этими модулями. Также приводятся примеры тестирования ПО на стеганографическое внедрение информации и затем стегоанализ.

1 НЕОБХОДИМЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ ДЛЯ ВСТРАИВАНИЯ ДАННЫХ В MP3

В первой главе описываются основные понятия, определения, структуры и методы, упоминающиеся в данной работе.

* 1. Основные понятия стеганографии

Стеганография – это скрытая передача информации по каналам связи, которые называются стегоканалами. Данные сообщения внедряются в какой-либо исходный объект, называемый стегоконтейнером, который далее передается совершенно открыто, и в результате этого формируется скрытый канал передачи сообщения (стегоканал) [1].

Современные информационные системы могут использовать в качестве стегоканалов мультимедиа данные (в данной работе - звук) из-за большой избыточности. Для того, чтобы передать скрытые данные, часть той самой избыточной информации заменяется на биты передаваемого сообщения.

Особенность цифровой стеганографии заключается в том, что надо выбрать файл, который будет способен скрыть передаваемое сообщение. Файлы изображений, звука или видео хорошо подходят по нескольким критериям:

* указанные типы файлов уже сжаты алгоритмом. Например, .jpeg, .mp3, .mp4, и .wav форматы - всё это примеры алгоритмов сжатия;
* эти файлы достаточно велики, что облегчает задачу нахождения участков, в которых можно скрыть зашифрованный текст;
* эти файлы отлично скрывают. То есть, можно прятать текстовое сообщение, которое должно быть скрыто в изображении или аудио клипе и человек, не знающий о возможности передачи информации таким образом, не обратит внимания, так как факт сокрытия незаметен.

Если стеганографическая утилита справляется хорошо со своей работой, обычный пользователь не обратит внимания на какую-дибо малейшую разницу в качестве картинки или аудио, даже если некоторые биты были изменены для того, чтобы скрыть сообщение.

Эффективность стеганографического метода определяется максимальным объемом встраиваемой информации, сложностью алгоритма встраивания и извлечения сообщений, а также устойчивостью к стегоанализу. Обобщенная схема стегосистемы (этап встраивания) представлена на рисунке 1. Извлечение информации производится в обратном порядке [2].



Рисунок 1 - Схема структуры стеганосистемы

Встраивание состоит из двух основных этапов:

* формирование рабочей области контейнера и формирование битового блока;
* процесс встраивания передаваемого сообщения в контейнер (формирование стегоконтейнера) [2].
  1. Цифровой звук

Звук состоит из колебаний, которые при оцифровке приобретают ступенчатый вид. Этот вид обусловлен тем, что компьютер может воспроизводить в любой короткий промежуток времени звук определенной амплитуды (громкости). Продолжительность этого промежутка определяет частота дискретизации. Например, файл с частотой дискретизации 44.1 kHz, это значит, что тот короткий промежуток времени равен 1/44100 секунды (следует из размерности величины Гц = 1/с). Современные звуковые карты поддерживают частоту дискретизации до 192 kHz [15].

От амплитуды сигнала зависит точность звука. Амплитуда выражается числом, занимаемым в памяти (файле) 8, 16, 24, 32 бит (теоретически можно и больше). Как известно, 8 бит = 1 байту, следовательно, какая-то одна амплитуда в какой-то короткий промежуток времени в памяти (файле) может занимать 1, 2, 3, 4 байта соответственно. Таким образом, чем больше число занимает места в памяти (файле), тем больше диапазон значений для этого числа, а значит и для амплитуды.

* 1 байт – 0..255
* 2 байта – 0..65 535
* 3 байта – 0..16 777 216
* 4 байта – 0..4 294 967 296

В моно варианте значения амплитуды расположены последовательно. В стерео же, например, сначала идет значение амплитуды для левого канала, затем для правого, затем снова для левого и так далее [15].

Совокупность амплитуды и короткого промежутка времени носит название сэмпл.

* 1. Описание файлов mp3 формата

Для того, чтобы иметь возможность скрыть сообщение в используемом стегоконтейнере, в данной работе - это mp3 файл, нужно знать, из чего он состоит, чтобы правильно его использовать.

MP3(англ. MPEG-1/2/2.5 Layer 3; но не MPEG-3) — это кодек третьего уровня, лицензируемый формат файла для хранения аудиоинформации.

MP3 является одним из самых распространённых и популярных форматов цифрового кодирования звуковой информации с потерями [1].

При создании MP3 используется алгоритм сжатия с потерями, который существенно уменьшает размер данных, которые необходимы для воспроизведения аудиозаписи и обеспечения качества звука при воспроизведении очень похожего на оригинальный. MP3-файлы могут создаваться с разным битрейтом (высоким или низким), который влияет на качество и размер получившегося в результате сжатия файла. Если файл создан со средним битрейтом равным примерно 128 кбит/с, то размер файла на выходе будет равен 1/11 от оригинального файла, так как битрейт несжатого аудио формата CD-Audio составляет 1411,2 кбит/с.

Человеческое ухо воспринимает только небольшую область спектра и терпимо относится к небольшим искажениям звука. Поэтому отбрасывая несущественную часть спектра и изменяя при компрессии данные так, чтобы они лучше сжимались и при этом были приемлемо похожи на оригинал можно добиться значительного увеличения коэффициента сжатия данных[3].

* 1. Структура mp3 файла

В работе реализованы три метода внедрения информации: в заголовок, в данные и в конец аудиофайла. Чтобы правильно разместить информацию, не стерев нужные данные, изменение которых может повлиять на восприятие ухом человека музыкальной композиции, что выдаст факт сокрытия информации, необходимо различать полезные биты от тех, которые можно изменить. Для этого ниже описывается его структура, начиная с заголовка, описана разница в их версиях.

MP3-файл состоит из нескольких фрагментов (фреймов) MP3, которые, в свою очередь, состоят из заголовка и блока данных. Фрагменты не являются независимыми элементами, и поэтому не могут быть извлечены произвольно. Блок данных MP3-файла содержит сжатую аудио информацию в виде частот и амплитуд. На приведённой диаграмме 1 приложения Б показано, что заголовок MP3 состоит из маркера, который служит для нахождения верного MP3-фрагмента. За ним следует бит, показывающий, что используется стандарт MPEG и два бита, показывающие использование layer 3; другими словами, это определяет MPEG-1 Audio Layer 3 или MP3. Последующие значения могут варьироваться в зависимости от типа MP3-файла. Стандарт ISO/IEC11172-3определяет диапазон значений для каждой секции заголовка, вместе с общей его спецификацией. Большинство MP3 файлов в настоящий момент содержатID3-метаданные [1].

Теги— метки в границах MP3-файла (в начале и/или в конце). В них может быть записана информация об авторстве, альбоме, годе выпуска и прочая информация о треке. В более поздних версиях тегов возможно хранение обложек альбомов и текстов песен.

Существуют различные версии тегов [5]:

* ID3v1– имеет фиксированный размер в 128 байт. Там можно хранить: название трека, исполнитель, альбом, год, комментарий, номер трека и жанр;

Структура заголовка ID3v1 показана в таблице 1.1.

Таблица 1.1 - Структура заголовка ID3v1

|  |  |  |
| --- | --- | --- |
| Поле | Длина, байт | Описание |
| Header | 3 | Маркер. Всегда содержит 'TAG' |
| Title | 30 | Название трека |
| Artist | 30 | Исполнитель |
| Album | 30 | Название альбома |
| Year | 4 | Год |
| Comment | 28 или 30 | Комментарий |
| Zero-byte | 1 | Если есть номер трека, установлен нулевой байт |
| Track | 1 | Номер трека |
| Genre | 1 | Индекс жанра или 255 |

Внедрение данных без изменений нужной информации возможно провести в поле комментариев. Так как это поле заголовка имеет фиксированную длину, мы сможем поместить всего 28 байт информации при выборе режима внедрения в заголовок.

* со временем, появилась и успешно используется вторая версия данных ID3v2.В отличии от первой версии, теги v2 имеют переменную длину и размещаются в начале файла, что позволяет поддерживать потоковое воспроизведение. Данные ID3v2 состоят из заголовка и последующих фреймов ID3v2 и занимают 1024 байта [8].

Разбиение по байтам и описание информации содержащейся в каждом из них показано в таблице 1.2.

