Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт математики и фундаментальной информатики
Базовая кафедра вычислительных и информационных технологий

УТВЕРЖДАЮ
Заведующий кафедрой
_____ /В.В. Шайдуров

«17» июня 2016 г.

БАКАЛАВРСКАЯ РАБОТА

Направление 02.03.01 Математика и компьютерные науки

СТЕГАНОГРАФИЯ И СТЕГАНОАНАЛИЗ НА АУДИО ФАЙЛАХ

Научный руководитель
кандидат философских наук, доцент _____ / Б.В. Олейников
17.06.2016 г.

Выпускник _____ / А.В. Стрельников
17.06.2016

Красноярск 2016

РЕФЕРАТ

Выпускная квалификационная работа по теме: «Стеганография и стегоанализ на аудио файлах» содержит 76 страниц текста, 20 рисунков, одно приложение, 26 использованных источников.

СТЕГАНОГРАФИЯ, СТЕГАНОАНАЛИЗ, АУДИО ФАЙЛЫ, WAVE, МНОГОТОМНОСТЬ, LSB МЕТОД, МЕТОДЫ СЖАТИЯ, СТАТИСТИЧЕСКИЕ ТЕСТЫ.

Цель работы – рассмотрение проблем стеганографии и стегоанализа на аудио файлах.

Основные задачи: рассмотреть особенности сокрытия и обнаружения скрытых вложений в аудио файлах, создать приложение для сокрытия данных в WAVE файлах, предложить методы обнаружения скрытых данных в аудио файлах.

Разработанное приложение позволяет скрывать данные в аудио файлах формата WAVE любой битности и частоты дискретизации, а также определять наличие стеговложений в определенных видах аудио файлов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Стеганография - это метод передачи сообщений, который собственно скрывает само наличие связи. В отличие от криптографии, где заинтересованное лицо точно может определить является ли передаваемое сообщение зашифрованным текстом, методы стеганографии позволяют встраивать секретные сообщения в безобидные послания так, чтобы невозможно было заподозрить существование встроенного тайного послания.

Слово "стеганография" в переводе с греческого буквально означает "тайнопись" (steganos - секрет, тайна; graphy - запись). К ней относится огромное множество секретных средств передачи сообщений, таких как невидимые чернила, микрофотоснимки, условное расположение знаков, тайные каналы и средства связи на плавающих частотах и т. д.

Стеганография занимает свою нишу в обеспечении безопасности: она не заменяет, а дополняет криптографию. Сокрытие сообщения методами стеганографии значительно снижает вероятность обнаружения самого факта передачи сообщения. А если это сообщение к тому же зашифровано, то оно имеет еще один, дополнительный, уровень защиты [1].

Стеганография берет свое начало в древней Греции. Первые упоминания о реализации скрытой связи были найдены в летописи Геродота, написанной в V веке до нашей эры. В ней был описан метод незаметной передачи посланий, применявшийся в войне между Персией и Грецией. Сам метод заключался в следующем. В древние времена для письма использовались дощечки, покрытые воском. Для скрытой передачи сообщения с дощечки соскабливался воск, сообщение писалось прямо по дереву, затем воск наносился заново [2].

В настоящее время в связи с бурным развитием компьютерных технологий появилось новое направление стеганографии – компьютерная (или цифровая) стеганография, которая направлена на встраивание сообщений в различные типы файлов (текстовых, графических, аудио, видео и др.). В связи с возрастанием роли глобальных компьютерных сетей цифровая стеганография приобретает большую значимость. Анализ источников сети Internet позволяет сделать вывод [3], что в настоящий момент цифровая стеганография используется для следующего:

1. Скрытая передача сообщений, используемая для различных целей;
2. Защита конфиденциальной информации от несанкционированного доступа;
3. Преодоление систем мониторинга и управления сетевыми ресурсами;
4. Камуфлирование ПО;

5. Защита авторского права на определенные виды интеллектуальной собственности.

В настоящее время разрабатываются новые методы компьютерной стеганографии, основанные на особенностях представления информации в цифровом виде. Часть этих методов использует модификацию палитры, неточность устройств оцифровки, избыточность аудио и видео файлов и др. подходы [4].

Несмотря на бурное развитие стеганографических методов, в свободном доступе имеется недостаточно ПО для стеганографии в аудио файлах. Проблема связана с тем, что методы вложения информации в аудио файлы разных битностей несколько различны. В настоящее время не существует универсальных программных решений для работы с аудио файлами разных битностей. В данной работе разрабатывается такое ПО.

Цифровая стеганография может использоваться и для криминальных целей. К примеру, в пособии по обучению террориста «Технологичный муджахид, учебное пособие для джихада»[5] присутствует глава, посвященная стеганографии. В связи с этим большую актуальность приобретает стегоанализ.

Стегоанализ – наука о выявлении факта передачи скрытой информации [6]. Предпринимаются попытки решения задач стегоанализа, в частности, для графических файлов[7]. Однако, общих подходов обнаружения скрытых вложений для всех типов файлов пока что не создано. В настоящей работе также рассматриваются и эти задачи. Рассматривается метод стегоанализа, основанный на сжатии файлов, а также предложен новый метод, основанный на частотном анализе.

Для стеганографии на аудиофайлах и для рассмотренных методов стегоанализа разработано оригинальное программное обеспечение, реализованное в виде пакета Stegora WaveHide.

1 Основные понятия стеганоргафии и стеганоанаиза на аудио файах

1.1 Основные определения

Сообщение – любые данные, предназначенные для передачи.

Контейнер – исходный файл, предназначенный для сокрытия данных (сообщений).

Стегосообщение – информация, встраиваемая в контейнер.

Пустой контейнер – исходный файл без встроенного сообщения,

Заполненный контейнер – контейнер, содержащий встроенное сообщение.

Объем контейнера – максимально возможная часть файла, пригодная для встраивания (включения) сообщения, зависит от метода встраивания.

Процент заполнения контейнера – доля контейнерного объема (для заданного метода встраивания), занятая встроенным сообщением.

1.2 Выбор формата аудио файла

Формат WAVE был выбран из тех соображений, что он идеально подходит для реализации алгоритма LSB в силу своей избыточности. В области данных аудио файлов формата WAVE хранятся несжатые и никоем образом не измененные данные, полученные напрямую с аналогово-цифрового преобразователя, поэтому реализовывать стеганографические алгоритмы на файлах данного типа несколько проще и понятнее.

Поскольку WAVE файлы имеют достаточно большой размер, они не используются для обмена в сети интернет и для хранения музыки на портативных устройствах (плееры, мобильные телефоны). WAVE файлы используются там, где необходимо сохранить первозданный вид файла высокого качества там, где нет ограничения на размер свободного дискового пространства. К примеру, они используются на студиях звукозаписи в программах для редактирования аудио, где экономят время на сжатии и распаковке данных.

В дальнейшем под аудио файлом будет пониматься только файл формата WAVE.

1.3 Описание стандарта WAVE

WAVE (Waveform Audio File Format) создан инженерами Microsoft и Intel в августе 1991 года. Он разрабатывался в качестве стандартного формата хранения звуковых данных в операционной системе Windows 3.1[8].

1.3.1 Структура WAVE файла

WAVE-файл состоит из заголовка, в котором описан формат и все характеристики аудио файла, и непосредственно области звуковых данных[9].

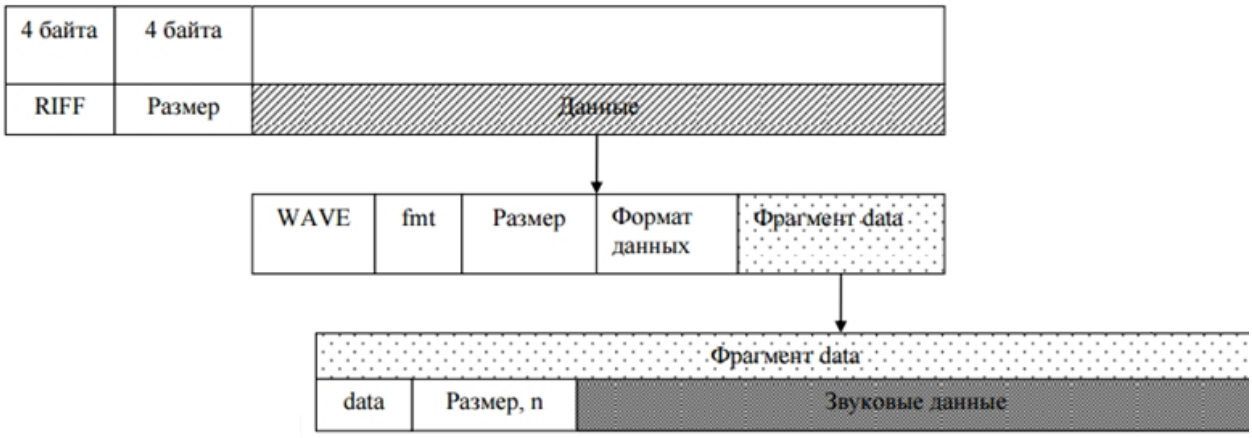


Рисунок 1 - Структура WAVE файла

1.3.2 Заголовок WAVE файла

Ниже в таблице приведены заголовок и описание полей [10] (здесь PCM – это импульсно-кодовая модуляция [11]).

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**. Это оставшийся размер подцепочки, начиная с этой позиции.
20..21 (2 байта)	audioFormat	Аудио формат. Для **PCM** = 1 (то есть, Линейное квантование). Значения, отличающиеся от 1, обозначают некоторый формат сжатия. Полный список в файле [12].

Местоположение	Поле	Описание
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.
32..33 (2 байта)	blockAlign	Количество байт для одного сэмпла, включая все каналы.
34..35 (2 байта)	bitsPerSample	Количество бит в сэмпле. 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 WAVE-данные.

1.3.3 Пример чтения заголовка из конкретного WAVE файла

Для примера описан файл «WhiteNoise.wav»., представляющий собой «белый шум» (рисунок 2).

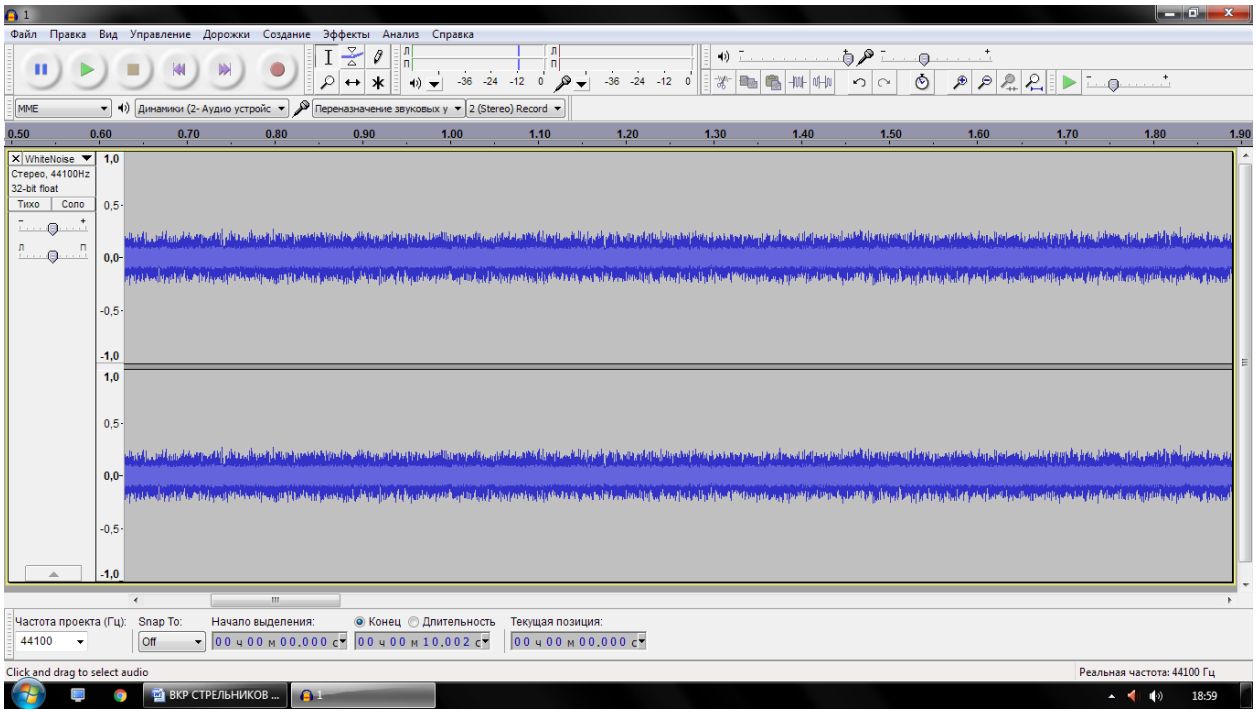


Рисунок 2 - Осциллограмма файла «WhiteNoise.wav»

1. 0..3 – символы ‘R’ ‘I’ ‘F’ ‘F’.
2. 4..7 – 2538452, размер файла в байтах.
3. 8..15 – символы ‘W’ ’A’ ‘V’ ‘E’ ‘f’ ‘m’ ‘t’ ‘ ’.
4. 16..19 – помещено число 16, указывающее, что это PCM.
5. 20..21 – число 3, PCM со сжатием
6. 22..23 – число 2 (количество каналов)
7. 24..27 – число 44100 – частота дискретизации
8. 28..31 – число 352800 – количество байт, переданных за секунду
9. 32..33 – число 8 – количество байт для одного семпла, включая все каналы
10. 34..35 – число 32 – количество бит в семпле
11. 36..39 – символы ‘d’ ‘a’ ‘t’ ‘a’
12. 40..43 – число 2538384 – количество байт в области данных

1.3.4 Определение области встраивания сообщений и необходимые понятия

Для задачи стеганографии и стегоанализа фрагмент data представляет аналогового сигнала. Набор значений фрагмента data и является основным источником внимания для организации вложений (в частности, методом LSB.[8]).

Звук состоит из колебаний, которые при оцифровке приобретают ступенчатый вид (см. рисунок 3). Этот вид обусловлен тем, что компьютер может воспроизводить в любой короткий промежуток времени звук определенной амплитуды (громкости) и этот короткий момент далеко не бесконечно короткий. Продолжительность этого промежутка и определяет частота дискретизации. Например, имеем файл с частотой дискретизации 44.1 КГц, 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 КГц.

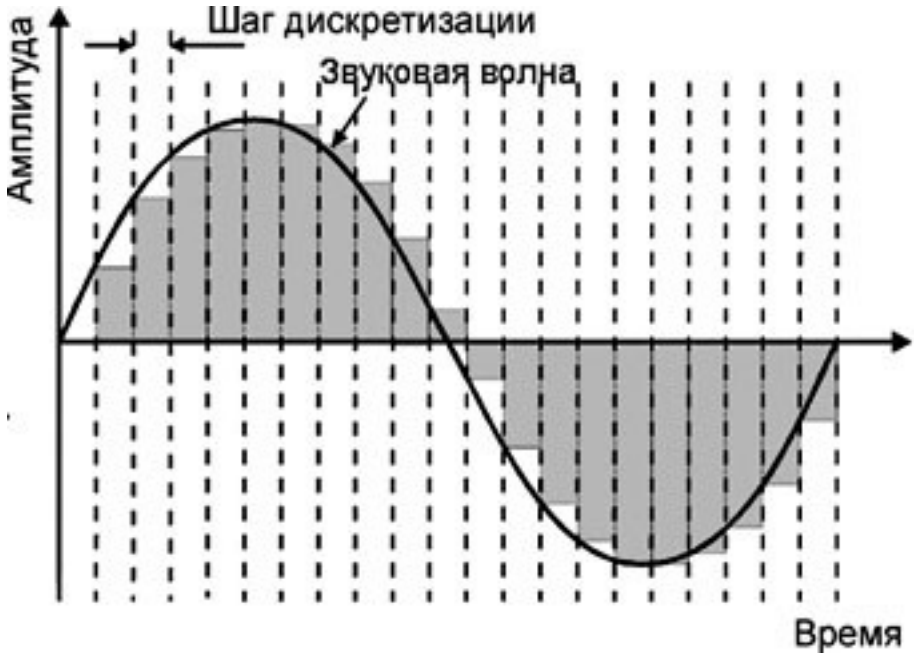


Рисунок 3 - Дискретизация звука.

Амплитудой называется громкость звука в определенный момент времени. 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.