Таблица 1.2 - Структура заголовка ID3v2

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Байты | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Пример, hex | 49 | 44 | 33 | 03 | 00 | 00 | 00 | 00 | 07 | 76 |
| Описание | Маркер 'ID3' | | | Версия | Суб-версия | Флаги | Длина ID3 в байтах | | | |

Весь mp3-файл состоит из фреймов, которые можно извлекать только последовательно. Фрейм содержит в себе заголовок и аудио данные [5].

Например, файл имеет размер 350670 байт. Есть ID3v1 теги (128 байт) и ID3v2 теги (1024 байта). Битрейт = 96. Следовательно, размер аудиоданных равен 350670 – 128 – 1024 = 349518 байт.

* 1. Механизм сжатия

Принцип сжатия основывается на снижении точности некоторых частей звукового потока, и это оказывается практически неразличимо для слуха большинства людей. В этом случае на первом этапе строится последовательность недлинных промежутков времени в виде диаграммы, затем удаляется часть информации, которую не отличить человеческим ухом, а оставшаяся информация сохраняется в сжатом виде [1].

Психоакустическая модель — использование феномена восприятия человеком звука, для сжатия информации с потерями при хранении звуковой информации (например, в файлах mp3) [2].

MP3, согласно психоакустической модели [2], использует отсечения спектров. Сигнал разбивается на отрезки, равные по продолжительности. После обработки каждый участок упаковывается в свой фрейм. Последовательность воспроизведения определяется порядком расположения фреймов. Каждый фрейм может кодироваться с разными параметрами.

Для расчетов используются предыдущий фрейм и следующий, так как для спектрального разложения необходим непрерывный входной сигнал. В таком сигнале есть гармоники, которые можно отсечь и при этом не каждый человек сможет ощутить разницу (наличие таких гармоник). Можно отсекать гармоники, которые лежат рядом с более интенсивными, гармоники с меньшей амплитудой, также можно заменять два или более близлежащих пика одним. Все это происходит исходя из требований к потоку на выходе.

Уменьшение потока происходит за счет спектрального разрежения, так как гармоники высоких частот удаляются только выборочно, затем, перед упаковкой во фреймы, используются математические методы сжатия.

Интервал значений битрейта может варьироваться от 8 до 320 кбит/с.

Кодирование начинается с того, что исходный сигнал с помощью фильтров разделяется на несколько, представляющих отдельные частотные диапазоны, сумма которых эквивалентна исходному сигналу [3].

Для каждого диапазона определяется величина маскирующего эффекта, создаваемого сигналом соседних диапазонов и сигналом предыдущего фрейма. Если она превышает мощность сигнала интересующего диапазона или мощность сигнала в нем оказывается ниже определенного опытным путем порога слышимости, то для данного фрейма данный диапазон сигнала не кодируется. При 128 кбит/с происходит срез частот выше 16,5 кГц, при 256 кбит/с такого не наблюдается.

Для оставшихся данных каждого диапазона определяется, сколькими битами на сэмпл мы можем пожертвовать, чтобы потери от дополнительного квантования были ниже величины маскирующего эффекта.

После завершения работы психоакустической модели формируется итоговый поток, который дополнительно кодируется по Хаффману, на этом кодирование завершается [2].

На практике эта схема сложнее и описать ее можно следующим образом.

Во-первых, необходимо согласовываться с требованиями скорости потока. В зависимости от кодера это приводит при повышении битрейта к разного рода релаксациям при отборе сохраняемой части исходного сигнала, а при понижении -- наоборот, к ужесточению критериев. То есть на высоких скоростях потока (например, 256 или 320 кбит/с) появляется возможность сохранить больше звуковой информации, чем на низких. Однако даже после обработки с помощью психоакустической модели оставшаяся аудиоинформация достаточно объемна, поэтому и приходится идти на потери.

Во-вторых, после каждого изменения исходного сигнала (отброса сигнала какого-то одного диапазона) величина маскирующего эффекта меняется, и, проводя отброс в разной последовательности, мы можем получать сильно отличающиеся результаты. Этот момент сильно замедляет работу кодеров [2].

* 1. Битрейт

В формате mp3 - потоковом аудио, использующем сжатие c потерей качества, «битрейт» показывает степень сжатия потока и определяет размер канала, для которого сжат поток данных. Обычно битрейт измеряют в килобитах в секунду (англ. kilobit per second, kbps).

Существует три режима сжатия потоковых данных:

* с постоянным битрейтом (англ. Constant bitrate, CBR);
* с переменным битрейтом (англ. Variable bitrate, VBR);
* с усреднённым битрейтом (англ. Average bitrate, ABR).

Постоянный битрейт —кодирование потоковых данных, при котором пользователь в начале задаёт необходимый битрейт, который не меняется во всем файле. Его основной плюс — возможность точно рассчитать размер конечного файла.

В случае переменного битрейта кодек сам выбирает значение исходя из желаемого уровня качества, причём в течение одного фрагмента битрейт может меняться. Необходимый битрейт при сжатии звука изменяется на основе психоакустической модели. Данный вариант даёт лучшее соотношение качества и размера выходного файла, но его размер предcказать сложно.

Усреднённый битрейт можно назвать гибридом двух первых видов битрейта: битрейт в кбит/c должен задаваться пользователем, далее программа изменяет его в некоторых пределах.

* 1. Описание файлов wav формата

Waveform Audio File Format (WAVE, WAV, от англ. waveform — «в форме волны») — формат файла-контейнера для хранения записи оцифрованного аудиопотока, подвид RIFF. Этот контейнер как правило используется для хранения несжатого звука в импульсно-кодовой модуляции. Однако контейнер не налагает каких-либо ограничений на используемый алгоритм кодирования [12].

Данные wav-файлов хранятся в файлах в так называемом RIFF-формате (Resource Interchange File Format - формат файла для обмена ресурсами).

Файл в формате RIFF содержит вложенные фрагменты (chunk's). На рисунке 2 показано, что внешний фрагмент состоит из заголовка и области данных [13].

|  |  |  |
| --- | --- | --- |
| DWORD | DWORD |  |
| "RIFF" | Размер | Данные |

Рисунок 2 - Фрагмент "RIFF"

Первое двойное слово заголовка содержит четырехбуквенный код FOURCC, который идентифицирует данные, хранящиеся во фрагменте. Второе двойное слово заголовка - размер области данных в байтах (без учета размера самого заголовка).

Область данных имеет переменную длину, однако она должна быть выровнена на границу слова и при необходимости дополнена в конце нулевым байтом до целого числа слов[13].

Область, обозначенная на рисунке 2 как "Данные", может содержать внутри себя другие фрагменты. Для файла, в котором хранятся звуковые данные (wav-файл), эта область содержит идентификатор данных "WAVE", фрагмент формата звуковых данных "fmt " (три символа "fmt" и пробел на конце), а также фрагмент звуковых данных (рисунок 3). Файл может дополнительно содержать фрагменты других типов, поэтому заголовок wav-файла не имеет фиксированный формат. Например, в файле может присутствовать фрагмент "LIST" или "INFO", содержащий информацию о правах копирования и другую дополнительную информацию [14].

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DWORD | DWORD |  | | | | | | |
| "RIFF" | Размер | Данные | | | | | | |
|  | | "WAVE" | "fmt" | Размер | Формат данных | Фрагмент "data" | | |
|  | | | | | | "data" | Размер | Звуковые данные |

Рисунок 3 - Формат wav-файла

Область, обозначенная на рисунке 3 как "Формат данных", описывает звуковые данные [13].

* 1. Структура wav

WAV файл ([Windows PCM](http://audiocoding.ru/%D0%B0%D1%83%D0%B4%D0%B8%D0%BE-%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D0%B8%D0%BA/pcm-pulse-code-modulation.html)) представляет собой две, четко делящиеся, области. Одна из них – заголовок файла, другая – область данных. В заголовке файла хранится такая информация, как:

* размер файла;
* количество каналов;
* частота дискретизации;
* количество бит в сэмпле (глубина звучания) [15].

Более подробное описание структуры формата можно увидеть в таблице 1.3.

Длина всего заголовка составляет 44 байта, далее следует блок данных.