Совокупность амплитуды и короткого промежутка времени носит название сэмпл.[10]

Битность – количество бит в сэмпле.

Частота дискретизации – частота взятия отсчетов непрерывного во времени сигнала при его дискретизации (в частности, аналого-цифровым преобразователем). Измеряется в герцах. Дискретизацией в общем случае называется преобразование непрерывной функции в дискретную. Дискретизация звука – преобразование аналогового звукового сигнала в цифровой. [13]

Итак, аудио файлы формата WAVE характеризуются битностью и частотой дискретизации.

2 Методы стеганографии для аудио файлов

2.1 Методы, исторически предлагаемые для стеганографии на аудио файлах.

2.1.1 Широкополосное кодирование.

В сигнал добавляется модулированный сообщением шум с амплитудой чуть выше предела маскирования. Преимуществом данной схемы является эффективность работы и высокая пропускная способность, недостатком - вносимые в сигнал слышимые искажения.

При сокрытии одного бита в последовательности коэффициентов выходная последовательность символов x_i' вычисляется следующим образом:

$$x_i' = \begin{cases} x_i + \omega_i |x_i| \alpha_i & s_i = 1 \\ x_i - \omega_i |x_i| \alpha_i & s_i = 1 \end{cases},$$

где $\omega_i \in -1, +1$ - случайная двоичная последовательность, α_i - порог слышимости i - той подполосы, x_i – символ исходной последовательности, s_i - скрываемый бит.

Для вычисления порога слышимости может быть использована психоакустическая модель, содержащаяся в формате кодирования MP3, или любая другая. Таким образом, метод позволяет управлять психоакустическим характером вносимых в сигнал искажений.

Для извлечения скрытого бита из последовательности коэффициентов используется функция корреляции принятых коэффициентов и исходной случайной последовательности. Следует отметить, что из-за ненадежности извлечения данный метод требует использования кодов коррекции ошибок. Это приводит к уменьшению как быстродействия, так и пропускной способности метода.

2.1.2 Фазовое кодирование.

В данном методе используется тот факт, что человеческое ухо воспринимает не значения фазы, а только их разность.

Сигнал разбивается на участки, значения фазы на первом участке используются для кодирования скрываемого сообщения, значения фаз остальных участков таким образом, чтобы разность фаз между участками осталась неизменной.

Для кодирования значений фаз, на множестве фаз выделяется набор равномерно распределенных значений, соответствующих битам 0 и 1. Значение фазы заменяется ближайшим значением, соответствующим

требуемому биту. Разность значений в наборе зависит от частоты полосы, и варьируется от $\frac{\pi}{12}$ на чувствительных полосах до $\frac{\pi}{4}$ на высокочастотных полосах.

Для кодирования одного бита скрываемого сообщения используется определенная последовательность изменений фаз, различная для кодирования 0 и для кодирования 1. Для извлечения скрытого сообщения используется следующая функция обнаружения:

$$q = \sum r_i(v_i - \varphi_i)^2 - r_i(u_i - \varphi_i)^2$$

где r_i, φ_i - амплитуда и фаза i- го полученного сигнала. $u = \{\alpha_0, \beta_1, \alpha_2, \beta_3\}$

$u = \{\alpha_0, \beta_1, \alpha_2, \beta_3\}$ - ожидаемая последовательность фаз при кодировании бита 1.

$v = \{\alpha_0, \beta_1, \alpha_2, \beta_3\}$ - ожидаемая последовательность фаз при кодировании бита 0.

α_i и β_i - ближайшие к φ_i значения фаз, соответствующие 1 и 0.

Если $q > 0$, бит скрытого сообщения принимается равным 1, иначе равным 0.

Метод обеспечивает высокую эффективность кодирования по критерию отношения сигнал-шум, однако его пропускная способность невелика, и составляет от 8 до 32 бит в секунду.

2.1.3 Эхо-кодирование.

Использует неравномерные промежутки между эхо-сигналами для кодирования последовательности значений. При наложении ряда ограничений соблюдается условие незаметности для человеческого восприятия. Эхо характеризуется тремя параметрами: начальной амплитудой, степенью затухания, задержкой. При достижении некоего порога между сигналом и эхом они смешиваются. В этой точке человек не может отличить эти два сигнала. Наличие этой точки сложно определить, так как она зависит от качества исходной записи и слушателя. Как правило, используется задержка около одной тысячной секунды, что вполне приемлемо для большинства записей и слушателей. Используются две различные задержки при кодирования нуля и единицы. Обе эти задержки должны быть меньше, чем порог чувствительности уха слушателя к получаемому эху.

Выше приведенные методы были взяты из работы[14]. Эти методы имеют ряд недостатков: они сложны для реализации и понимания, вносят явные искажения в аудио файл и имеют очень низкую пропускную

способность. Поэтому при разработке программного обеспечения для стеганографии в настоящее время они используются достаточно редко.

2.2 Современные методы сокрытия сообщений

2.2.1 Метод LSB (Least Significant Bit, Младший Значащий Бит)

Является методом, использующим избыточность звуковых файлов. Как известно, младшие разряды цифровых отсчетов содержат очень мало полезной информации. Их заполнение дополнительной информацией практически не влияет на качество восприятия, что и обеспечивает возможность скрытия. У данной группы методов имеется ряд отличительных особенностей. Сначала рассмотрим негативные особенности. С изменением информации искажаются статистические характеристики цифровых потоков. Ввиду этого для снижения компрометирующих признаков требуется коррекция статистических характеристик. К достоинствам можно отнести: возможность скрытой передачи большого объема информации, возможность защиты авторского права, скрытого изображения, товарной марки, регистрационных номеров и т.п. [15].

Пример работы метода LSB:

Исходный файл: (00100111 11101001 11001000)

(00100111 11001000 11101001)

(11001000 00100111 11101001)

Сообщение: 01000001

Результат работы метода:

(0010011**0** 11101001 11001000)

(0010011**0** 11001000 11101000)

(11001000 00100111 11101001)

Подчеркнуты только измененные биты исходного файла.

В силу своей простоты и прозрачности реализации метод LSB в настоящее время широко применяется в стеганографии. Вероятно, можно предложить и другие эффективные методы.

3 Обзор существующего программного обеспечения для стеганографии в аудио файлах.

3.1 Общие (желаемые) требования к реализации стеганографических методов для аудио файлов:

1. Сохранность целостности контейнера
2. Неотличимость на слух файлов с сообщением и без него
3. Файл не должен вызывать подозрений
4. Возможность работы с файлами любой битности
5. Возможность определять объем контейнера
6. Многотомность контейнера

В качестве общего замечания: все ниже перечисленное ПО для сокрытия информации использует метод LSB; ПО, использующее другие методы распространено не так широко.

3.2 Существующее ПО

3.2.1 DeepSound.

DeepSound [16] – свободно распространяемое ПО с дружественным интерфейсом, заявленной поддержкой форматов WAVE, APE и FLAC и встроенным конвертером. 2015 год. При попытке скрыть картинку в аудио файле, состоящем из тишины, файл-контейнер был наполнен шумами, как видно на рисунке 6.

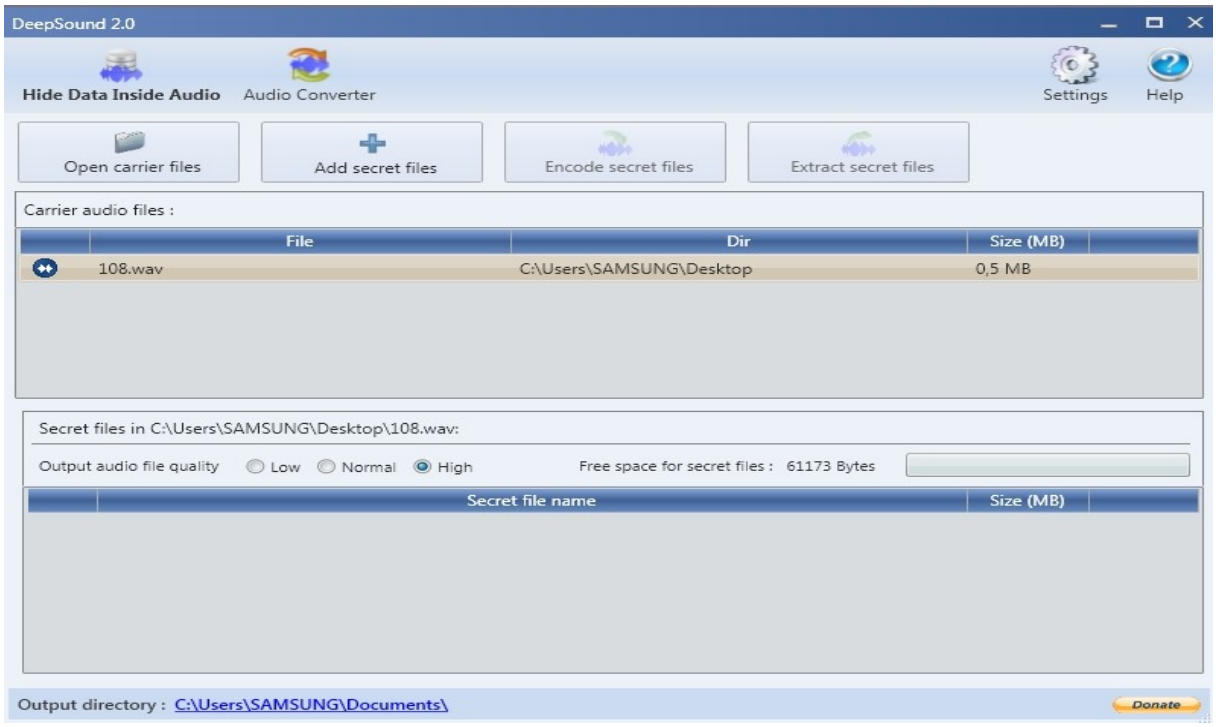


Рисунок 5 - DeepSound

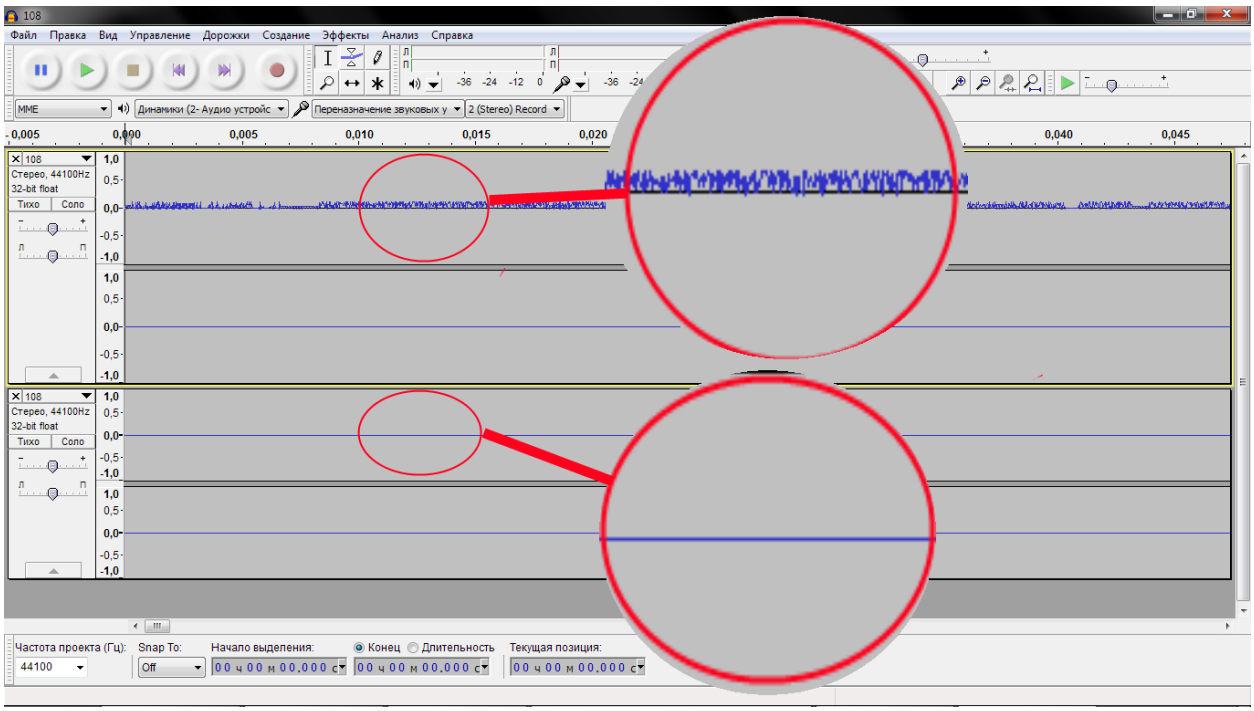


Рисунок 6 - Осциллограмма файла с вложением с помощью программы DeepSound(сверху) и файла тишины.

Отсюда можно сделать вывод, что программа работает некорректно хотя бы с каким-то процентом файлов, а именно вносит заметные человеческому уху изменения в аудио файл, следовательно, она не является универсальным стеганографическим решением для аудио файлов.

3.2.2 Xiao Steganography [17].

Свободно распространяемое ПО для сокрытия данных в файлах типа BMP и WAV. 2012 год. Есть возможность выбора рассеивающей функции(SHA, MD5, MD4, MD2) и алгоритма шифрования(RC2, RC4, DES). При попытке скрыть информацию были услышаны искажения (как на рисунке 6). Следовательно, это ПО опять же не является универсальным.

Рисунок 7 - Xiao Steganography

3.2.3 SilentEye [18].

Свободно распространяемое ПО для сокрытия данных в файлах разных типов, в том числе и WAVE. 2008 год. Отличилась весьма неудобным диалогом открытия файлов, а также отказом в работе с аудио файлами формата WAVE.

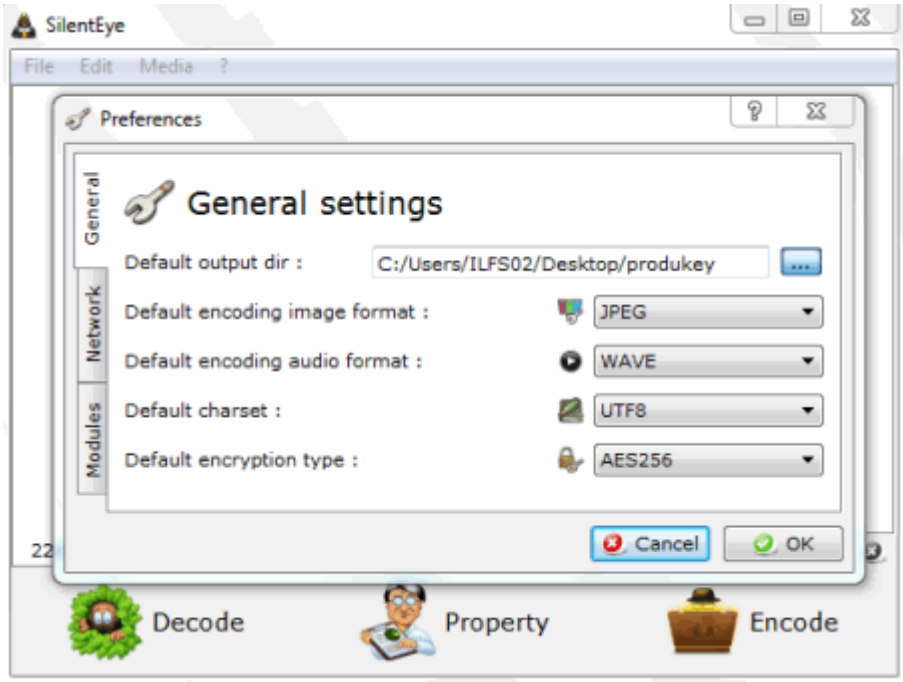


Рисунок 8 - SilentEye

3.2.4 StegoStick beta [19].

Свободно распространяемое ПО для сокрытия данных в файлах разных типов, в том числе и WAVE. 2013 год. Не отображается максимально

возможный размер контейнера, в файле со стегосообщением возникли шумы (как на рисунке 6).