Таблица 1.3 - Структура WAV файла

|  |  |  |
| --- | --- | --- |
| Местоположение | Поле | Описание |
| 0..3 (4 байта) | chunkId | Содержит символы "RIFF" в ASCII кодировке (0x52494646 в big-endian представлении). Является началом RIFF-цепочки. |
| 4..7 (4 байта) | chunkSize | Это оставшийся размер цепочки, начиная с этой позиции. Иначе говоря, это размер файла - 8, то есть, исключены поля chunkId и chunkSize. |
| 8..11 (4 байта) | format | Содержит символы "WAVE" (0x57415645 в big-endian представлении) |
| 12..15 (4 байта) | subchunk1Id | Содержит символы "fmt " (0x666d7420 в big-endian представлении) |
| 16..19 (4 байта) | subchunk1Size | 16 для формата [PCM](http://audiocoding.ru/%D0%B0%D1%83%D0%B4%D0%B8%D0%BE-%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D0%B8%D0%BA/pcm-pulse-code-modulation.html). Это оставшийся размер подцепочки, начиная с этой позиции. |
| 20..21 (2 байта) | audioFormat | Аудио формат, полный список можно получить [здесь](http://audiocoding.ru/wav_formats.txt). Для[PCM](http://audiocoding.ru/%D0%B0%D1%83%D0%B4%D0%B8%D0%BE-%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D0%B8%D0%BA/pcm-pulse-code-modulation.html) = 1 (то есть, Линейное квантование). Значения, отличающиеся от 1, обозначают некоторый формат сжатия. |
| 22..23 (2 байта) | numChannels | Количество каналов. Моно = 1, Стерео = 2 и т.д. |
| 24..27 (4 байта) | sampleRate | Частота дискретизации. 8000 Гц, 44100 Гц и т.д. |
| 28..31 (4 байта) | byteRate | Количество байт, переданных за секунду воспроизведения. |
| Продолжение таблицы 1.1 | | |
| 32..33 (2 байта) | blockAlign | Количество байт для одного сэмпла, включая все каналы. |
| 34..35 (2 байта) | bitsPerSample | Количество бит в сэмпле. Так называемая "глубина" или точность звучания. 8 бит, 16 бит и т.д. |
| 36..39 (4 байта) | subchunk2Id | Содержит символы "data" (0x64617461 в big-endian представлении) |
| 40..43 (4 байта) | subchunk2Size | Количество байт в области данных. |
| 44.. | data | Непосредственно WAV-данные. |

* 1. Ограничение встраиваемой информации во фреймы аудио

Каждому стеганографическому приложению необходимо определенное соотношение между размером встраиваемого сообщения и устойчивостью этого сообщения к внешним воздействиям (в том числе и стегоанализу) [9].

Для большинства методов, которыми пользуются для сокрытия информации в цифровых контейнерах, складывается зависимость надежности системы от объема встраиваемых данных, показанная на рисунке 4.

Данная зависимость показывает, что при увеличении объема встраиваемых данных снижается надежность системы (при неизменности размера контейнера). Таким образом, используемый в стегосистеме контейнер накладывает ограничения на размер встраиваемых данных [9].

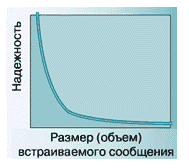


Рисунок 4 - Зависимость надежности системы от размера встраиваемого сообщения

Например, в wav-файл размером 121 кб в среднем можно разметить 5-7 байт скрываемой информации. Если размер звукового файла составляет 627 кб, в него можно внедрить сообщение, размер которого не больше 36 байт, при этом конечный mp3 файл займет около 57 кб [11].

* 1. Метод встраивания: замена наименее значащих битов

При внедрении информации в звуковые файлы методом НЗБ необходимо правильно выбрать номера разряда отсчета, в который можно поместить скрываемые данные.

Исследования показали, что для скрытой передачи информации можно использовать 2 младших разряда отсчетов звукового контейнера. При этом уловить на слух наличие внедренной информации невозможно [16].

Отсчеты представляют собой значения амплитуды в определенные моменты времени, значения которых могут быть как положительными, так и отрицательными.

Экспериментально установлено, что визуально можно обнаружить наличие вложения в сигнале при изменении девятого разряда в отсчете. Внедрение информации в разряды 10…16 путем визуального анализа спектров сигналов обнаружить практически невозможно. Данные оценки иллюстрирует рисунок 5

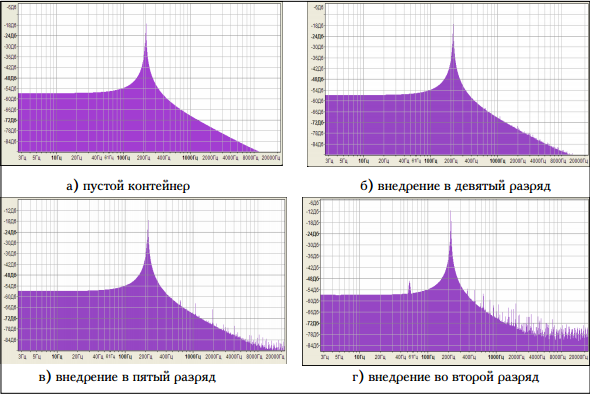


Рисунок 5 - Осциллограммы синусоидального сигнала с модифицированными битами

Искажения, вносимые методом замены наименьшего значащего бита в реальном звуковом сигнале, можно также визуально обнаружить путем анализа спектра сигнала.

В качестве контейнера использовался звуковой файл с уровнем квантования 16 бит, который содержал запись инструментального симфонического произведения. На рисунке 6а изображен спектр пустого контейнера.

Внедрение в шестнадцатые разряды (самые младшие) каждого отсчета не вносило заметных искажений (рисунок 6б). Изменения в десятом разряде надежно выявлялись путем анализа спектра (рисунок 6в), но плохо различались на слух. При внедрении информации в седьмой разряд искажения были различимыми даже на слух (рисунок 6г). Наиболее заметные искажения обнаруживались на участках с низким уровнем звукового сигнала («полная тишина»). Если исключить внедрение информации на участках с низким уровнем громкости, то для внедрения можно использовать разряды с 14 по 16.

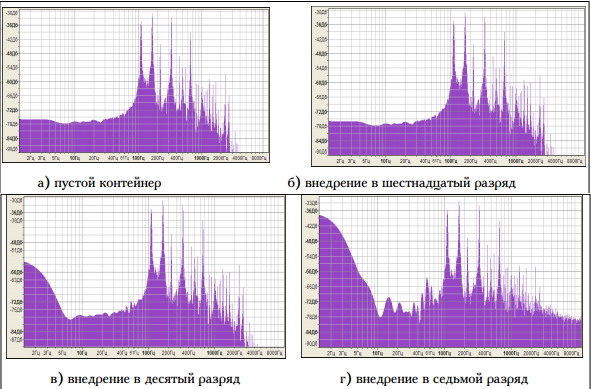


Рисунок 6 - Спектр звукового сигнала

Для повышения скрытности внедренной информации в программе использован модифицированный метод замены наименьшего значащего бита. Информация разделяется на фрагменты и распределяется по нескольким звуковым файлам. Такой подход позволяет осуществить защиту авторских прав не только на отдельный музыкальный файл, но и на весь альбом. В качестве контейнеров используются звуковые файлы формата WAV. Ключом для извлечения сообщения служит последовательность файлов, в которых были скрыты фрагменты сообщения. Для повышения степени защиты информации скрываемое сообщение можно предварительно зашифровать с использованием различных симметричных алгоритмов.

Если в контейнере содержится «полная тишина» (т. е. отсчеты с малой амплитудой), то программа пропускает их, внедряя информацию на других участках фонограммы. Для пропуска участков фонограммы с низким уровнем звука файл-контейнер разбивается на серии (блоки) от «полной тишины» до «полной тишины».

Ключ распределения информации генерируется в следующем виде: количество бит, заменяемых в одном отсчете, количество модифицируемых отсчетов в серии, количество серий, в которых сохранена информация. Например, если ключ 2:2:3, то контейнер будет заполнен следующим образом: файл разбивается на серии, количество которых не менее трех, в каждой серии содержится как минимум два отсчета, в которых два последних бита заменяются значимой информацией. Данный пример иллюстрирует рисунок 8. Серым цветом на рисунке 8 выделены отсчеты, в которых сохранена информация

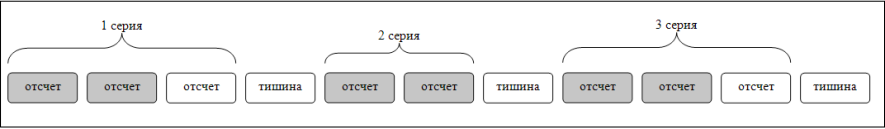


Рисунок 8 - Распределение информации в контейнере

* 1. Метод встраивания: цифровой водяной знак

ЦВЗ могут содержать различные данные: авторские права, идентификационный номер, управляющую информацию. Наиболее удобными для защиты с помощью ЦВЗ являются неподвижные изображения, аудио и видео файлы.