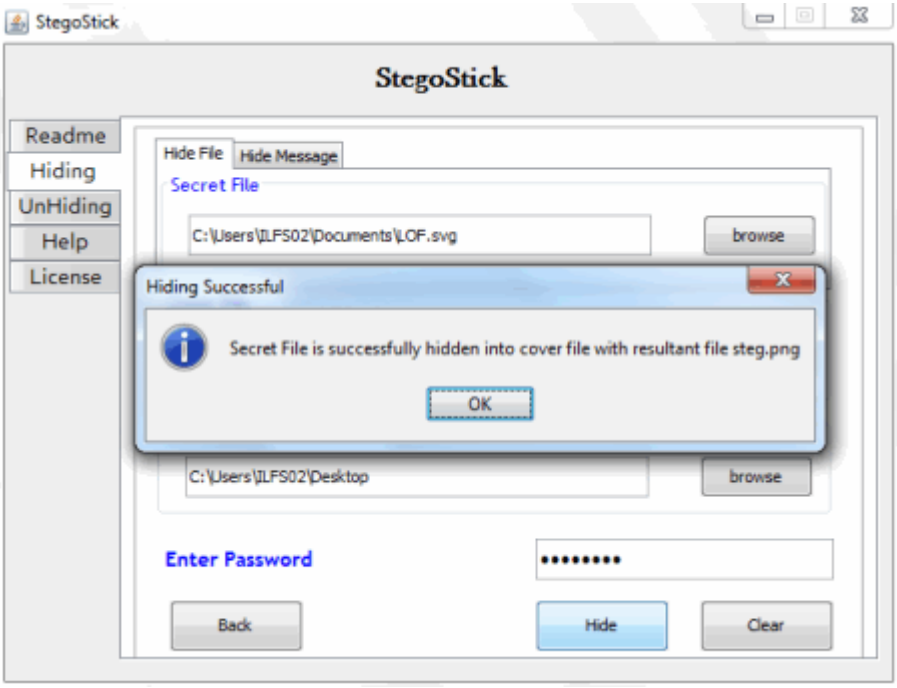


Рисунок 9 - StegoStick beta

Вывод по обзору.

Тестирование вышеуказанных пакетов показало, что ни один из них не соответствует изложенным выше требованиям к стеганографическим пакетам. Программы либо вносят слышимые искажения в аудио файлы, либо отказываются с ними работать.

4. Существующие методы и ПО для стегоанализа в аудио файлах

Для обнаружения вложений в аудио файлы исследователями в разное время были предложены следующие методы:

4.1 Поиск в фазовой области аудиоданных.

Алгоритм поиска вставок в фазовую область заключается в следующем [20]. При обработке каждого фрейма данных аудио сигнал извлекался и декодировался. В зависимости от параметров сжатия фрейм мог содержать 578 или 1156 отсчетов на канал. Таким образом, полученные данные разбиваются на фрагменты по 512 отсчетов. Для полученных фрагментов вычисляется 512 точечное быстрое преобразование Фурье. Далее по знаку мнимой части комплексного спектра определяются квадранты фаз гармоник и проводится частотный анализ этих значений. Затем отыскиваются преобладающие значения. Для этого вычисляются рейтинги пар квадрантов 1–2 и 3–4 по формуле

$$R_{i,j} = \frac{N_i + N_j}{N}$$

где $R_{i,j}$ — рейтинг пары квадрантов i,j ; N_i — частота встречаемости i- го квадранта; N — количество компонент в векторе фаз.

4.2 Стегоанализ аудио файлов на основе методов сжатия.

Идея предлагаемого метода заключается в том, что файл, содержащий однородные данные, имеет свою статистическую структуру [8]. Если использовать его в качестве контейнера для секретного сообщения, то после внедрения сообщения в контейнер нарушится статистическая структура контейнера и повысится его энтропия. Таким образом, при использовании алгоритмов сжатия исходный («пустой») контейнер сжимается, как правило, лучше, чем заполненный. Значит, если степень сжатия предполагаемого контейнера больше некоторого порогового значения, то с большой долей уверенности можно сказать, что контейнер пуст, в противном случае с большой долей уверенности можно судить о присутствии сообщения в контейнере. Метод анализа заключается в сравнении коэффициентов сжатия исходного контейнера и его полностью заполненной копии. Полностью заполненная копия получается при помощи псевдослучайного изменения младших бит исходного («пустого») контейнера. К обоим файлам применяется метод сжатия данных и анализируются их коэффициенты сжатия. Если эти коэффициенты близки по значению, то весьма вероятно, что исходный файл содержал скрытое сообщение. И, напротив, при большой разнице в коэффициентах сжатия выносится результат об отсутствии

скрытой информации в файле. При практической реализации этого метода контейнер и его заполненная копия рассматривается не целиком, а делятся на несколько равных частей. Это позволяет увеличить точность метода, кроме того, вводится дополнительный параметр, регулирующий соотношение ошибок I и II рода. Заметим, что метод анализа применяется не ко всему файлу, а именно к фрагменту файла, содержащему звуковые данные.

Формально разработанный алгоритм выглядит следующим образом. Пусть $X = \{x_1, \ldots, x_n\}$ – последовательность байт звуковых данных. $|X| = N$ – длина последовательности. Последовательность X разобьем на K равных отрезков. Пусть $\Phi(X)$ – алгоритм сжатия, примененный к X. Введем величину $f(X,n) = |\Phi(X_n)| / |X_n|$ -- коэффициент сжатия отрезка n последовательности X универсальным кодом Ф.

Обозначим через $\varphi(X)$ псевдослучайное изменение младших бит последовательности X (алгоритм LSB). Тогда $Y = \varphi(X)$ – заполненный контейнер X. Введем величину: $\Delta(X, n) = |f(X, n) – f(Y, n)|$

Для определения факта включения секретного сообщения выбирается пороговое значение для величины Δ и производится оценка количества отрезков, на которых значение величины не превышает порог. Если таких отрезков больше, чем половина от их общего количества, то считается, что исходная последовательность X содержала скрытые данные; в противном случае последовательность X считается «пустой».

4.3 Существующее ПО для стегоанализа

Путем исследования интернет источников удалось обнаружить, что в свободном доступе наибольшее количество пакетов по стегоанализу предусматривает работу в основном с графическими файлами, в частности, с файлами формата JPEG [21], [22]. Что касается аудио файлов, то для них программное обеспечение по стегоанализу обнаружить оказалось затруднительно (либо в свободном доступе оно вообще отсутствует), что подчеркивает актуальность работы в данном направлении.

5. Разработка приолжения для стеганографии и стегоанализа для аудио файлов

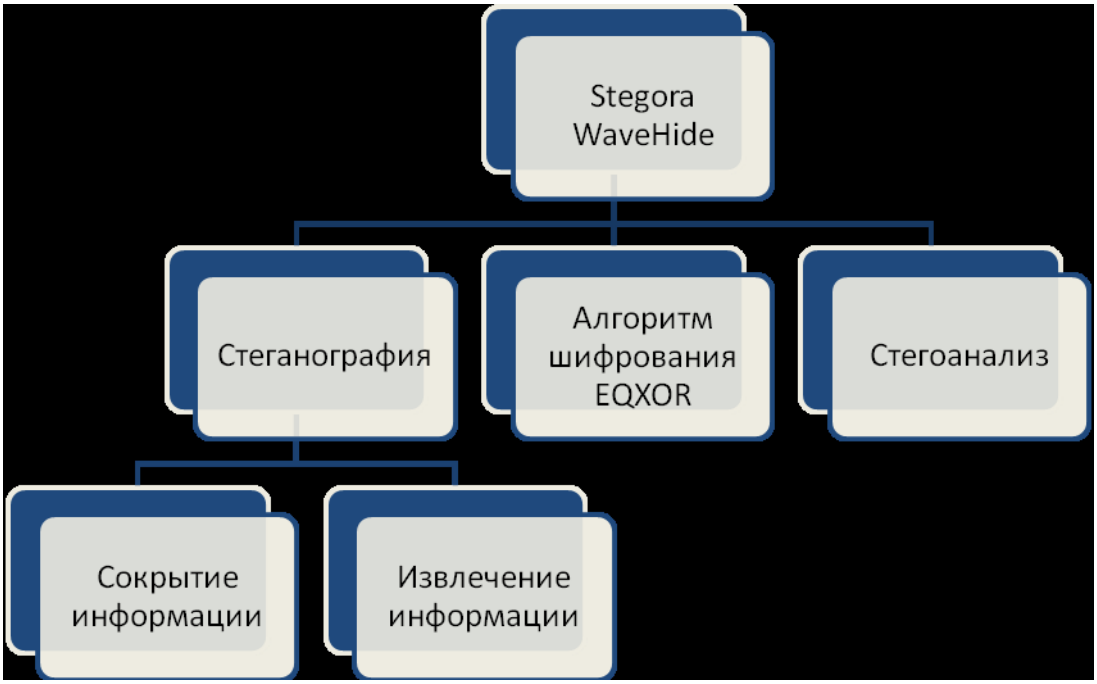
5.1 Назначение и структура ПО

Для стеганографии на аудиофайлах и для рассмотренных методов стегоанализа разработано оригинальное программное обеспечение, реализованное в виде пакета Stegora WaveHide.

Приложение выполняет стеговложения в аудио файлы формата WAVE любой частоты дискретизации и битности со свойством многотомности, а также в приложении реализована собственная методика обнаружения таких вложений и метод обнаружения сообщений в аудио файлах на основе методов сжатия.

ПО было разработано в среде Embarcadero Delphi XE5.

5.2 Программная структура пакета и процедуры



В приложении разработаны следующие процедуры и функции:

1. Функция FromBytesToCardinal – переводит данные из типа Byte в тип Cardinal. Используется в процедуре WaveUnHide.
2. Процедура FromCardinalToBytes – переводит данные из типа Cardinal в тип Byte. Используется в процедуре HideWave.
3. Функции ByteToBin и BinToByte – переводят данные из типа Byte в тип Bin и обратно (bin – массив [0..7] элементов из множества 0..1). Используются в процедурах HideWave, WaveUnhide.
4. Процедура first – реализует EQXOR-шифрование. Используется в процедуре HideWave.

5. Процедура last – реализует EQXOR-дешифрование. Используется в процедуре WaveUnHide.
6. Процедура HideWave – реализует LSB алгоритм стеганографического сокрытия в WAVE-файле.
7. Процедура WaveUnHide – реализует извлечение информации, сокрытой при помощи алгоритма LSB.
8. Процедура Button2Click – запрашивает имена файлов для извлечения скрытой информации и вызывает процедуру WaveUnHide.
9. Процедура Button3Click – запрашивает имя файла-контейнера для дальнейшего использования, выводит размер контейнера.
10. Процедура Button1Click – запрашивает имя файла, который требуется скрыть, вызывает процедуру HideWave.
11. Процедура BitBtn1Click – запрашивает имя файла-ключа.
12. Процедура Button4Click – реализует метод стегоанализа, основанный на частотном анализе.
13. Процедура Button5Click – выводит результаты статистических тестов NIST.
14. Процедура Button6Click – реализует метод стегоанализа, основанный на алгоритмах сжатия.

5.3 Описание стеганографического блока программы

В программе в качестве стеганографического алгоритма был реализован алгоритм LSB. Как уже говорилось ранее суть данного алгоритма состоит в замене наименьших значащих битов аудио файла битами сообщения. На вход программе подается файл-контейнер, файл-сообщение, а также ключевой файл.

Файл-сообщение шифруется с помощью алгоритма EQXOR и ключевого файла.

Суть алгоритма EQXOR состоит в следующем: берется две последовательности бит (в рассматриваемом случае первая последовательность это файл-сообщение, вторая – ключевой файл). К примеру, 01001001 и 11010100. Если i-й бит первой последовательности аналогичен биту второй, то в выходной последовательности на i-том месте пишется единица, иначе – ноль. Таким образом, для наших последовательностей из примера результат будет таким: 01100010. (В сущности, это алгоритм XOR, только логической операцией вместо «исключающего ИЛИ» выступает операция эквивалентности[23]).

Затем получаем размер зашифрованного файла и вместимость контейнера. Вместимость контейнера вычисляется по формуле:

$$Size = \frac{F}{BPS} - 132,$$

где F – размер файла-контейнера, BPS – количество байт на семпл (имеется в заголовке аудио файла формата WAVE, считывается непосредственно из файла).

Затем в зависимости от количества байт на семпл последовательно производится замена двух младших битов файла-контейнера (если BPS = 1, то замене подвергается каждый байт данных файла-контейнера, если BPS = 2, то каждый третий, и т.д.). Вначале записывается размер скрываемого файла в байтах. Замена производится до тех пор, пока все биты файла-сообщения не будут записаны, либо до конца файла-контейнера. Если достигнут конец файла-контейнера, то выводится сообщение о том, что контейнер закончен, и нужно выбрать следующий контейнер.

В программе это реализовано следующим образом.

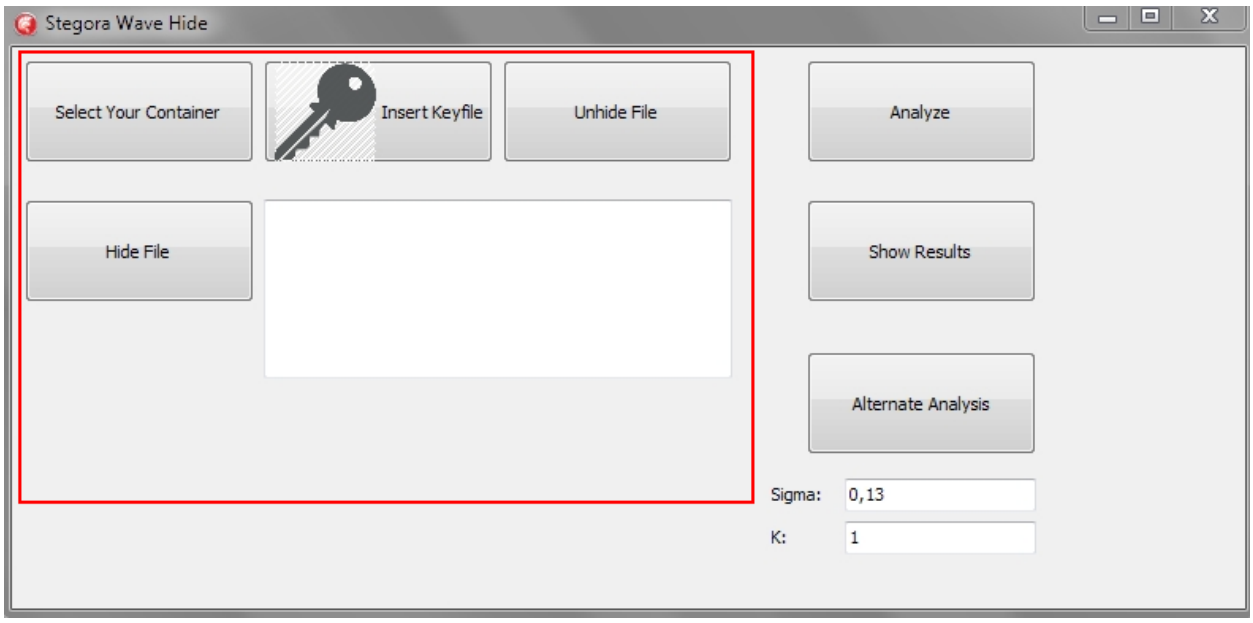


Рисунок 10 - Главное окно программы. Красным прямоугольником выделена стеганографическая часть.

В данной части программы имеется ряд процедур и функций: first, last, HideWave, BitBtn1Click, WaveUnHide.

Порядок действий при сокрытии файлов следующий:

1. Нажатием на кнопку «Insert Keyfile» вызывается процедура BitBtn1Click. В процедуре выбирается ключевой файл через стандартный инструмент Delphi OpenDialog. Глобальной переменной Ideal присваивается имя ключевого файла.

2. Нажатием на кнопку «Select Your Container» выбирается файл-контейнер. В стандартном текстовом редакторе Memo1 выводится

размер контейнера (количество килобайт максимально возможной информации для сокрытия).

3. Нажатием на кнопку «Hide File» выбирается скрываемый файл. Затем вызывается процедура first, в которой реализуется алгоритм XOR для выбранного и ключевого файла. Результатом процедуры является зашифрованный файл, который впоследствии и будет скрыт. Затем в процедуре считывается информация о битности аудио файла. И, наконец, происходит само сокрытие до тех пор, пока не достигнут конец файла-контейнера, либо зашифрованного файла-сообщения. Сокрытие происходит следующим образом: в аудио файле, последовательно, в зависимости от его битности, два последних значащих бита заменяются битами зашифрованного файла-сообщения. Если достигнут конец файла-сообщения, то в контейнер дописываются байты исходного файла. Если же достигнут конец файла-контейнера, то пользователю необходимо выбрать дополнительный контейнер.