Основные требования, предъявляемые к водяным знакам: надёжность и устойчивость к искажениям, незаметности, робастности к обработке сигналов. ЦВЗ имеют небольшой объём, но для выполнения указанных выше требований, при их встраивании используются более сложные методы, чем для встраивания обычных заголовков или сообщений.

Перед помещением ЦВЗ в контейнер, водяной знак нужно преобразовать к подходящему виду.

Для повышения устойчивости к искажениям часто применяют помехоустойчивое кодирование или используют широкополосные сигналы. Начальную обработку скрытого сообщения делает прекодер. Важная предварительная обработка ЦВЗ — вычисление его обобщенного Фурье-преобразования. Эта новая функция описывает коэффициенты («амплитуды») при разложении исходной функции на элементарные составляющие — гармонические колебания с разными частотами. Это повышает помехоустойчивость. Первичную обработку часто производят с использованием ключа — для повышения секретности. Потом водяной знак «укладывается» в контейнер (например, путем изменения младших значащих бит).

Во многих стегосистемах для записи и считывания ЦВЗ используется ключ. Он может предназначаться для ограниченного круга пользователей или же быть секретным.

В стегодетекторе происходит обнаружение ЦВЗ в защищённом им файле, который, возможно, мог быть изменён. Эти изменения могут быть связаны с воздействиями ошибок в канале связи, либо преднамеренными помехами. В большинстве моделей стегосистем сигнал-контейнер можно рассмотреть как аддитивный шум. При этом задача обнаружения и считывания стегосообщения уже не представляет сложности, но не учитывает двух факторов: неслучайности сигнала контейнера и запросов по сохранению его качества. Учет этих параметров позволит строить более качественные стегосистемы. Для обнаружения факта существования водяного знака и его считывания используются специальные устройства — стегодетекторы. Для вынесения решения о наличии или отсутствии водяного знака используют, к примеру, расстояние по Хэммингу, взаимокорреляцию между полученным сигналом и его оригиналом.

* 1. Алгоритм шифрования DES

Был реализован данный алгоритм для шифрования встраиваемого сообщения, которое производится для того, чтобы информация не была доступна злоумышленникам.

Data Encryption Standard—симметричный алгоритм шифрования. DES имеет блоки по 64 бита и 16 цикловую структуру сети Фейстеля, для шифрования использует ключ с длиной 56 бит.

Входными данными для блочного шифра служат блок размером n бит и k-битный ключ. На выходе, после применения шифрующего преобразования, получается n-битный зашифрованный блок.

Шифр DES преобразует открытый текст из 64 битов следующим образом:

* производит начальную перестановку (IP);
* расщепляет блок на левую и правую половины;
* осуществляет 16 раундов с одним и тем же набором операций;
* соединяет половины блока;
* производит конечную перестановку.

Конечная перестановка обратна начальной. Разворачивание ключа дает 16 подключей по 48 битов каждый, выделяя их из 56-битного основного ключа.

Схема шифрования алгоритма DES указана на рисунке 9.

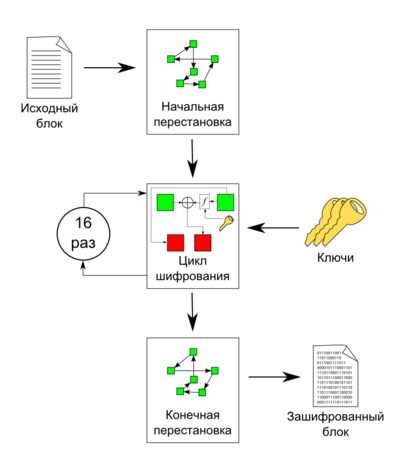


Рисунок 9 - Схема шифрования алгоритма DES

Ключи ki получаются из начального ключа k (64бит= 8 байтов или 8 символов в ASCII) следующим образом. Восемь бит, находящихся в позициях 8, 16, 24, 32, 40, 48, 56, 64 убираются из ключа kтаким образом чтобы каждый байт содержал нечетное число единиц. Это используется для обнаружения ошибок при обмене и хранении ключей. Затем делают перестановку для расширенного ключа (кроме добавляемых битов 8, 16, 24, 32, 40, 48, 56, 64) [9].

* 1. Аналитический обзор программ, реализующих методы стеганографии в аудио файлах

Известны три стеганографические программы, позволяющие встраивать информацию в звуковые файлы формата MP3.

1.13.1 MP3Stego

В 1998 г. в сети Интернет была представлена программа MP3Stego. Исходными данными для работы программы являются:

* файл ИКМ, 16-битное кодирование отсчетов, частота дискретизации 44 100 Гц (файл формата wav);
* текстовый файл, содержащий встраиваемое сообщение;
* секретный пароль.

Результат работы программы представляет собой звуковой файл формата MP3 со встроенным в него текстовым сообщением.

Биты встраиваемого сообщения кодируются значениями числа бит, необходимых для кодирования коэффициентов дискретно-косинусного преобразования (ДКП) и масштабирования кодом Хаффмана. Если необходимо закодировать нулевой бит, то значение этого числа должно быть четным, а если единичный, то нечетным [10].

Необходимую четность получают следующим образом: уменьшается шаг квантования; осуществляется квантование коэффициентов ДКП с новым шагом; подсчитывается число бит, необходимых для кодирования коэффициентов ДКП, и суммируется с числом бит, необходимых для кодирования коэффициентов масштабирования. Блок-схема алгоритма встраивания одного бита информации представлена на рисунке 10.

Минус данной программы заключается в нестойкости к атакам активного противника (к уничтожению скрытых данных).

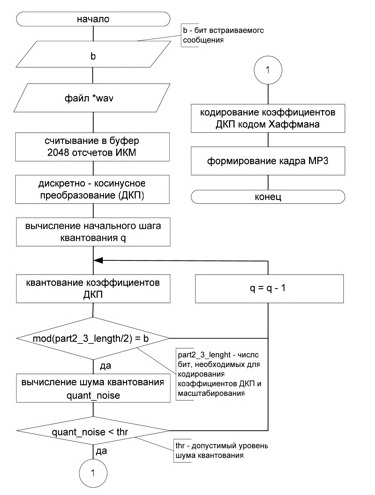


Рисунок 10 - Блок-схема алгоритма встраивания одного бита информации

1.13.2 Camouflage

Следующей стеганографической программой, позволяющей внедрять информацию в файлы MP3, стала программа Camouflage. Исходными данными для работы программы являются:

* звуковой файл формата МРЗ;
* внедряемый файл любого формата;
* пароль.

Алгоритм работы программы Camouflage заключается в дописывании внедряемых данных в конец файла. Перед внедрением данные шифруются по алгоритму Rijndael. Их объем не лимитирован [10].

Camouflage позволяет скрывать данные, скремблируя их, а затем помещая в выбранный файл. Скремблирование есть приведение информации к виду, по различным характеристикам похожему на случайные данные. Скремблирование выравнивает спектр сигнала, частоты появления различных символов и их цепочек. Этот камуфлированный файл, выглядит и ведет себя как обычный файл, то есть можно хранить и использовать его не привлекая внимания.

Например, можно создать файл изображения, которое выглядит и ведет себя так же, как любой другой файл изображения, но содержит скрытые зашифрованные файлы.

Такие файлы могут быть позже благополучно извлечены.

Для обеспечения дополнительной безопасности можно установить пароль на замаскированные файлы. Этот пароль будет требоваться при извлечении файлов.

1.13.3 Puff

Последней стеганографической программой, представленной в 2004 г., стала программа Puff. Для запуска программы необходимо задать:

* звуковой файл формата МРЗ;
* внедряемый файл любого формата;
* пароль;
* коэффициент сжатия;
* уровень скрытности.

На первом этапе работы программы внедряемый файл сжимается с заданным пользователем коэффициентом сжатия. На втором этапе осуществляется шифрование сжатого файла на ключе, генерируемом от пароля длиной от 16 до 32 символов. Третий этап заключается во внедрении зашифрованных данных в биты стаффинга звукового файла MP3. Данные биты служат для выравнивания скорости цифрового потока MP3 и при декодировании не используются. Блок-схема алгоритма работы программы Puff представлена на рисунке 11.