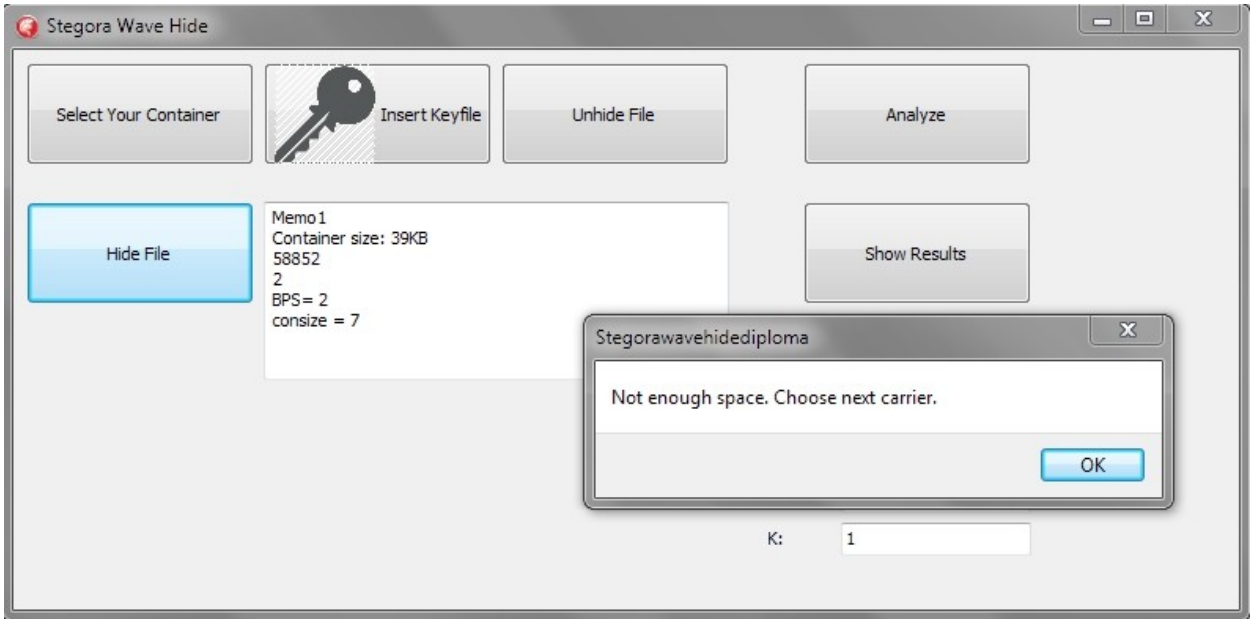


Рисунок 11 - Требование программы предоставить новый контейнер для сокрытия информации.

На выходе получается аудио файл с вложением, идентичный изначальному файлу по размеру, а также по звучанию. Результаты работы программы приведены на рисунке 12 и рисунке 13.

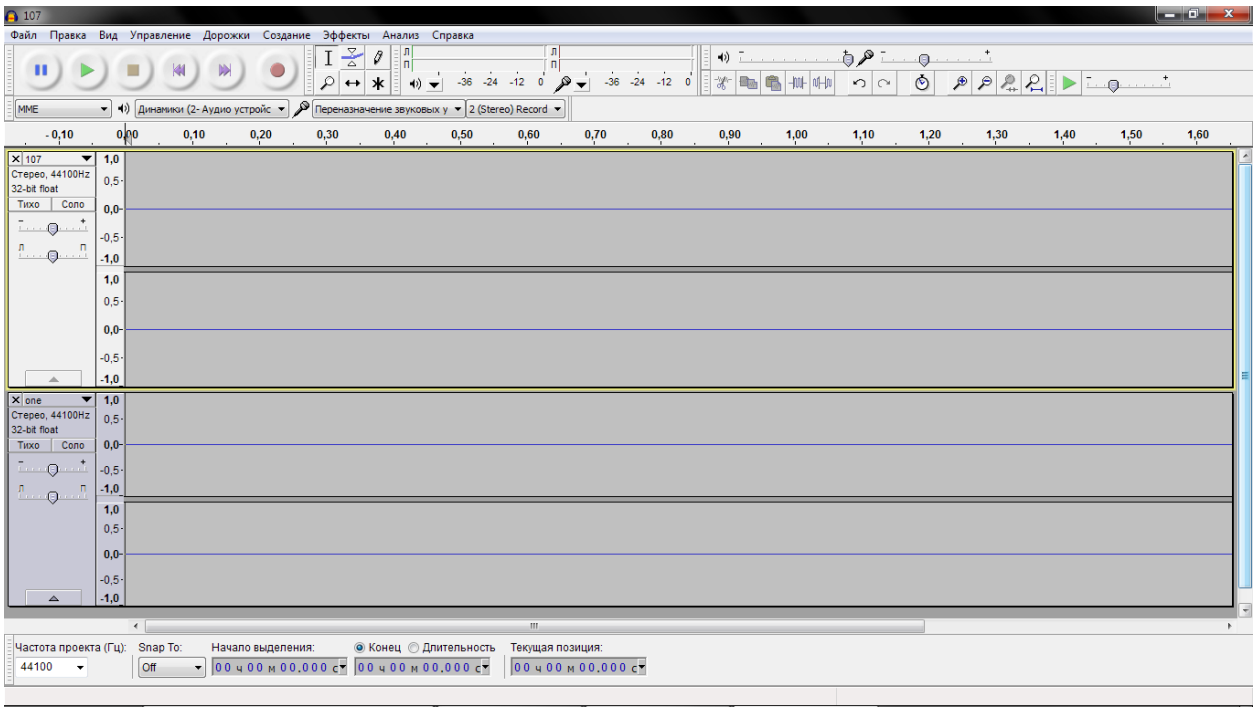


Рисунок 12 - Осциллограмма «пустого» аудио файла (сверху, 107.wav) и аудио файла с вложением с использованием ПО Stegora WaveHide(снизу, one.wav)

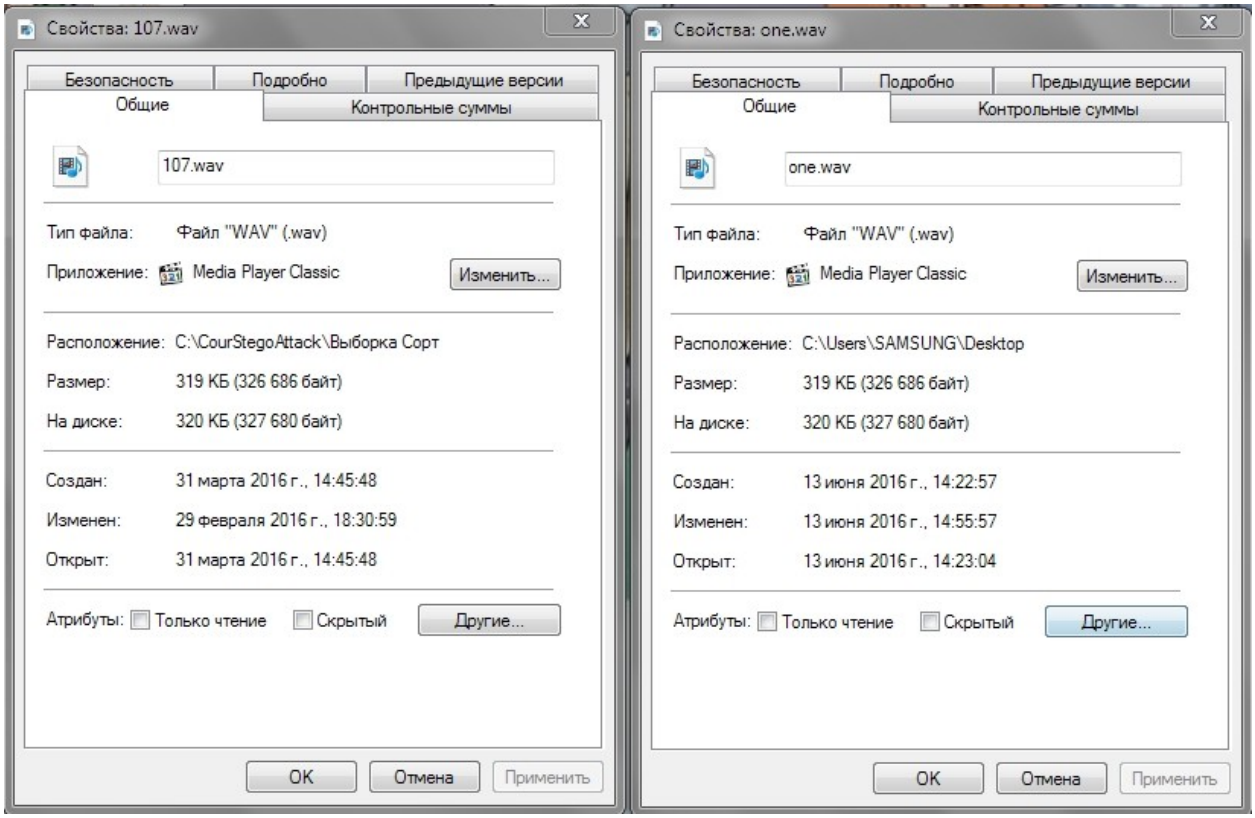


Рисунок 13 - Соответствие размеров файла с вложением и без

5.4 Процедура извлечения стегосообщения из файла.

На вход подается аудио файл-контейнер и ключевой файл. Из файла-контейнера считывается его битность, а также размер скрытого в нем файла-сообщения. Последовательное считывание младших бит идет согласно размеру файла-сообщения. Если достигнут конец файла-контейнера, то пользователь подает на вход программе еще один файл-контейнер.

В программе это реализовано следующим образом.

Порядок действий при извлечении стегосообщения следующий:

1. Нажатием на кнопку «Insert Keyfile» вызывается процедура BitBtn1Click. В процедуре выбирается ключевой файл через стандартный инструмент Delphi OpenDialog. Глобальной переменной Ideal присваивается имя ключевого файла.

2. Нажатием на кнопку «Unhide File» вызывается процедура WaveUnHide. Пользователю требуется выбрать файл со стегосообщением. В начале из файла считывается его битность. Затем считывается размер сокрытого файла. Запускается цикл от нуля до размера сокрытого файла, в котором согласно битности производится считывание двух последних бит и запись их в выходной файл. Если достигнут конец файла, то пользователю необходимо предоставить программе дополнительный стего контейнер (рисунок 14).

На выходе получается файл-сообщение с неопределенным форматом. Предполагается, что формат файла-сообщения известен получателю.

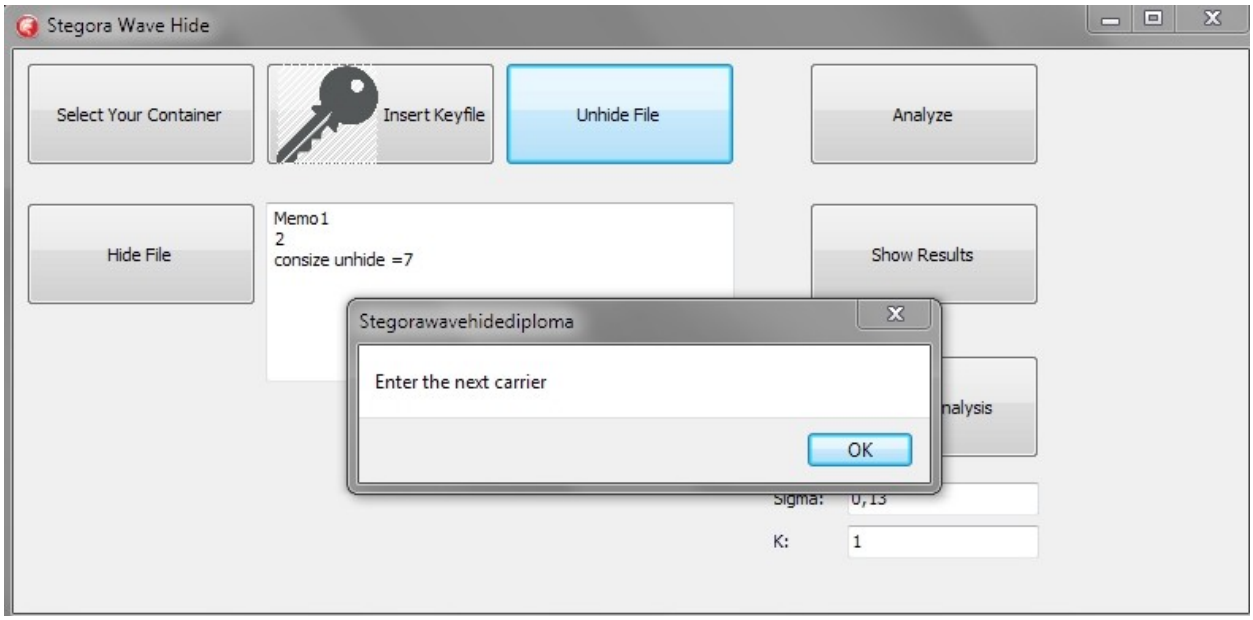


Рисунок 14 - Требование программы предоставить следующий контейнер для извлечения информации

6 Описание стегоаналитического блока программы

6.1 Описание разработанного подхода стегоанализа.

В основу предлагаемого подхода положена **гипотеза о не случайности** наименьших значащих бит аудиофайлов [24]. Хотя, существует мнение, что НЗБ (Наименее Значащие Биты) аудиофайлов являются случайными.

На самом деле это не так. Хотя человеческое ухо и не заметит изменений звукового файла при изменении последних битов, статистические параметры звукового файла будут изменены.

Перед сокрытием данные обычно архивируются (для уменьшения объема) или шифруются (для обеспечения дополнительной стойкости сообщения при попадании в чужие руки). Это делает биты данных очень близкими к случайным. В этом случае последовательное встраивание такой информации заменит НЗБ звукового файла случайными битами. В этом и кроется основная суть подхода – улавливание разницы между распределением младших бит в пустом и заполненном контейнерах.

Для проверки этого подхода была разработана определенная методика, предполагающая использование пакета статистических тестов NIST, разработанного национальным институтом стандартов и технологий (США, 2014 год) [25].

Тесты пакета NIST были разработаны для статистической проверки двоичных последовательностей на случайность [26]. Эти тесты основаны на статистических свойствах, которыми обладают только случайные последовательности.

Для отработки методики была создана тестовая база из 108 разнотипных аудио файлов, предполагая использование их в качестве контейнера. В базу были включены шумы (белый, черный, коричневый, розовый, зеленый и т.п.), ровные синусоиды, записи речи, инструментов (гитара, саксофон, барабаны и т.п.), а также полноценные музыкальные композиции. База состоит из файлов разных битностей (8, 16, 24, 32 бита) и частот дискретизации. Размер файлов от 38 КБ до 41522 КБ.

Чтобы отслеживать поведение случайности младших бит, необходимо упорядочить файлы по какому-либо признаку. В качестве такого признака было выбрано относительное количество нулевых байт в файле. Относительное количество нулевых байт определяется как отношение нулевых байт файла к общему количеству байт. В соответствии с этим признаком были упорядочены все 108 тестовых файлов.

Для проверки гипотезы в исходную базу (кроме пустых контейнеров) были добавлены частично заполненные, а именно, во все файлы из базы были осуществлены стеговложения (при помощи разработанного ПО Stegora

WaveHide) на 10%, 50% и 100% от максимальной возможности рассматриваемого стегоконтейнера.

В качестве теста был использован частотный побитовый тест пакета NIST [26], который оценивает соотношение между нулями и единицами в двоичной последовательности. Алгоритм этого теста заключается в следующем [25]: файл рассматривается как битовая последовательность, при этом единица принимается за +1, ноль за 0. Считается сумма последовательности. Затем вычисляется статистика по формуле:

$$S_{obs} = \frac{|S|}{\sqrt{n}},$$

где |S| – сумма последовательности, n – количество элементов в последовательности.

Вычисляется P-значение через дополнительную функцию ошибок:

$$P_{value} = erfc(\frac{s_{obs}}{\sqrt{2}}),$$

Дополнительная функция ошибок вычисляется так:

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-t^2}\, dt$$

И если результат больше, чем 0,01, то последовательность признается случайной с уровнем доверия 99%.