Согласно классификации методов внедрения информации все рассмотренные в статье программы реализуют форматные методы. Их суть заключается в поиске служебных полей формата, модификация которых не приводит к возникновению ошибок в приложениях, поддерживающих данный формат файла [10].

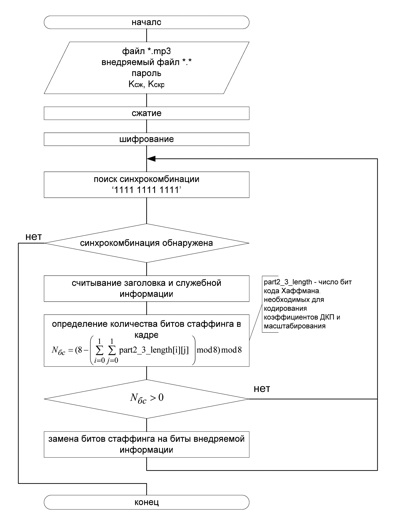


Рисунок 11 - Блок-схема алгоритма работы программы Puff

Для более эффективного использования контейнера необходимо применять методы внедрения информации, позволяющие размещать информацию в области данных файла.

* 1. Выводы по главе

В первой главе были описаны основные термины, понятия и методы, которые используются в работе, изучены принципы действия алгоритмов сжатия, шифрования и внедрения данных. В связи с ознакомлением с теоретическим материалом, появилась возможность сформулировать постановку задачи, которую необходимо решить в ходе данной работы.

* 1. Постановка задачи

Целью работы является разработка программного средства стеганографического сокрытия информации в mp3 файлах.

Необходимо реализовать три режима внедрения данных в аудиофайлы: вставка в заголовки, вставка в тело данных, вставка в конец.

Пользователь должен иметь возможность выбора аудиозаписи со своего компьютера, также должен выбирать из файлов своего компьютера текстовый файл с сообщением.

Реализовать в первом режиме внедрения данных чтение информации из заголовка mp3 файлов, возможность изменения существующих данных в тегах.

Во втором режиме внедрения необходимо реализовать шифрование скрываемого сообщения, сжатие и внедрение его в mp3 файл.

Третий режим реализует открытие mp3 файла для правки и внедряет зашифрованные данные в конец файла.

2 ПОСТРОЕНИЕ АЛГОРИТМОВ

В главе рассмотрено построение алгоритмов, которые использовались при реализации программы.

2.1 Структура программы

На рисунке 12 изображена блок-схема, описывающая обобщенный алгоритм работы всего приложения.



Рисунок 12 – Блок-схема работы приложения

На первой стадии алгоритма оператору предлагается выбрать режим работы приложения, выбрать mp3 файл и выбрать текстовый файл с сообщением, которое необходимо внедрить. Исходное сообщение шифруется. Затем происходит встраивание информации в битовый блок. Полученный файл сохраняется.

2.2 Встраивание ЦВЗ

Для обработки звуковых сигналов была создана программа, которая способна преобразовать исходный аудио файл формата mp3 в wav-файл и добавлять в него случайный водяной знак и сравнивать два сигнала – эталонный и проверяемый – делая вывод о наличии во втором водяного знака. Цифровой водяной знак будет представлен вектором бит сообщения. Длина вектора зависит от продолжительности аудиосигнала и размера стегоконтейнера.

При загрузке файла программа сначала считывает заголовок, оставшаяся часть файла записывается в буфер и разбивается на целое количество блоков, каждый из которых состоит из N каналов, где N = 2n .

Каждый сэмпл в файле состоит из NumChannels каналов. Каждый канал в свою очередь определяется BlockAlign / NumChannels байтами.

Произведя загрузку файла программа готова добавлять в него ЦВЗ. Разложив сигнал на спектр определим, какие частоты будут являться стегоконтейнером. Для звукового сигнала такое разложение будет естественным, поскольку звук фактически является набором волн различной частоты и амплитуды.

Чтобы обеспечить устойчивость (робастность) ЦВЗ, выбирать такие частоты нужно очень тщательно. Один из вариантов выбора – область наивысших (младших) частот, входящих в сигнал. Они в наименьшей мере определяют его вид, поэтому их изменение не вызовет слышимых помех. Выберем h таких частот. В таком случае фактическим контейнером водяного знака будет являться множество байт, соответствующих всему первому каналу. Это обуславливается тем, что даже незначительное изменение одной из частот разложения меняет весь сигнал.

Итак, мы имеем C стегоконтейнеров, состоящих из N значений выбранного канала. Выполняем C прямых преобразований с каждым из этих блоков. Результатом работы преобразований будут C векторов длины N, описывающих коэффициенты разложения на спектр частот. Первые коэффициенты полученных векторов соответствуют наиболее значимым (несущим) частотам, а последние ― младшим частотам. Соответственно наша цель – последние h частот. Практика показывает, что уже после 5й частоты вклад в вид сигнала каждой последующей несущественен.

Основываясь на этом можно взять h = N - 5 младших частот.

Получив спектры разложений и выделив в них младшие частоты, нужно добавить наш ЦВЗ в полученный контейнер. Сделать это можно разными способами, например ослабив громкие частоты и усилив тихие. Для этого вводятся два дополнительных параметра T и ф , где первый определяет порог амплитуды, а второй – величину изменения. Таким образом,

fij = af, если f < T & f/a, если f >= T

Введем правило, определяющее способ добавления битов знака в частотную область:

fij = f, при w = 0 & (af, если f < T & f/a, если f >= T) при w=1,

где w ― ЦВЗ и w (0,1,...,m), m = Ch.

Применим это правило к отобранным частотам спектров и выполним C обратных преобразований. Из полученных новых блоков, неизмененных каналов, старого заголовка и неизмененного остатка соберем новый wav-файл. Также отдельно сохраним файл, содержащий битовую последовательность.

2.3 Встраивание НЗБ

После выбора аудиозаписи пользователем и указания сообщения, которое необходимо скрыть, текст шифруется алгоритмом DES и внедряется.

Далее необходимо в теле аудиоданных найти либо незанятые биты, либо те биты, при замене которых, не приведут к искажению композиции. Поиск в описываемой программе осуществляется по модификации алгоритма "Наименее значащий бит", таким образом, что при обнаружении последовательности "111 111 11", она воспринимается, как та, которую без последствий возможно заменить и так далее.

При обнаружении такой последовательности первым делом проверяется к какой части данных она принадлежит, для этого мы читаем служебную информацию и заголовки.

Затем определяем количество битов, подлежащих замене, в кадре по формуле:

N = (8 - (∑1i=0∑1j=0len[i][j]) mod8) mod8.

Где len - число бит кода Хаффмана, необходимых для кодирования коэффициентов дискретно-косинусного преобразования, которое используется для формирования рабочей области контейнера.

При N> 0, происходит замена битов стаффинга на биты внедряемой зашифрованной информации. Если условие N> 0 не выполняется, то происходит возврат к поиску последовательности бит, для которых условие будет выполнено.

На выходе данного метода внедрения получается копия mp3 файла, воспроизводимая также, как и оригинал, но с зашифрованным сообщением.

Алгоритм работы приложения в режиме встраивания в тело файла показан на рисунке 13.

2.4 Алгоритм шифрования данных

В алгоритме DES используется один 64-битовый ключ: 56 бит значащие и 8 бит – проверочные для контроля на четность. Алгоритм обрабатывает блоки данных по 64 бита. Ключ разбивается на 16 отдельных 48-битовых подключей по одному на каждый раунд, который называется циклом Фейстеля*.*

Действие функции F. Структура функции F алгоритма DES схематически изображена на рисунке 17 приложения В.Каждый раунд алгоритма DES состоит из шести шагов.



Рисунок 13 – Общая блок-схема встраивания данных в тело файла

Шаг 1. Перестановка с расширением. Правая половина из 32 битов растягивается до 48 битов и перемешивается. Это помогает рассеиванию связи между входными битами и выходными. Перестановка с расширением (отличная от начальной) выбирается так, чтобы один входной бит воздействовал на две замены через S-блоки. Это помогает распространять зависимости и создает лавинный эффект (малое различие между двумя наборами входных данных превращается в большое на выходе).