6.1.1 Тестирование базы файлов на предмет вложений с помощью частотного анализа

С помощью рассмотренного выше теста последовательности наименьших значащих бит всех 108 файлов из базы были проверены на случайность.

Результаты представлены на рисунках 15-18.

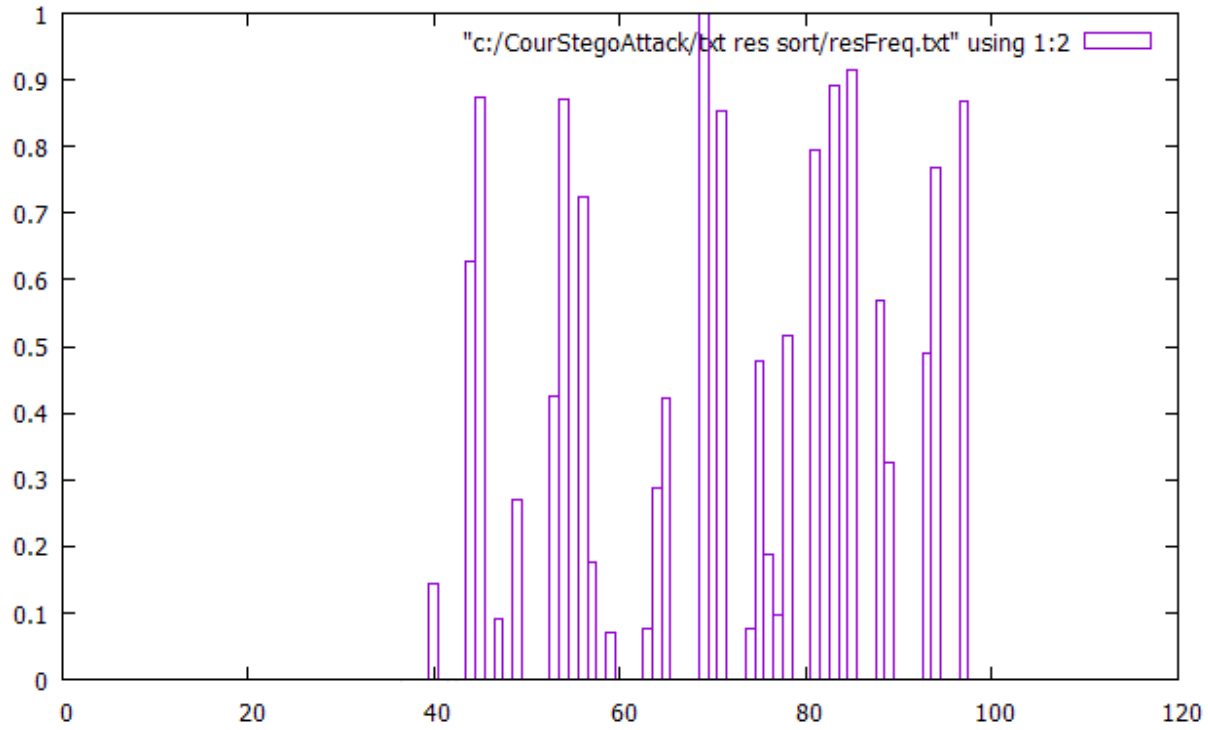


Рисунок 15 - Проверка на случайность последовательности LSB 108 «пустых» аудио файлов.

Как видно из рисунка 15, последовательности LSB у файлов с относительным количеством нулевых байт менее 0,08(значение для 40 файла) неслучайны.

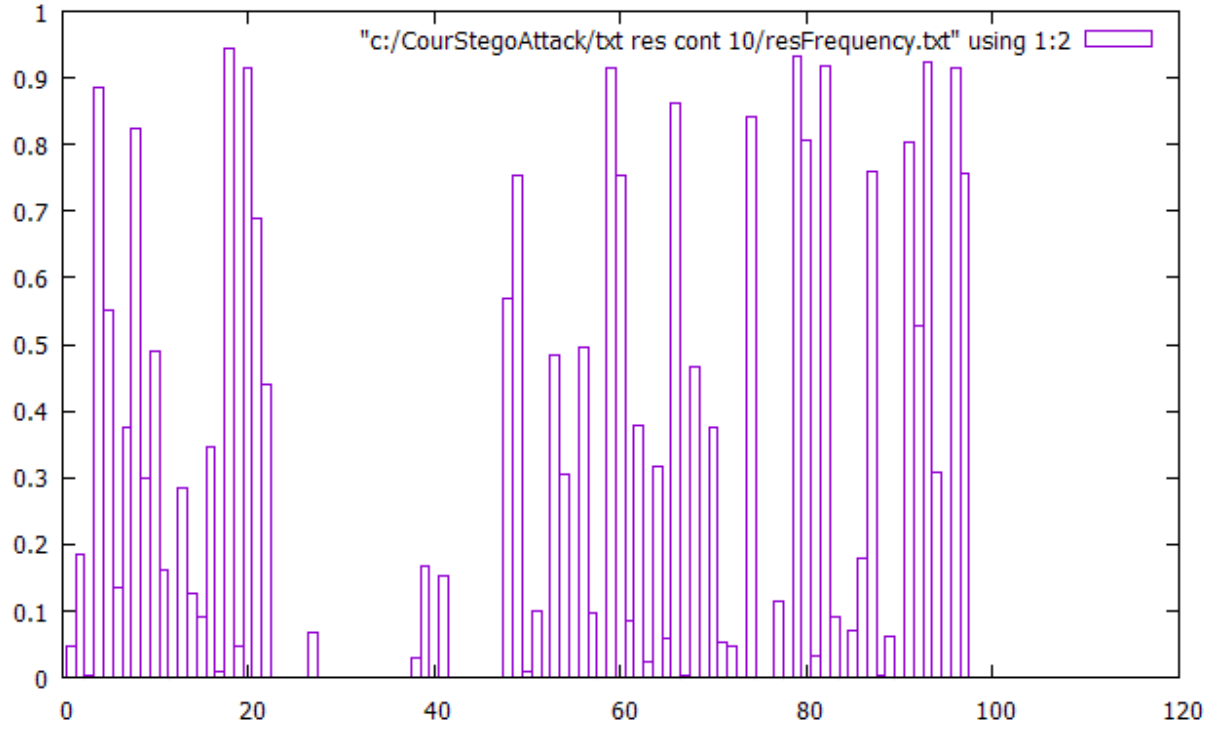


Рисунок 16 - Проверка на случайность последовательности LSB 108 аудио файлов, заполненных на 10% от максимальной возможности.

Из рисунка 16 видно, что при десяти процентном стеговложении последовательности LSB у файлов с относительным количеством нулевых байт меньше 0,0348 (значение для 22-го файла) случайны.

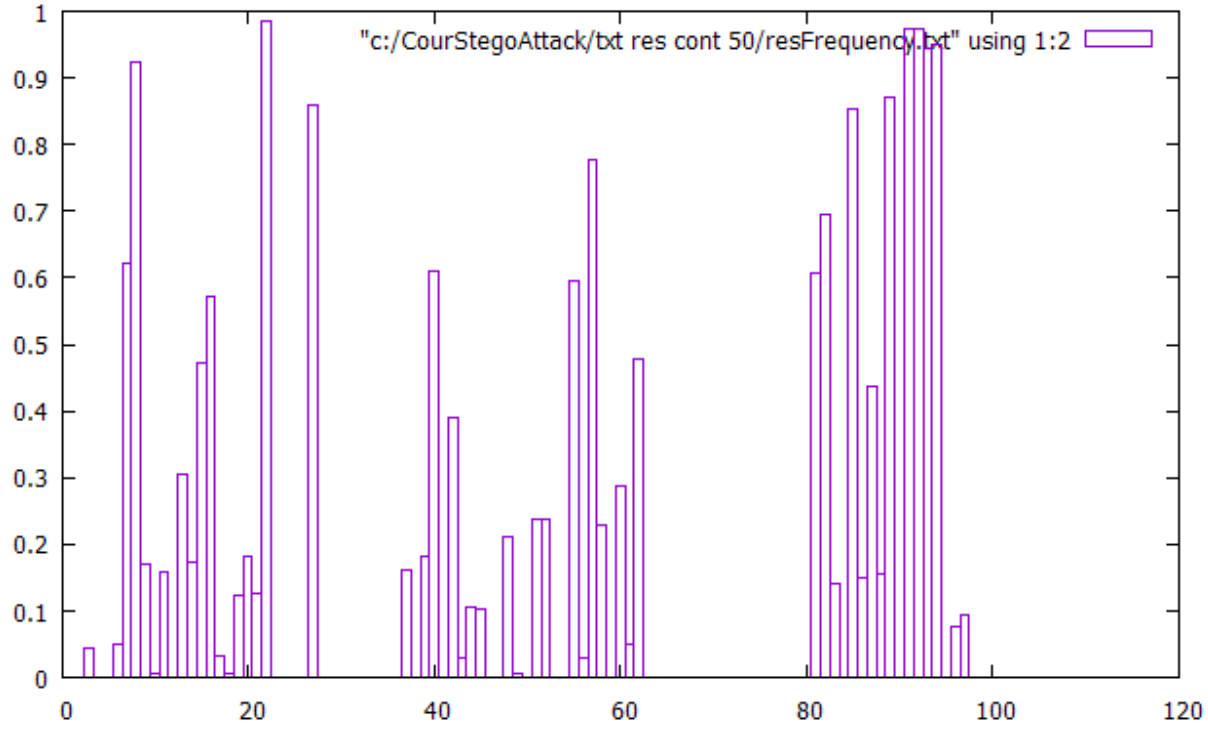


Рисунок 17 - Проверка на случайность последовательности LSB 108 аудио файлов, заполненных на 50% от максимальной возможности.

Из рисунка 17 видно, что при пятидесяти процентном стеговложении последовательности LSB у файлов с относительным количеством нулевых байт меньше 0,0348 (значение для 22-го файла) в большинстве своем случайны.

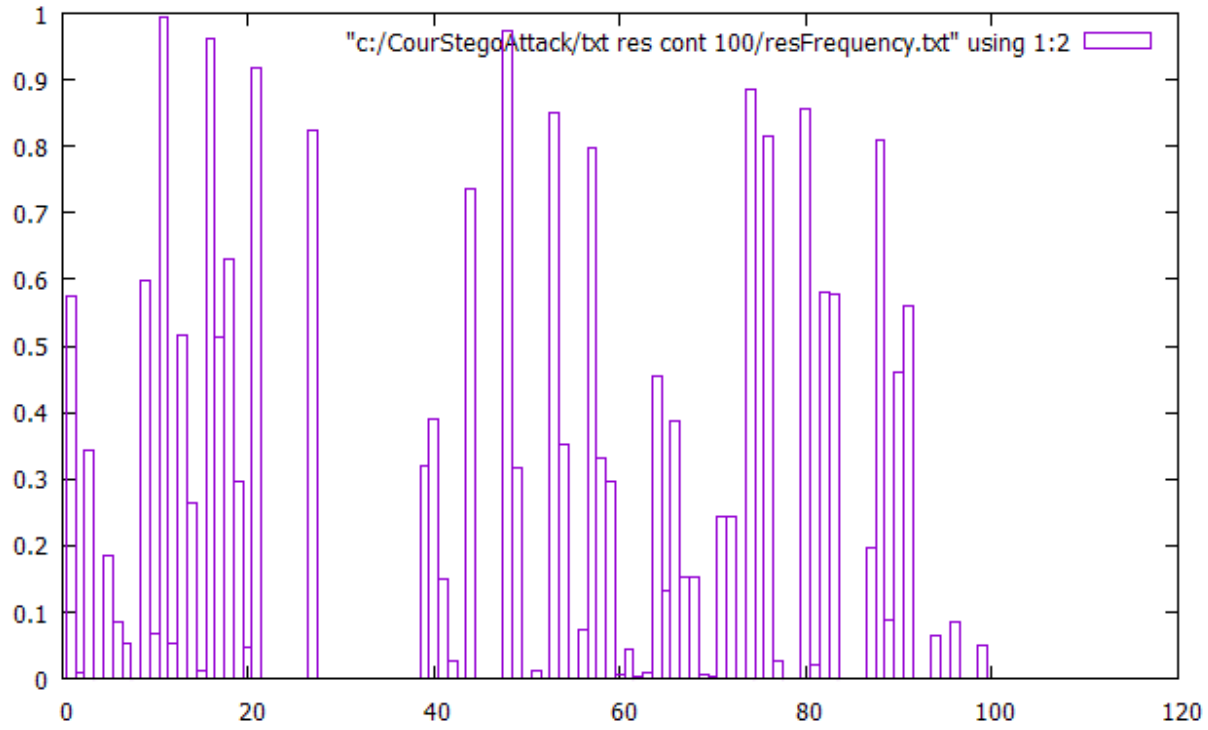


Рисунок 18 - Проверка на случайность последовательности LSB 108 аудио файлов, заполненных на 100% от максимальной возможности.

Из Рисунка 18 видно, что при ста процентном стеговложении последовательности LSB у файлов с относительным количеством нулевых байт меньше 0,0348 (значение для 22-го файла) случайны.

Практическое использование методики для определения вложений в аудио файл:

1. Берется проверяемый файл

2. Определяется относительное число нулевых байт

3. Если это число меньше, чем 0,0038 (подобрано эмпирически), то файл проверяется частотным тестом NIST.

Если тест показал, что последовательность младших битов файла является случайной (т.е. результат частотного теста больше, чем 0,01), то это с большой долей уверенности говорит о том, что в файле имеются стеговложения.

В программе данная методика была реализована следующим образом.

1. При нажатии на кнопку «Analyze» пользователю необходимо сначала выбрать файл для анализа, затем файл, в который сохранится последовательность LSB выбранного файла. В стандартном текстовом редакторе Memo1 появится относительное количество нулевых байт.

Откроется пакет статистических тестов NIST. В нем необходимо в выбрать файл с последовательностью младших бит анализируемого файла, формат чтения: текстовый, количество последовательностей: 1, длину последовательности выставить в соответствии с размером файла и нажать на кнопку «Тест».

2. При нажатии на кнопку «Show Results» открывается текстовый документ с результатами тестов с их названиями.

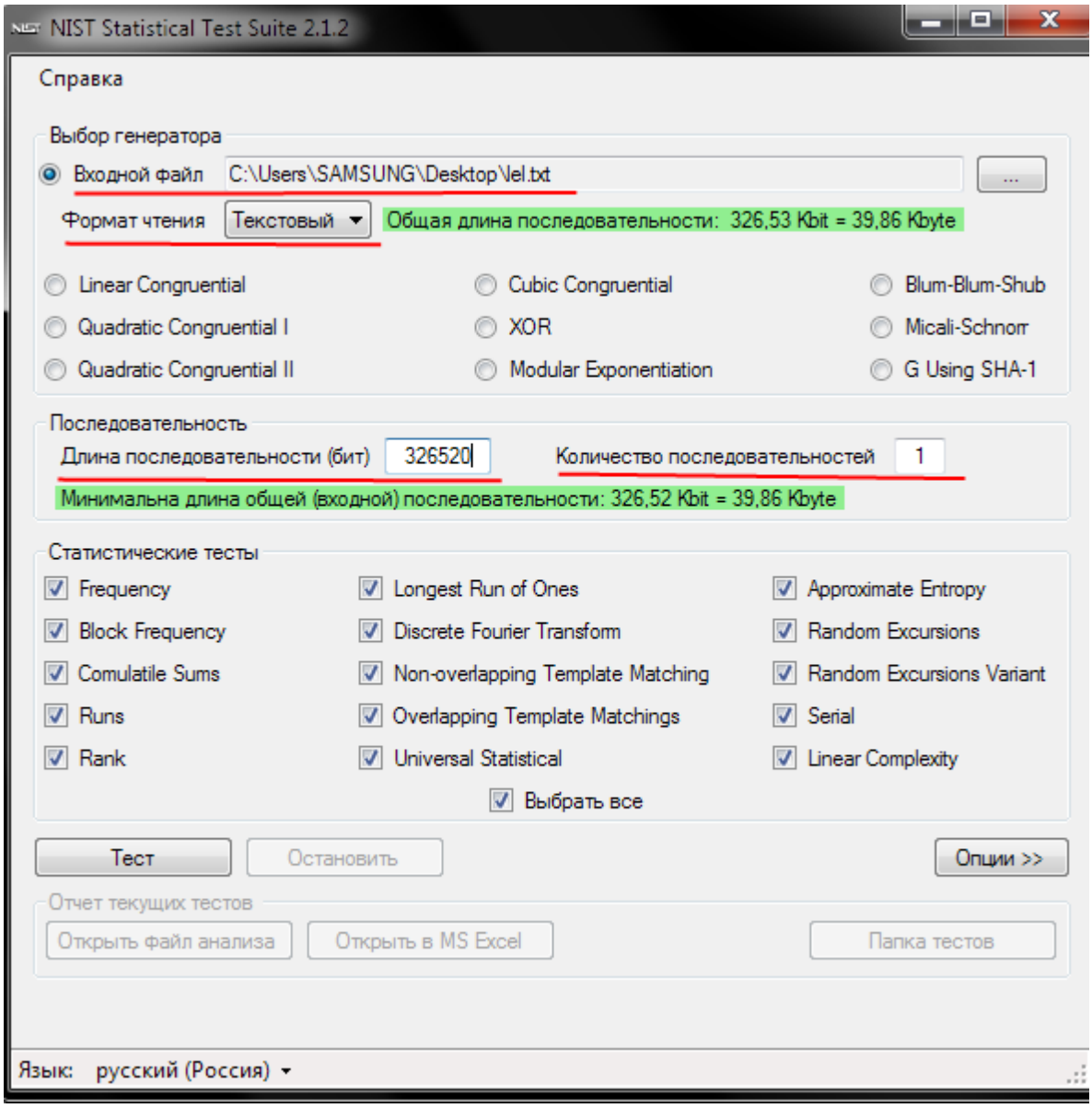


Рисунок 19 - Окно пакета статистических тестов NIST.