Шаг 2. Сложение с подключом. К строке из 48 битов, полученной после перестановки с расширением, и подключу (его длина тоже 48 битов) применяется операция исключающего ИЛИ, т. е. каждая пара соответствующих битов складывается по модулю 2.

Шаг 3. Расщепление. Результат предыдущего шага расщепляется на 6 частей по 8 битов в каждом.

Шаг 4. S-блок. Каждый 6-битовый блок передается в один из восьми S-блоков (блоков подстановки), где он превращается в набор из 4 битов. S-блоки — нелинейные компоненты алгоритма. Каждый S-блок представляет собой поисковую таблицу из четырех строк и шестнадцати столбцов. Шесть входящих в S-блок битов определяют, какую строку и какой столбец необходимо использовать для замены. Первый и шестой бит задают номер строки, а остальные — номер столбца. Выход S-блока — значение соответствующей ячейки таблицы.

Шаг 5. Р-блок. На этот момент у нас есть восемь групп 4-битовых элементов, которые комбинируются здесь в 32-битовую строку и перемешиваются, формируя выход функции F.

Шаг 6. Дешифрование осуществляется при помощи того же самого ключа, который используется для шифрования, но с адресацией бит, видоизмененной так, чтобы дешифрование являлось бы обратным процессу шифрования.

2.5 Алгоритм внедрения данных в заголовок

На вход подается аудиозапись и текстовое сообщение, как и в предыдущих алгоритмах.

Затем происходит подсчет байт до необходимого тега заголовка, в данной работе используется тег комментариев, соответственно указатель съезжает на байт, откуда начинаются комментарии.

Далее сообщение шифруется, записывается в тег заголовка и сохраняется.

На выходе воспроизводимая копия аудиозаписи.

На рисунке 14 изображена блок-схема режима работы приложения "встраивание в тег".



Рисунок 14 – Блок-схема алгоритма встраивания в заголовок mp3 файла

2.6 Алгоритм встраивания методом НЗБ

Блок-схема встраивания данных в стегоконтейнер методом замены наименее значащих битов показана более подробно на рисунке 15.

На вход подается стегоконтейнер (mp3 или wav файл) и текстовое сообщение, которые необходимо скрыть.

Аудио файл разбивается на N блоков равной длины и затем поблочно вычисляется размер максимально возможного сообщения для внедрения. Если в контейнере не хватает места для выбранного сообщения, то появляется сообщение об ошибке с последующим возвратом к выбору или изменению файла с сообщением.

Если в число битов, которые можно заменить, помещается наше сообщение, то происходит внедрение путем замены битов, которые не несут полезного сигнала.

По завершении встраивания, контейнер сохраняется.

2.7 Алгоритм извлечения данных

На рисунке 17 показана блок-схема алгоритма извлечения сообщения из стегоконтейнера.



Рисунок 15 - Блок-схема встраивания данных в стегоконтейнер методом замены наименее значащих битов



Рисунок 16 - Блок-схема извлечения данных поблочно

2.7 Выводы по главе

В этой главе разработана общая схема работы программы, подробно рассмотрены основные схемы работы алгоритмов, выявлены особенности реализации программного средства, необходимые для дальнейшего программного конструирования. В главе были рассмотрены основные схемы работы для написания программного средства.

3 ПРОГРАММНОЕ КОНСТРУИРОВАНИЕ СОКРЫТИЯ ДАННЫХ МЕТОДОМ СТЕГАНОГРАФИИ

* 1. Обоснование выбора средств разработки

Большинство модулей программного средства, где необходима высокая производительность (например, кодирование или декодирование),реализованы на языке программирования С++. Был выбран именно этот язык, так в нем есть все необходимые компоненты для реализации выбранных алгоритмов. Также этот язык чаще требовался для решения поставленных задач в лабораторных и курсовых работах.

Общий проект реализовывался на языке Java с подключением библиотек С++ при помощи технологии JNI.

Для создания данного ПО использована среда разработки Eclipse. Эта среда очень проста в использовании. В ней есть все необходимые свойства для корректной работы программного средства.

Java Native Interface (JNI) — стандартный механизм для запуска кода, под управлением виртуальной машины Java(JVM), который написан на языках С/С++ или Ассемблера, и скомпонован в виде динамических библиотек, позволяет не использовать статическое связывание. Это даёт возможность вызова функции С/С++ из программы на Java, и наоборот.

Данная версия была выбрана из за того, что JNI изначально разрабатывался для обеспечения двоичной совместимости, для совместимости приложений, написанных с использованием JNI, для любых виртуальных машин Java на конкретной платформе. Поэтому скомпилированный С/С++ код должен исполняться Java-машинами например, в различных средствах разработки, для конкретной платформы. Более ранние интерфейсы не удовлетворяли условию двоичной совместимости.

3.2 Практическая реализация внедрения ЦВЗ

Алгоритм побитного сокрытия ЦВЗ, основанный на разбиении входного файла на блоки одинакового размера и записи в последние байты данных блоков замаскированных битов сообщения. Запись сообщения происходит за счет увеличения или уменьшения текущего байта входного файла на величину, зависящую от значения, полученного в результате логического добавления байта сообщения и индекса бита, что соответствует значению итератора основного цикла шифрования. Суть логической операции заключается в следующем: в цикле for к байту сообщения, получаемого из потока файла-сообщения в цикле while, логично добавляется значение двоичного представления числа 1, сдвинутого влево на значение, соответствующее текущей итерации цикла.

Реализация описанного выше процесса происходит в функции public void Hide(Stream messageStream, Stream keyStream), примерная схема показана на рисунке 17.

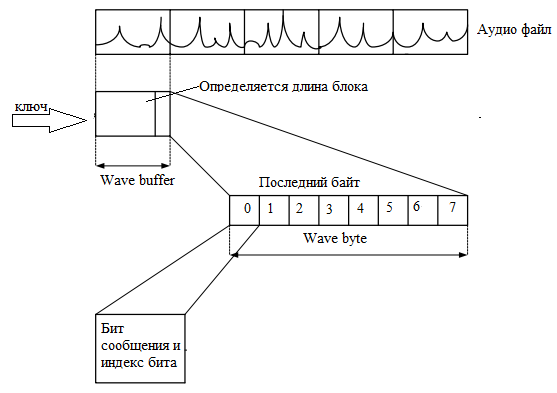


Рисунок 17 - Схема сокрытия данных

На вход функции подается поток байт сообщения и ключ. Также

Берем бит сообщения:

bit = (byte)(((message & (byte)(1 << bitIndex)) > 0) ? 1 : 0);

Сравнивая полученное значение со значением 0, т. е. определяя знак данного числа, получим значение, что будет необходимо для перестановки байта входного файла. Исходя из полученного значения и результата операции mod 2 с текущим байтом, выполняем увеличение или уменьшение данного байта.

if((bit == 1) && ((waveByte % 2) == 0))

{ waveByte += 1; }

else if((bit == 0) && ((waveByte % 2) == 1)) waveByte -= 1;

3.3 Извлечение данных

Входной файл снова делится на блоки, которые определяются значением ключа; для каждого блока выбирается последний байт в качестве байта со скрытым сообщением. Далее, для получения каждого из 8 бит, которые будут входить в байт сообщения, выполняем операцию mod 2 с текущим байтом. Полученное значение сдвигается на величину, соответствующую итератору цикла, и будет определять разряд этого бита. В этом же цикле к байту сообщения прилагается полученное значение. По выходу из цикла в поток сообщения вносится сформированный байт.

Это происходит в функции UnHide.

for(int bitIndex=0; bitIndex<8; bitIndex++)

Читаем байт ключа:

keyByte = GetKeyValue(keyStream);

Формируем блок:

for(int n=0; n<keyByte; n++)

{ sourceStream.Read(waveBuffer, 0, waveBuffer.Length); } Получаем последний байт блока:

waveByte = waveBuffer[bytesPerSample-1];

Получаем значение текущего бита:

bit = (byte)(((waveByte % 2) == 0) ? 0 : 1);

Записываем в байт сообщения полученный бит в соответствующий

разряд;

message += (byte)(bit << bitIndex).

3.4 Основные модули, используемые в программе

3.4.1 Модуль CmdWorker.java

Основной модуль программы. В нем реализованы основные функции: сокрытие данных, извлечение данных. Содержит следующие функции.

* Функция, осуществляющая сокрытие сообщения.

appendData(BaseParamparam)- в ней происходит шифрование данных, вывод шифрованных данных, сокрытие информации в файл, создание перед этим копии файла, в который и помещаются данные.