Общий вывод по тестированию. Установлено, что предложенная методика работает весьма эффективно (определяются даже 10% вложения), но только на аудиофайлах с малым относительным количеством нулевых байт.

6.2 Метод стегоанализа аудио файлов на основе алгоритмов сжатия.

Для проверки сравнительной эффективности предложенной методики стегоанализа в работе был реализован известный метод стегоанализа, основанный на алгоритме сжатия, описанном в [8].

6.2.1 Замечания и недостатки стегоанализа на основе алгоритма сжатия, изложенного в работе [8].

В работе [8]

- Не указано количество изменяемых младших бит.

- Нет никакой информации о виде аудио файлов, подвергавшихся исследованию.

- Вполне вероятно, что были использованы аудио файлы схожего вида, следовательно, данный алгоритм нельзя считать универсальным.

- Были использованы непопулярные в наше время 8- и 16-битные WAVE файлы. Сейчас более актуальны аудио файлы битностью от 24 и выше.

6.2.2 Реализация метода стегоанализа на основе алгоритма сжатия

При реализации ввиду сделанных выше замечаний, принято:

- виды исследуемых файлов максимально произвольные (представленные в базе);

- количество младших бит, заменяемых на случайные – 2 (как это было сделано в стеганографической части разработанного приложения);

- битность файлов от 8 до 32.

В разработанном программном обеспечении данный метод реализован следующим образом:

В интерфейсе разработанного пакета Stegora WaveHide при нажатии на кнопку «Alternate Analysis» и вводе коэффициентов sigma и K пользователю требуется выбрать файл для анализа. В начале файл разбивается на K равных частей. Затем производится сжатие каждой из частей при помощи методов из стандартной библиотеки zlib и высчитывается отношение размера сжатого куска файла к несжатому. После в изначальном файле два младших бита изменяются на случайные и процедура повторяется, высчитывается отношение размеров сжатого куска к несжатому. Для каждого куска рассчитывается параметр Δ. Если количество кусков, параметр Δ для которых меньше, чем введенный коэффициент sigma, больше половины, то

выносится решение о наличии в рассматриваемом файла стеговложений. В противном случае выносится решение об отсутствии таких вложений.

6.2.3 Тестирование базы файлов на предмет вложений с помощью метода, основанном на алгоритме сжатия

Тестовая база из 108 аудио файлов была проверена на стеговложения с помощью данного метода. На графиках ниже показаны значения параметра Δ. Сравнивались значения этого параметра для «пустых» и заполненных (с помощью разработанного ПО Stegora WaveHide) на 100% файлов. Результаты представлены на графиках ниже. По оси обсцисс на графиках находятся номера тестируемых файлов, по оси ординат – полученное значение параметра Δ.