* Функция, осуществляющая извлечение ранее скрытых данных.

extractData(BaseParamparam) - реализовано извлечение данных из указанного аудиофайла, создание нового текстового файла и помещение извлеченных данных в этот файл.

3.4.2 Модуль des.java

Осуществляет шифрование данных. Содержит следующую функцию.

* Функция, осуществляющая шифрование по блоку.

des\_encrypt\_block(ulong64 src, ulong64 \*keys)

Осуществляет генерацию ключа, сохраненного ранее, для дальнейшего использования в функции извлечения данных.

* Функция, осуществляющая дешифрование по блоку. Считывает с файла ключ и выводит на экран полученные значения.

des\_decrypt\_block(ulong64 src, ulong64 \*keys)

3.4.3 Модуль Mp3TagAppender

Реализован класс Mp3TagAppender, в котором существуют следующие функции:

append(AppenderRequestrequest) - функция сокрытия данных в тегах заголовка mp3 файла версии ID3v1. Она получает указатель на запись в тег комментариев и вносит туда текст зашифрованного сообщения.

saveMprFile(AppenderRequestrequest, MP3Filemp3File) - функция, сохраняющая изменения, произведенные в заголовках аудиозаписи.

3.4.4 Модуль Mp3TagResolver

В данном модуле реализован класс Mp3TagResolver, содержащий функцию resolve(ResolverRequest request), которая извлекает сокрытые данные из тега комментариев.

3.4.5 Модуль TailDataAppender и TailDataResolver

Реализуют классы по сокрытию данных и извлечению их из конца аудиозаписи.

Функция append(AppenderRequestrequest) - приписывает зашифрованные данные после всех данных аудиофайла.

resolve(ResolverRequestrequest) - извлекает данные из конца файла после указателя APPEND\_TEXT.

3.5 Реализация алгоритма сокрытия данных в теги

Для скрытия сообщения, необходимо выбрать распределение бит, по которому будут внедряться в аудиозапись, соответствующие биты секретного сообщения. Так же необходима переменная для хранения исходного текста сообщения.

Код открытия файла и чтения информации из тегов.

fs = newFileStream(path, FileMode.Open);

// получаемфайлпоуказанномупути

byte[] buffer = newbyte[128];

// стандартныйразмертегаID3v1 128б

UTF8Encodingencoder = newUTF8Encoding();// конвертируемтегвстроку

string tag = encoder.GetString(buffer);

if (tag.Substring(0, 3) == "TAG")

// первые 3 байтасодержатслово TAG

При помощи следующих строк мы получаем информацию, содержащуюся в указанных полях тега.

Title = RemoveWhiteSpace(tag.Substring(3, 30));

Artist = RemoveWhiteSpace(tag.Substring(33, 30));

Album = RemoveWhiteSpace(tag.Substring(63, 30));

Year = RemoveWhiteSpace(tag.Substring(93, 4));

Comment = RemoveWhiteSpace(tag.Substring(97,28));

3.6 Диаграмма взаимодействия модулей

На рисунке 18 представлена диаграмма, которая описывает связи и взаимодействия между основными модулями программы.

На ней показано, что основной класс - это BaseParam, который содержит описание всех функций и команды.

Так как каждый алгоритм при шифровании и дешифровании алгоритмы сжатия и шифрования, то на диаграмме их изобразим связанными.

Рисунок 18 - Диаграмма связей

3.7 Выводы

В этой главе была рассмотрена структура программного средства, его основные модули, функции, их входные и выходные данные, подробно описана программная реализация методов стеганографии в mp3, приведены обоснования выбора средств разработки. В ходе выполненной работы было реализовано программное средство, соответствующее поставленным требованиям постановки задачи.

1. ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА И АНАЛИЗ ИСПОЛЬЗУЕМОГО АЛГОРИТМА
   1. Тестирование программы и программного интерфейса

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы изучены основные принципы построения стеганографических систем, позволяющих скрыть сам факт передачи сообщения. Рассмотрены алгоритмы встраивания скрываемой информации, их достоинства и недостатки, существующие программные средства, реализующие стеганографические методы скрытия информации. Были созданы блок-схемы работы программы в целом и отдельных подсистем, осуществлена постановка задачи разработки ПО, произведен обзор и тестирование разработанного программного средства.

Определены особенности встраивания информации в аудио файлы формата mp3, исследован новый метод внедрения данных.

Для встраивания информации в заголовок mp3 файлов были изучены виды тегов, состав заголовка и способы внедрения информации в поля существующих тегов.

Разработано программное средство, реализующее сокрытие информации в mp3 файлах, используя три режима внедрения: в заголовок, в тело данных, в конец файла. Реализован алгоритм сокрытия информации и извлечения уже ранее сокрытой информации модификацией методаLSB, алгоритм сжатия Хаффмана и алгоритм шифрования данных DES, а так же механизмы рассмотрения аудио файла на факт наличия скрываемого сообщения.

Программное средство прошло этапы разработки, тестирования и устранения ошибок. Данное программное средство может быть использовано для сокрытия информации и извлечения уже ранее скрытой информации, а так же для обнаружения факта встраивания информации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Стеганография [Электронный ресурс] – URL:<http://ru.wikipedia.org/wiki/Стеганография>
2. Компьютерная стеганография [Электронный ресурс] – URL:<http://nestego.blogspot.com/2012/04/blog-post_25.html>
3. СэломонД.Сжатие данных изображения и звука.–Техносфера, 2006
4. Основы электроакустики [Электронный ресурс] – URL:http://audioakustika.ru/node/1233
5. MP3 [Электронный ресурс] – URL:http://ru.wikipedia.org/wiki/MP3
6. Внутри MP3 – URL:http://habrahabr.ru/post/103635/
7. Обзор современных алгоритмов стеганографии [Электронный ресурс] URL:<http://technomag.edu.ru/doc/370605.html>
8. Сжатие аудиоданных [Электронный ресурс] - URL:http://ru.wikipedia.org/wiki/Сжатие\_аудиоданных
9. <http://citforum.ru/internet/securities/stegano.shtml>
10. Алгоритм DES [Электронный ресурс] - URL:http://ru.wikipedia.org/wiki/DES
11. Аналитический обзор программ, реализующих методы стеганографии в аудио файлах [Электронный ресурс] - URL:http://www.itsecurity.groteck.ru/articles2/Oborandteh/obzor\_steganogr\_programm\_dlya\_zvukov\_faylov\_formata\_mp3/
12. <http://ru.wikipedia.org/wiki/WAV>
13. <http://graphicon.ru/oldgr/courses/cg02b/assigns/hw-2/help/wavfmt2.htm>
14. Фролов А.В, Фролов Г.В " Мультимедиа для Windows", Москва, "Диалог-МИФИ", 1995 г.
15. [http://audiocoding.ru/статьи/структура-wav-файла.html](http://audiocoding.ru/%D1%81%D1%82%D0%B0%D1%82%D1%8C%D0%B8/%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0-wav-%D1%84%D0%B0%D0%B9%D0%BB%D0%B0.html)
16. <http://www.psuti.ru/referat/Alenin.pdf>
17. [Дэвид М. Ахмад, Идо Дубравский, Хал Флинн, Джозеф «Кингпин» Гранд, Роберт Грэм, Норис Джонсон, K2, Дэн «Эффугас» Камински, Ф. Уильям Линч, Стив Манзуик, Райян Пемех, Кен Пфеил, Рэйн Форест Паппи, Райян Расселл. Защита от хакеров корпоративных сетей](http://anyreads.com/read/#428603).Компания АйТи, ДМК Пресс, 2005. - 864с.
18. Алексеев А.П., Аленин А.А Скрытая передача данных в звуковых файлах формата WAV // ИКТ. Т.8, №3, 2010. – С.101-106.
19. Сергиенко А.Б. Цифровая обработка сигналов. СПб.: Питер, 2003. – 608 с.
20. Аленин А.А., Алексеев А.П. Исследование методов обнаружения вложений в звуковых файлах формата WAV // Безопасность информационных технологий. 2011. Т. 9. №1. С. 51-56.
21. Конахович Г. Ф., Пузыренко А. Ю. Компьютерная стеганография. Теория и практика. – «МК-Пресс», 2006. – 288 с
22. Барсуков В.С. Оценка уровня скрытности мультимедийных стеганографических каналов хранения и передачи информации. Специальная Техника. – 2000. – № 1.
23. Marvel L. Image Steganography for hidden communication. PhD Thesis. Univ.of Delaware, 1999.