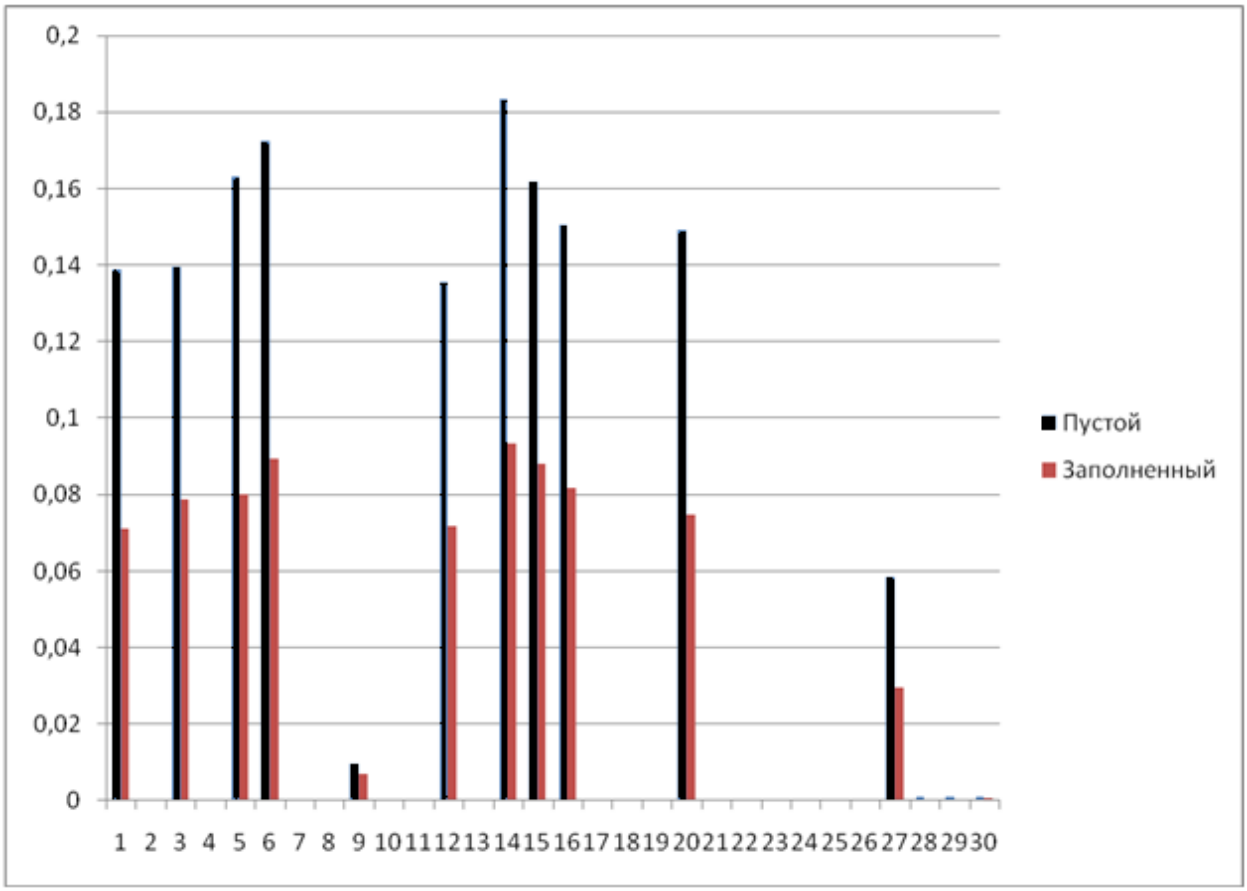


Рисунок 21 – Тестирование файлов с 1 по 30

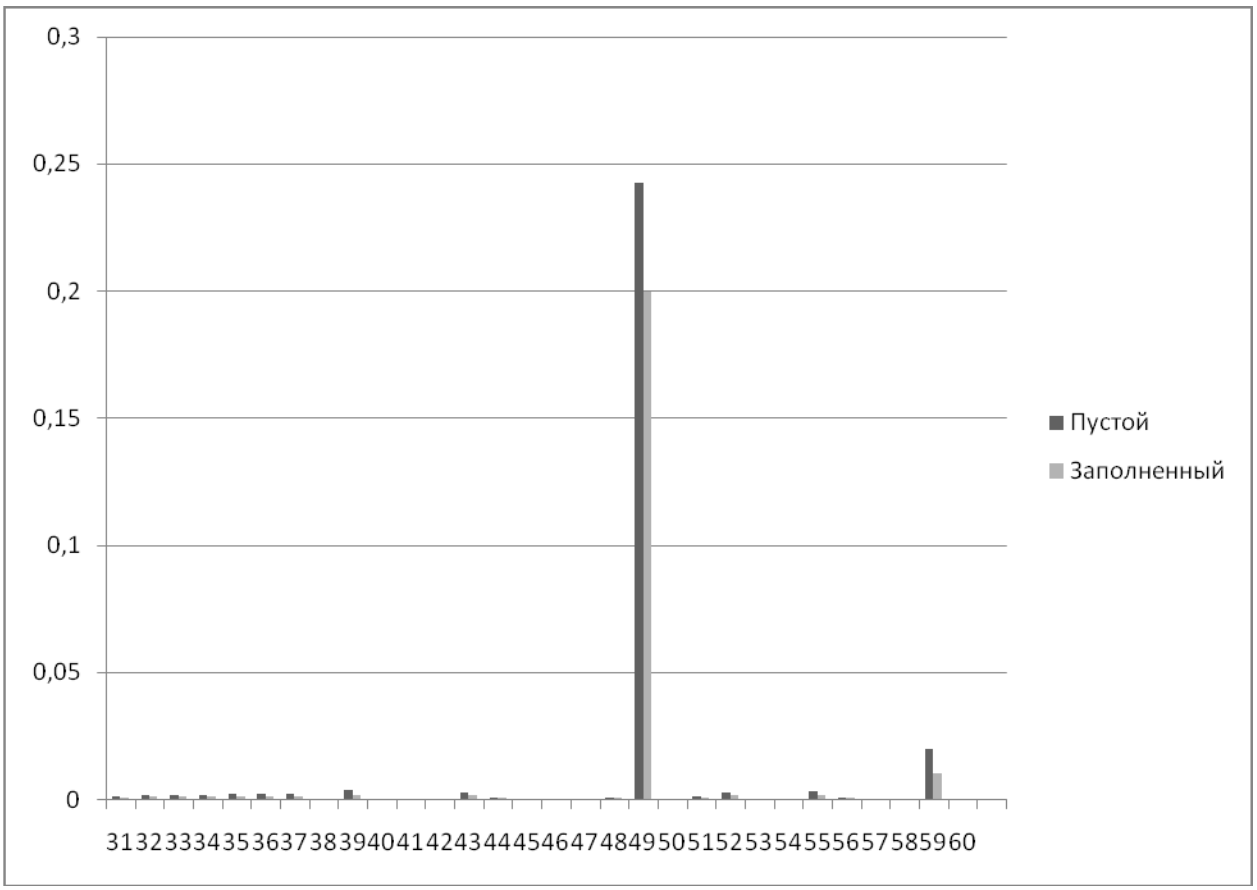


Рисунок 22 – Тестирование файлов с 31 по 60

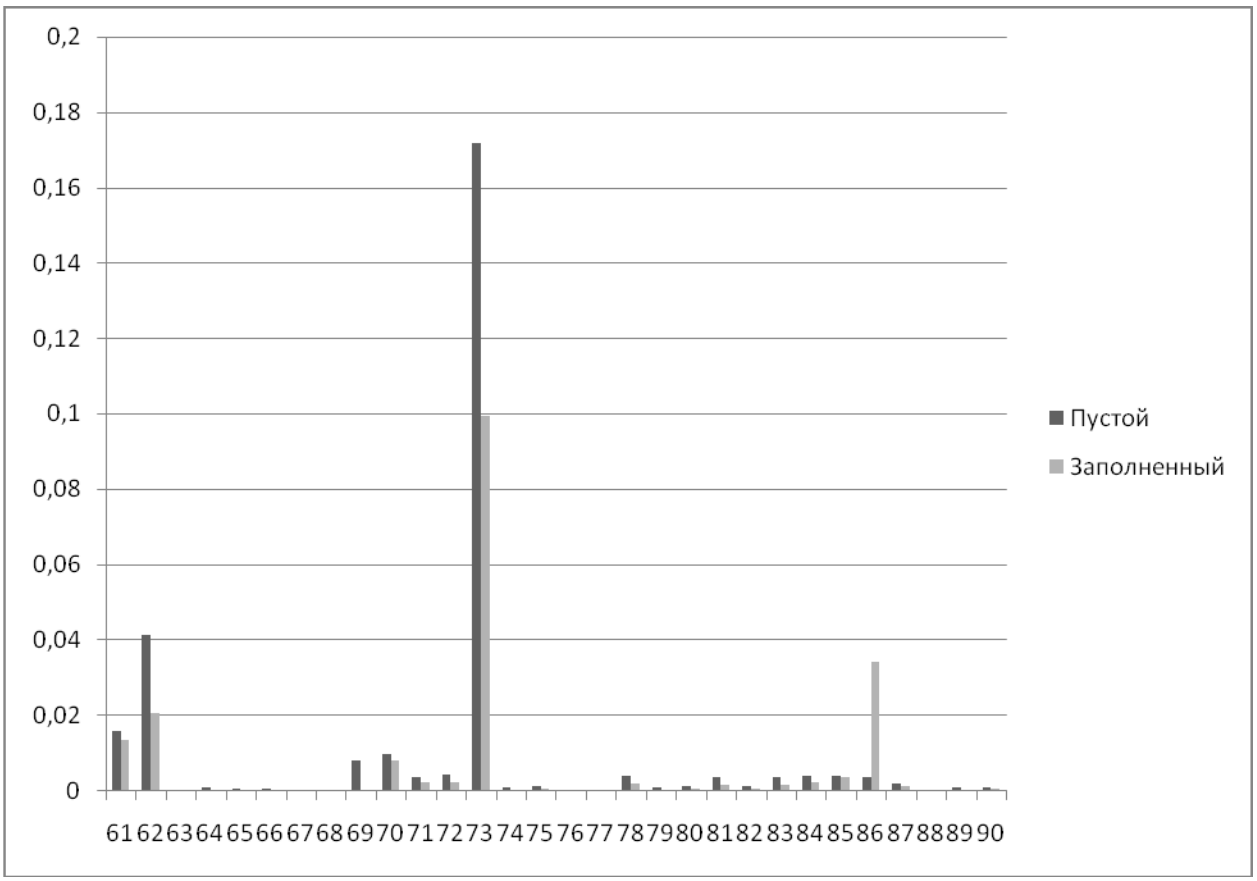


Рисунок 23 – Тестирование файлов с 61 по 90

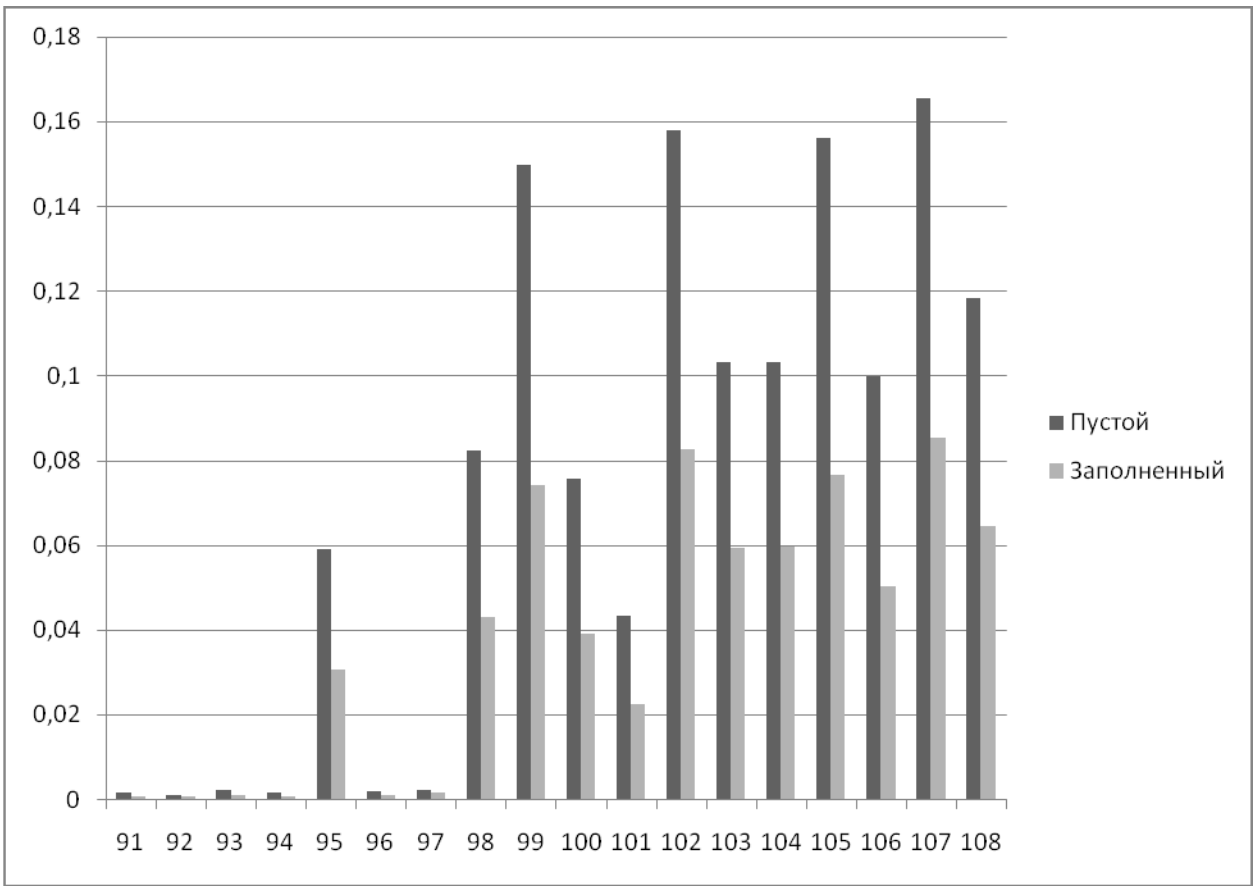


Рисунок 24 – Тестирование файлов с 91 по 108

Как видно из результатов тестирования разработанного ПО (см. рисунки 21-24), заполненный контейнер сжимается в среднем в два раза хуже. Следовательно, метод стегоанализа, основанный на алгоритмах сжатия можно использовать для определения наличия вложений в аудио файлах.

Но поскольку параметр Δ различен для разных видов файлов, для эффективного применения данного метода необходимо классифицировать файлы и для каждого класса установить свое пороговое значение Δ.

6.3 Сравнительный вывод по рассматриваемым методам стегоанализа.

Результаты тестирования базы из 108 различных аудио файлов формата WAVE показали, что метод, основанный на алгоритмах сжатия [8] требует дополнительной доработки в плане классификации аудио файлов. С большой долей уверенности можно говорить о возможности применения данного метода для определения стеговложений в аудио файлах, принадлежащих к одному классу.

Результаты тестирования этой же базы, но применяя метод, основанный на частотном анализе, показали следующее. Метод работает весьма эффективно для файлов, относительное количество нулевых байт для которых меньше, чем 0,038. Файлы, относящиеся к данному классу отличаются большой информационной нагруженностью. К этому классу относятся шумы, записи музыкальных инструментов с большим количеством шумов (большое количество шума возникает из-за использования некачественных АЦП).

Поэтому для эффективного обнаружения стеговложений необходима доработка обоих методов и, возможно их комплексное использование.

ЗАКЛЮЧЕНИЕ

В настоящей бакалаврской работе рассмотрена проблема стеганографии и стегоанализа на аудио файлах и получены следующие результаты:

Сделан обзор существующего программного обеспечения для сокрытия информации в WAVE файлах, который показал, что на данный момент в свободном доступе отсутствует ПО, удовлетворяющее требованиям к стеганографическому ПО на аудио файлах.

Разработано приложение (пакет Stegora WaveHide) для стеганографического сокрытия данных в аудио файлах формата WAVE с помощью метода LSB (Least Significant Bit, Наименьший Значащий Бит). Приложение поддерживает WAVE файлы любой битности, частоты дискретизации, определяет объем контейнера и реазует многотомность. Также реализован алгоритм извлечения скрытых данных. Шифрование данных перед сокрытием производится с помощью алгоритма XOR. Приложение отвечает всем требованиям, поставленным к стеганографическому программному обеспечению и может использоваться для сокрытия данных в аудио файлах формата WAVE.

Также в настоящей работе рассмотрена проблема стегоанализа в аудио файлах формата WAVE, предложена собственная методика для определения наличия стеганографических вложений в аудио файлах на основе статистических тестов последовательностей младших бит. Разработанная методика позволяет обнаруживать вложения, сделанные с помощью алгоритма LSB, в некоторые виды аудио файлов формата WAVE. В будущем планируется усовершенствовать данную методику и расширить ее на все виды аудио файлов.

Для сравнительного анализа в работе реализован алгоритм стегоанализа в аудио файлах на основе известного метода, основанного на алгоритме сжатия, описанного в статье [8].

Проведен сравнительный анализ этих двух подходов к обнаружению вложений. Обнаружены их относительные достоинства и недостатки .

Дальнейшее развитие данной работы заключается в усовершенствовании разработанного программного обеспечения для стеганографии в аудио файлах формата WAVE, разработке универсального метода для определения вложений в аудио файлы. Также планируется рассмотреть данную проблему на видео файлах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Генне, О. В. Основы стегоанализа // Защита Информации. – 2000. – №3. – С. 57-58.

2. Гребенников В.В., История Криптологии & Секретной Связи [Электронный Ресурс] : глава 6.1, вступление в историю стеганографии, 2012. Режим доступа: http://cryptohistory.ru/book/chast-6-istoriya-steganografii/61-vstuplenie/

3. Барсуков В.С., Романцов А.П. Несколько слов о стеганографии// Специальная Техника. – 1998. – № 4. – С. 25-26.

4. Энциклопедический Фонд России [Электронный Ресурс] : Стеганография. – 2004. – Москва. – Режим доступа: http://www.russika.ru/termin.asp?ter=3747

5. Web Archive [Электронный ресурс] : Abdul Hameed Bakier, Technical Mujahid, a Training Manual for Jihadis. – 2010. – Нью-Йорк. – Режим доступа – http://web.archive.org/web/20101013075447/www.jamestown.org/programs/gta/single/?tx_ttnews%5Btt_news%5D=1057&tx_ttnews%5BbackPid%5D=182&no_cache=1

6. Быков, С. Ф., Мотуз, О. В. Основы стегоанализа // Защита информации. – 2000. – №3. – С. 38-41.

7. Стегоанализ графических данных в различных форматах [Электронный ресурс] : Жилкин М.Ю. – 2008. – Новосибирск. – Режим доступа: http://www.sbras.ru/ws/YM2007/12817/paper.html

8. С. Ю. Очимов, Стегоанализ аудиофайлов, базирующийся на алгоритмах сжатия // Вестник СибГУТИ. – 2010. – №1 . – С. 33-37.

9.Wave file format – формат звукового файла WAVE [Электронный ресурс] : microsin. – 2011. – Москва. – Режим доступа: http://microsin.net/programming/pc/wav-format.html

10. Структура WAVE файла [Электронный ресурс] : Audio Coding. – 2008. – Москва. – Режим доступа – http://audiocoding.ru/article/2008/05/22/wav-file-structure.html

11. Импульсно-кодовая модуляция [Электронный ресурс] : Википедия. – 2016. – Режим доступа: https://ru.wikipedia.org/wiki/Импульсно-кодовая_модуляция

12. Полный список аудио форматов [Электронный ресурс] : Audio Coding. – 2008. – Москва. – Режим доступа: http://audiocoding.ru/assets/meta/2008-05-22-wav-file-structure/wav_formats.txt

13. Гурский, Д. И. ActionScript 2: программирование во Flash MX / Д. И. Гурский. – Санкт-Петербург : Питер, 2004. – 860 с.

14. Е.Л. Зорин, Н.В. Чичиварин. Стеганография в САПР : учебное пособие. – Москва : МГТУ им. Н.Э. Баумана, 2013. – 90 с.

15. Забелин, М. А. Стегоанализ аудиоданных на основе методов сжатия // Вестник СибГУТИ. – 2010. – №1 . – С. 41-46.

16. DeepSound. [Электронный ресурс] : Описание ПО и ссылка на скачивание. – 2015. – Словения. – Режим доступа: http://jpinsoft.net/DeepSound/Overview.aspx

17. Xiao Steganography. [Электронный ресурс] : Описание ПО и ссылка на скачивание. – 2012. – Пекин. – Режим доступа – http://download.cnet.com/Xiao-Steganography/3000-2092_4-10541494.html

18. SilentEye. [Электронный ресурс] : Описание ПО и ссылка на скачивание. – 2008. – Пекин. – Режим доступа: http://silenteye.v1kings.io/index.html?i1s1

19. StegoStick beta. [Электронный ресурс] : Описание ПО и ссылка на скачивание. – 2013. – Сараево. – Режим доступа: https://sourceforge.net/projects/stegostick/

20. Кокорин П. П. О методах стегоанализа в аудиофайлах // Труды СПИИРАН. – 2007. – №4. – С. 239-246.

21. Ben-4D. [Электронный ресурс] : Описание ПО и ссылка на скачивание. – 2010. – Хельсинки. – Режим доступа: https://sourceforge.net/projects/ben4dstegdetect/

22. Digital Invisible Ink Toolkit. [Электронный ресурс] : Описание ПО и ссылка на скачивание. – 2008. – Новая Зеландия. – Режим доступа: https://sourceforge.net/projects/diit/?source=typ_redirect

23.Алгоритм XOR [Электронный ресурс] : Википедия. – 2016. Режим доступа: https://en.wikipedia.org/wiki/XOR_swap_algorithm

24. Raggo M.T. Data Hiding: Exposing Concealed Data in Multimedia, Operating Systems, Mobile Devices and Network Protocols. M.T. Raggo. – Массачусетс : Syngress, 2012. – 270 с.

25. Статистическая проверка случайности двоичных последовательностей методами NIST [Электронный ресурс]: Код Безопасности. – Москва. – Режим доступа: https://habrahabr.ru/company/securitycode/blog/237695/

26. Пакет статистических тестов NIST [Электронный ресурс]: Описание пакета и ссылка на скачивание. – 2014. – Гетейсберг. – Режим доступа: http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html