Приложение А. Техническое задание на программное средство

|  |  |  |
| --- | --- | --- |
| «СОГЛАСОВАНО»  Рук. дипломной работы:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Долгов В.В.  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2013 г. |  | «УТВЕРЖДЕНО»  зав. кафедрой «ПОВТ и АС»  \_\_\_\_\_\_\_\_\_\_\_\_\_ Нейдорф Р.А.  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2013 г. |

П.А.1 Введение

П.A 1.1 Наименование программы

Наименование программы - «Стеганография в mp3 файлах».

П.A 1.2 Область применения

Областью применения данного программного средства является сфера образования.

П.A 1.3 Объект внедрения

Данное программное средство предназначено для внедрения на кафедру ПОВТиАС ДГТУ.

П.А.2 Основания для разработки

П.А. 2.1 Основания для проведения разработки

Разработка проводится на основании задания на преддипломную практику факультета «Информатика и вычислительная техника» (ИиВТ) Донского Государственного Технического Университета (ДГТУ) специальности «Компьютерная безопасность». Задание утверждено заведующим кафедрой «ПОВТ и АС» Нейдорфом Р.А.

П.А. 2.2 Наименование и условное обозначение темы разработки

Наименование темы разработки – «Стеганография в mp3 файлах».

П.А.3 Назначение разработки

П.А.3.1 Функциональное назначение

Функциональное назначение программного средства заключается в скрытой передаче информации в аудиозаписях.

П.А.3.2 Эксплуатационное назначение

Эксплуатационное назначение программного средства заключается в использовании его в сфере безопасности информации..

П.А.4 Требования к программе или программному изделию

П.А.4.1 Требования к функциональным характеристикам

Программное средство должно иметь понятный и удобный графический интерфейс. В меню программы входит выбор режима стеганографии в аудиозапись.

Необходимо реализовать три режима внедрения данных в аудиофайлы: вставка в заголовки, вставка в тело данных, вставка в конец.

Пользователь должен иметь возможность выбора аудиозаписи со своего компьютера, также должен выбирать их файлов своего компьютера текстовый файл с сообщением, либо вводить его вручную.

Реализовать ограничение на расширение выбранных файлов.

Реализовать в первом режиме внедрения данных чтение информации из заголовка mp3 файлов, возможность изменения существующих данных в тегах, возможность просмотра информации путем вывода в отдельное окно программы.

Во втором режиме внедрения необходимо реализовать два способа записи информации в тело данных: в первом способе скрываемое сообщение шифруется, сжимается и внедряется в mp3 файл, второй способ реализует преобразование mp3 файла в формат wav, внедряет шифрованные данные и делает обратное преобразование.

Третий режим реализует открытие mp3 файла для правки и внедряет зашифрованные данные в конец файла.

Программное средство должно корректно выполнять функцию встраивания исходного сообщения в контейнер. В качестве контейнера разрешается использовать только аудио файлы в формате mp3. При некорректной эксплуатации должно быть показано сообщение об ошибке.

П.А.4.2 Требования к надежности

Надежное функционирование программного средства должно быть обеспечено выполнением пользователем совокупности организационно-технических мероприятий, перечень которых приведен ниже:

* организацией правильного ввода данных инициализирующих структур;
* правильным функционированием технических средств;
* использованием лицензионного программного обеспечении;
* регулярным выполнением требований ГОСТ 51188-98. Защита информации. Испытания программныхсредств на наличие компьютерныхвирусов.

Отказы программы возможны вследствие некорректных действий оператора при взаимодействии с операционной системой.

П.А.4.3 Условия эксплуатации

Для функционирования программного продукта необходимо соблюдение всех требований и правил эксплуатации компьютерной техники. Дополнительных требований и ограничений не вводится. Требования к персоналу, работающему с данным программным продуктом: знания в области информатики и вычислительной техники, а также умение элементарного взаимодействия с ПО.

П.А.4.4 Требования к составу и параметрам технических средств

В состав технических средств должен входить IВМ-совместимый персональный компьютер (ПЭВМ), включающий в себя:

- процессор IntelPentiumили AMDс тактовой частотой не менее 2.0GHz;

- оперативную память объемом не менее 512 Мбайт;

- монитор,

- устройства ввода-вывода.

П.А.4.5 Требования к информационной и программной совместимости

П.А. 4.5.1 Требования к информационным структурам и методам решения

Требования к информационным структурам на входе и выходе, а также к методам решения не предъявляются.

П.А. 4.5.2 Требования к исходным кодам и языкам программирования

Требования к языкам программирования и средам разработки не предъявляются.

П.А. 4.5.3 Требования к программным средствам, используемым программой.

Для корректного функционирования программного средства необходима операционная система семейства Windows не ниже XP, предустановленные звуковые драйверы, реализациявиртуальной машины, необходимая для исполненияJava-приложений - JRE версии не ниже 1.6.

П.А. 4.5.4 Требования к защите информации и программ

Требования к защите информации и программ не предъявляются.

П.А.4.6 Требования к маркировке и упаковке

Требования к маркировке и упаковке отсутствуют.

П.А.4.7 Требования к транспортированию и хранению

Условия транспортирования, места хранения, условия складирования и сроки хранения в различных условиях должны соответствовать требованиям, предъявляемым к носителям информации на которых будет содержаться данное программное изделие. Приложение может храниться на жестком диске, на Flash-носителе, на компакт-дисках.

П.А.4.8 Требования к программной документации

В состав программной документации должно входить следующее:

* техническое задание на проект (ГОСТ 19.201-78 ЕСПД);
* руководство системного программиста по ГОСТ 19.503 — 79;
* руководство программиста по ГОСТ 19.504 — 79;
* руководство оператора по ГОСТ 19.505 — 79.

П.А.4.9 Стадии и этапы разработки

* Изучение предметной области (с 01.07.2013 по 07.07.2013):
  + получение информации об исследуемых методах;
  + исследование математического решения поставленной задачи;
  + поиск вариантов решения поставленных задач;
  + подготовка технического задания.
* Проектирование программного средства (с 11.08.2013 по 37.08.2013):
  + определение структуры программного комплекса (набор функций и процедур, достаточных для реализации);
  + определение структуры алгоритмов и модулей.
* Программная реализация (с 01.09.2013 по 1.11.2013):
  + разработка алгоритмической части;
  + получение дополнительной информации о стандартных функциях языка программирования:
  + разработка текстов программных модулей;
  + проектирование интерфейса.
* Отладка программного средства в статике (c 2.11.2013 по 5.11.2013):
  + объединение функций в единый модуль;
  + локализация ошибок, корректировка исходных текстов, перекомпиляция,
* Разработка технической документации (с 5.11.2013 по 24.11.2013):
  + изготовление документации (отчетов);
  + изготовление эксплуатационной документации (инструкции, руководства, методические указания).
* Тестовые испытания программного комплекса (с 25.11.2013 по 30.11.2013):
  + проверка программы на исключительные ситуации;
  + испытание на полноту функционирования.

П.А.4.10 Порядок контроля и приемки

Порядок и контроль приёмки определяются заведующим кафедрой «ПОВТ и АС» и основаны на демонстрации знаний технологии и умении создавать программные средства для различных предметных областей. Главным требованием к приемке является наличие правильно работающей программы иллюстрируемой тестовым примером и отчета, представленного в печатном виде.

Разработала

студентка группы ВКБ61 «\_\_\_» \_\_\_\_\_\_\_\_\_ 2013 год

Головина Д.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРИЛОЖЕНИЕ Б СТРУКТУРА MP3 ФАЙЛА



Диаграмма 1 – Структура MP3 файла

ПРИЛОЖЕНИЕ В СХЕМА ФУНКЦИИ F АЛГОРИТМА ШИФРОВАНИЯ DES

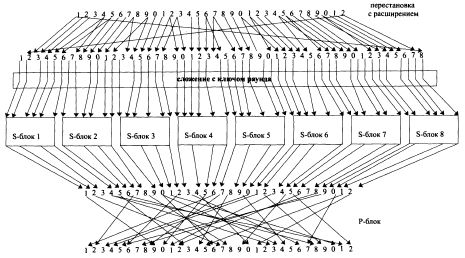


Рисунок 17 - Схема работы функции F алгоритма DES