ПРИЛОЖЕНИЕ А

Исходный код разработанного программного обеспечения.

```
unit WaveHideDiploma;

interface

uses

  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,

  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Buttons, Vcl.StdCtrls, Vcl.ComCtrls, zlib, ShellApi;

type

  bin = array[0..7] of 0..1;

  TypeSize = array[0..3] of byte;

  TForm5 = class(TForm)

    Button1: TButton;

    Button2: TButton;

    BitBtn1: TBitBtn;

    OpenDialog1: TOpenDialog;

    Button3: TButton;

    SaveDialog1: TSaveDialog;

    Label1: TLabel;

    Memo1: TMemo;

    OpenDialog2: TOpenDialog;

    Button4: TButton;

    Button5: TButton;

    Button6: TButton;

    Edit1: TEdit;

    Edit2: TEdit;

    Label2: TLabel;

    Label3: TLabel;

    SaveDialog2: TSaveDialog;
```

```
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    Procedure HideWave(finn, foutn, fstn: string);
    procedure Button2Click(Sender: TObject);
    procedure WaveUnHide(foutn, ffinn: string);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form5: TForm5;
  Container, Ideal: string;
  AnalyzeFileName: string;
implementation

{$R *.dfm}

function FromBytesToCardinal(by: TypeSize): cardinal; inline;
begin
  result := By[0] + (by[1] shl 8) + (by[2] shl 16) + (by[3] shl 24);
end;

procedure FromCardinalToBytes(const AInput: cardinal; out by: TypeSize); inline;
```

```
begin
  by[0] := byte(AInput);
  by[1] := byte(AInput shr 8);
  by[2] := byte(AInput shr 16);
  by[3] := byte(AInput shr 24);
end;

function bytetobin(b: byte): bin;
var
  i: integer;
  m: bin;
begin
  for i := 7 downto 0 do
   begin
     m[i]:= b mod 2;
     b:= b div 2;
   end;
  result:= m;
end;

function bintobyte(b: bin): byte;
var
  ii, jj, pow: integer;
begin
  for ii := 0 to 7 do
    begin
      pow:= 1;
      for jj:= 1 to 7-ii do
        pow:= pow*2;
      result:= pow * b[ii] + result;
```

```
    end;
end;
//Реализация Алгоритма XOR

procedure first(fin, fout, fsteg: string);
var
  f_in: file of byte;
  f_out: file of byte;
  f_steg: file of byte;
  buf, i: byte;
  d, df, ds: bin;
begin
  assign(f_in, fin);
  assign(f_out, fout);
  assign(f_steg, fsteg);

  reset(f_in);
  rewrite(f_out);
  reset(f_steg);



  while not eof(f_steg) do
  begin


    if eof(f_in) then
      reset(f_in);////

    blockread(f_in, buf, 1);
```

```
    d:= bytetobin(buf);
    blockread(f_steg, buf, 1);
    df:= bytetobin(buf);
    for i := 0 to 7 do
      begin
        if d[i] = df[i] then
          ds[i]:= 1
        else
          ds[i]:= 0;
      end;
    buf:= bintobyte(ds) + 1;

    blockwrite(f_out, buf, 1);
  end;


  closefile(f_in);
  closefile(f_out);
  closefile(f_steg);


end;

procedure last(fin, fout, fend: string);
var
  f_in: file of byte;
  f_out: file of byte;
  f_end: file of byte;
  buf, i: byte;
  d, df, ds: bin;
```

```
begin
  assign(f_in, fin);
  assign(f_out, fout);
  assign(f_end, fend);
  reset(f_in);
  reset(f_out);
  rewrite(f_end);



  while not eof(f_out) do
  begin
    if eof(f_in) then
      reset(f_in);
    blockread(f_in, buf, 1);
    d:= bytetobin(buf);
    blockread(f_out, buf, 1);
    df:= bytetobin(buf);

    for i := 0 to 7 do
      begin
        if df[i] = 1 then
          ds[i]:= d[i];
        if df[i] = 0 then
          ds[i]:= 1 - d[i];
      end;
    buf:= bintobyte(ds) + 1;
    blockwrite(f_end, buf, 1);
  end;
```

```
  closefile(f_in);
  closefile(f_out);
  closefile(f_end);



end;

procedure TForm5.WaveUnHide(foutn, ffinn: string);
var
  F_out: file;
  F_fin: file;
  i, j, a, ia: integer;
  d, df: bin;
  buf: byte;
  col, consize: cardinal;
  by: typesize;
  bps: integer; //Bytes per sample

begin
  assignfile(F_out, foutn);
  assignfile(F_fin, ffinn);
  reset(F_out, 1);
  rewrite(F_fin, 1);
  consize:= filesize(f_out);
  for i := 0 to 33 do
    begin
      Blockread(F_out, buf, 1);
      dec(consize);
```

```
 end;

blockread(F_out, buf, 1);
dec(consize);
BPS:= buf div 8;
if foutn[length(foutn)] = 'p' then
  bps:= 1;
memo1.lines.Add(inttostr(bps));

for i := 35 to 104 - bps  do
 begin
   Blockread(F_out, buf, 1);
   dec(consize);
 end;

a:= 0;
i:= 0;

while i < 4 do

 begin
   j:= bps;                    //    Участок отвеачающий
     while j > 1 do              //    за работу с
     begin                       //    различными
       Blockread(F_out, buf, 1);     //    bps wav
       dec(consize);
       dec(j);                   //
     end;

   blockread(f_out, buf, 1);
```

```
    dec(consize);
    d:= bytetobin(buf);
    df[a]:= d[6];
    inc(a);
    df[a]:= d[7];
    inc(a);
    if a = 8 then
      begin
        a:= 0;
        buf:= bintobyte(df) + 1;//////
        by[i]:= buf;
        inc(i);
      end;
  end;

col:= frombytestocardinal(by);
ia:= 0;

while ia < col do
  begin
    if consize <= (bps*4) + 1 then
    begin
      memo1.Lines.Add('consize unhide =' + inttostr(consize));
      closefile(f_out);
      showmessage('Enter the next carrier');
      if OpenDialog1.Execute then
        foutn:= OpenDialog1.FileName;
      memo1.Lines.Add(foutn);
      assignfile(f_out, foutn);
      reset(f_out, 1);
```

```
      consize:= filesize(f_out);

      for i := 0 to 33 do
       begin
         Blockread(F_out, buf, 1);
         dec(consize);
      end;

       blockread(F_out, buf, 1);
       dec(consize);
       BPS:= buf div 8;
       if foutn[length(foutn)] = 'p' then
         bps:= 1;
       memo1.lines.Add(inttostr(bps));

      for i := 35 to 104 - bps  do
      begin
       Blockread(F_out, buf, 1);
       dec(consize);
      end;

     end;



    a:=0;
    while a < 8 do
begin
    ////////////////////////////////
    j:= bps;                    //  Участок отвеачающий
      while j > 1 do            //    за работу с
```

```
      begin                    //   различными
        Blockread(F_out, buf, 1);   //   bps wav
        dec(consize);
        dec(j);                  //
      end;
    ////////////////////////////////


    blockread(f_out, buf, 1);
    dec(consize);
    d:= bytetobin(buf);
    df[a]:= d[6];
    inc(a);
    df[a]:= d[7];
    inc(a);

end;
       a:= 0;
       buf:= bintobyte(df) + 1;
       inc(ia);

       blockwrite(f_fin, buf, 1);
  end;

  closefile(F_out);
  closefile(F_fin);
  last(Ideal, ffinn, ffinn + 'end');
  deletefile(ffinn);

end;
```

```
procedure TForm5.HideWave(finn, foutn, fstn: string);
var
  f: textfile;
  col, consize: cardinal;
  d, df: bin;
  Buf, a: byte;
  F_in: File;
  buf1: array[0..2047] of byte;
  F_out: file;
  F_steg: File;
  i, j, it: integer;
  by: typesize;
  BPS: integer; //Bytes per Sample

begin
  assignfile(F_in, finn);
  assignfile(F_out, foutn);
  first(Ideal, 'C:\Users\Public\mid.comp', fstn);
  fstn:= 'C:\Users\Public\mid.comp';

  assignfile(F_steg, fstn);     ////fstn поставить блять!!!
  assignfile(f, 'c:\CourMyProj\tata.txt');
  rewrite(f);
  reset(F_in, 1);
  reset(F_steg, 1);
  rewrite(F_out, 1);
  col:= filesize(f_steg);
  memo1.Lines.Add(inttostr(col));
  FromCardinalToBytes(col, by);
```

```
consize:= filesize(f_in);

for i := 0 to 33 do                ///
 begin
   Blockread(F_in, buf, 1);
   dec(consize); /////DECCONSIZE
   Blockwrite(F_out, buf, 1);
 end;

 blockread(F_in, buf, 1);
 dec(consize); /////DECCONSIZE
 BPS:= buf div 8;
 if finn[length(finn)] = 'p' then
   bps:= 1;

 col:= filesize(f_steg);
 memo1.lines.Add(inttostr(bps));
 Blockwrite(F_out, buf, 1);

 for i := 35 to 104 - bps  do
  begin
   Blockread(F_in, buf, 1);
   dec(consize); /////DECCONSIZE
   Blockwrite(F_out, buf, 1);
  end;

//В контейнер записывается размер скрываемого файла

for i := 0 to 3 do
begin
```

```
  d:= bytetobin(by[i]);
  a:= 0;
  while a < 8 do
    begin
      j:= bps;                    //    Участок отвеачающий
      while j > 1 do              //    за работу с
      begin                       //    различными
        Blockread(F_in, buf, 1);     //    bps wav
        dec(consize);               //
        Blockwrite(F_out, buf, 1);   //
        dec(j);                     //
      end;
      blockread(F_in, buf, 1);
      dec(consize);
      df:= bytetobin(buf);
      writeln(f, buf);
      df[6]:= d[a];
      inc(a);
      df[7]:= d[a];
      inc(a);
      buf:= bintobyte(df);
      blockwrite(F_out, buf, 1);
    end;
end;

while col > 0 do

begin
  blockread(F_steg, buf, 1);
  d:= bytetobin(buf);
```

```
a:= 0;
dec(col);

///Многотомность

if consize <= (bps*4) + 1 then
begin
  memo1.Lines.Add('BPS= ' + inttostr(BPS));
  memo1.Lines.Add('consize = ' + inttostr(consize));

  while not eof(f_in) do
  begin
    blockread(f_in, buf, 1);
    blockwrite(f_out, buf, 1);
  end;

  closefile(f_in);
  closefile(f_out);

  showmessage('Not enough space. Choose next carrier.');
  if opendialog1.Execute then
    finn:= OpenDialog1.FileName;
  assignfile(f_in, finn);
  showmessage('Choose the name of new encrypted file.');
  if savedialog1.Execute then
    foutn:= SaveDialog1.FileName;

  assignfile(f_out, foutn);
  reset(f_in, 1);
  rewrite(f_out, 1);
```

```
  consize:= filesize(f_in);

  for i := 0 to 33 do
  begin
  Blockread(F_in, buf, 1);
  dec(consize);
  Blockwrite(F_out, buf, 1);
 end;

blockread(F_in, buf, 1);
dec(consize);

BPS:= buf div 8;
if finn[length(finn)] = 'p' then
  bps:= 1;
memo1.lines.Add(inttostr(bps));
Blockwrite(F_out, buf, 1);

for i := 35 to 104 - bps  do
 begin
  Blockread(F_in, buf, 1);
  dec(consize);
  Blockwrite(F_out, buf, 1);
 end;

 end;

 while a < 8 do
  begin
    j:= bps;                    //    Участок отвеачающий
```

```
    while (j > 1)  do            //    за работу с
    begin                        //    различными
      Blockread(F_in, buf, 1);   //    bps wav
      dec(consize);              //
      Blockwrite(F_out, buf, 1); //
      dec(j);                    //
    end;

    blockread(F_in, buf, 1);
    dec(consize);
    df:= bytetobin(buf);
    writeln(f, buf);
    df[6]:= d[a];
    inc(a);
    df[7]:= d[a];
    inc(a);
    buf:= bintobyte(df);
    blockwrite(F_out, buf, 1);
  end;
end;

it:= (filesize(f_in) - filepos(f_in)) div 2048;
for i := 1 to it do
  begin
    blockread(f_in, buf1, 2048);
    blockwrite(f_out, buf1, 2048);
  end;

while not eof(f_in) do
  begin
```

```
      blockread(f_in, buf, 1);
      blockwrite(f_out, buf, 1);
    end;

  closefile(F_in);
  closefile(F_out);
  closefile(F_steg);
  showmessage('File is successfully hidden.');
end;

//UnHide
procedure TForm5.Button2Click(Sender: TObject);
var
  f1, f2: string;
begin
  if OpenDialog1.Execute   then
    f1:= OpenDialog1.FileName;
  if SaveDialog2.Execute then
    f2:= SaveDialog2.FileName;
  WaveUnHide(f1, f2);
end;

//Choose a carrier
procedure TForm5.Button3Click(Sender: TObject);
var
  f: file of byte;
  i: integer;
  bps: integer;
  buf: byte;
```

```
begin
  if OpenDialog1.Execute   then
      container:= OpenDialog1.FileName;
  assignfile(f, container);
  reset(f);

  for i := 0 to 33 do                ///
    begin
      Blockread(f, buf, 1);
    end;

   blockread(f, buf, 1);
   BPS:= buf div 8;
  memo1.Lines.Add('Container size: ' + inttostr((-100 - 4*bps + (filesize(f) div (bps*4))) div
1024) + 'KB');
  closefile(f);
end;



///////СТЕГОАНАЛИЗ!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!////////////////////////////////

procedure TForm5.Button4Click(Sender: TObject);
  var
    ftex: string;

function empty(fin: string): double;
var
  f_n: file of byte;
  col, cole: int64;
  buf, i: byte;
  d, d1, d2, b, b1, b2: bin;
```

```
  bps, j: integer;
begin
  col:= 0;
  cole:= 0;
  assignfile(f_n, fin);
  reset(f_n);
  for i := 0 to 33 do                ///
    begin
      Blockread(F_n, buf, 1);
    end;

   blockread(F_n, buf, 1);
   BPS:= buf div 8;

   for i := 35 to 102 - bps  do              ///
    begin
      Blockread(F_n, buf, 1);
    end;

  while not eof(f_n) do
  begin
    Blockread(F_n, buf, 1);
    if buf = 0 then
    inc(cole);
    inc(col);
  end;

  result:= cole / col;
  closefile(f_n);
end;
```

```
procedure sumoflsb(fin: string; fout: string);
var
  f_in: file of byte;
  f_out: textfile;
  buf, i: byte;
  d, df: bin;
  col, bps, j: integer;
begin
  col:= 0;
  assignfile(f_in, fin);
  reset(f_in);

  assignfile(f_out, fout);
  rewrite(f_out);

  for i := 0 to 33 do                ///
    begin
      Blockread(F_in, buf, 1);
    end;

   blockread(F_in, buf, 1);
   BPS:= buf div 8;

   for i := 35 to 104 - bps  do               ///
    begin
      Blockread(F_in, buf, 1);
    end;

  while not eof(f_in) do
```

```
  begin
    j:= bps;                    //    Участок отвеачающий
       while not eof(f_in) and (j > 1) do              //    за работу с
       begin                        //    различными
         Blockread(F_in, buf, 1);     //    bps wav
         dec(j);                    //
       end;

      if eof(f_in) then
        exit;

      blockread(f_in, buf, 1);
      d:= bytetobin(buf);
      write(f_out, d[6]);
      write(f_out, d[7]);
  end;
  closefile(f_in);
  closefile(f_out);
end;

begin

  //ftex:= 'C:\Текстовые документы и все, что с ними связано\ИМ\!!Диплом\Stegora Wave
Hide\results\res.txt';
  if opendialog1.Execute then
       AnalyzeFileName := OpenDialog1.FileName;
  if savedialog2.Execute then
    ftex:= savedialog2.FileName;

  memo1.Lines.Add(AnalyzeFileName);
  memo1.Lines.Add(floattostr(empty(AnalyzeFileName)));
```

```
  sumoflsb(AnalyzeFileName, ftex);

  shellexecute(Handle, 'open', 'C:\Текстовые документы и все, что с ними
связано\ИМ\!!Диплом\!!Стегоанализ\NIST-statistical-test-suite-UI-
2.1.2\Release\NIST_UI.exe', nil, nil, SW_SHOWNORMAL);

end;


procedure TForm5.Button5Click(Sender: TObject);

  var

    f1: textfile;

    f2: textfile;

    c: double;

begin

   assignfile(f2, 'C:\Текстовые документы и все, что с ними связано\ИМ\!!Диплом\Stegora
Wave Hide\results\results.txt');

   assignfile(f1, 'C:\Текстовые документы и все, что с ними связано\ИМ\!!Диплом\Stegora
Wave Hide\Win32\Debug\experiments\AlgorithmTesting\ApproximateEntropy\results.txt');

   reset(f1);

   read(f1, c);

   closefile(f1);


   rewrite(f2);

   writeln(f2, AnalyzeFileName);

   writeln(f2,'Approx: ',  c);


   assignfile(f1, 'C:\Текстовые документы и все, что с ними связано\ИМ\!!Диплом\Stegora
Wave Hide\Win32\Debug\experiments\AlgorithmTesting\BlockFrequency\results.txt');

   reset(f1);

   read(f1, c);

   closefile(f1);

   writeln(f2,'Block: ',  c);
```

```
  assignfile(f1, 'C:\Текстовые документы и все, что с ними связано\ИМ\!!Диплом\Stegora Wave Hide\Win32\Debug\experiments\AlgorithmTesting\CumulativeSums\results.txt');
  reset(f1);
  read(f1, c);
  closefile(f1);
  writeln(f2,'CumulativeSums: ',  c);


  assignfile(f1, 'C:\Текстовые документы и все, что с ними связано\ИМ\!!Диплом\Stegora Wave Hide\Win32\Debug\experiments\AlgorithmTesting\FFT\results.txt');
  reset(f1);
  read(f1, c);
  closefile(f1);
  writeln(f2,'FFT: ',  c);

  assignfile(f1, 'C:\Текстовые документы и все, что с ними связано\ИМ\!!Диплом\Stegora Wave Hide\Win32\Debug\experiments\AlgorithmTesting\Frequency\results.txt');
  reset(f1);
  read(f1, c);
  closefile(f1);
  writeln(f2,'Frequency: ',  c);

  assignfile(f1, 'C:\Текстовые документы и все, что с ними связано\ИМ\!!Диплом\Stegora Wave Hide\Win32\Debug\experiments\AlgorithmTesting\LongestRun\results.txt');
  reset(f1);
  read(f1, c);
  closefile(f1);
  writeln(f2,'LongestRun: ',  c);

  assignfile(f1, 'C:\Текстовые документы и все, что с ними связано\ИМ\!!Диплом\Stegora Wave Hide\Win32\Debug\experiments\AlgorithmTesting\Rank\results.txt');
```

```
  reset(f1);

  read(f1, c);

  closefile(f1);

  writeln(f2,'Rank: ',  c);


  assignfile(f1, 'C:\Текстовые документы и все, что с ними связано\ИМ\!!Диплом\Stegora Wave Hide\Win32\Debug\experiments\AlgorithmTesting\Runs\results.txt');

  reset(f1);

  read(f1, c);

  closefile(f1);

  writeln(f2,'Runs: ',  c);


  assignfile(f1, 'C:\Текстовые документы и все, что с ними связано\ИМ\!!Диплом\Stegora Wave Hide\Win32\Debug\experiments\AlgorithmTesting\Serial\results.txt');

  reset(f1);

  read(f1, c);

  closefile(f1);

  writeln(f2,'Serial: ',  c);

  closefile(f2);



////////////////



  shellexecute(Handle, 'open', 'C:\windows\notepad.exe', 'C:\Текстовые документы и все, что с ними связано\ИМ\!!Диплом\Stegora Wave Hide\results\results.txt', nil, SW_SHOWNORMAL);


end;


//////ОЧИМОВ!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!//////////////
```

```
procedure TForm5.Button6Click(Sender: TObject);
var
  fn: string;
  f, f1, f2: file of byte;
  k, r, ii, iii, j: integer;
  b0:  double;
  s: byte;
  fx, fy: array of double;
  reresult: integer;
  sig: double;

  procedure CompressFile(const Source, Dest : String);
var
   SourceFile, DestFile : TFileStream;
   compr : TCompressionStream;
   bytecount : Integer;
begin
   SourceFile := TFileStream.Create(Source, fmOpenRead);
   DestFile := TFileStream.Create(Dest,fmCreate);
   try
     bytecount := SourceFile.Size;
     //store the original size in bytes
     DestFile.Write(bytecount, SizeOf(bytecount));
     //create the compression stream (after storing the bytecount!)
     compr := TCompressionStream.Create(clMax, DestFile);
     try
       //compress the content of the file
       compr.CopyFrom(SourceFile,bytecount);
     finally
       compr.Free;
```

```
    end;
   finally
    SourceFile.Free;
    DestFile.Free;
   end;
end;

  procedure HideWaveRandom(finn, foutn: string);
var
  f: textfile;
  col, consize: cardinal;
  d, df: bin;
  Buf, st, a: byte;
  F_in: File;
  buf1: array[0..2047] of byte;
  stop: array[0..3] of byte;
  F_out: file;

  i, j, it: integer;
  by: typesize;
  mid: string;
  BPS: integer; //Bits per Sample

  count, current: integer;

begin
  assignfile(F_in, finn);
  assignfile(F_out, foutn);

  randomize;
```

```
assignfile(f, 'c:\CourMyProj\tata.txt');
rewrite(f);

reset(F_in, 1);

rewrite(F_out, 1);



count:= col;

//memo1.Lines.Add(inttostr(col));
FromCardinalToBytes(col, by);

for i := 0 to 33 do                ///
  begin
    Blockread(F_in, buf, 1);
    Blockwrite(F_out, buf, 1);
  end;

 blockread(F_in, buf, 1);

 BPS:= buf div 8;
 if finn[length(finn)] = 'p' then
   bps:= 1;

 Blockwrite(F_out, buf, 1);
 for i := 35 to 104 - bps  do              ///
  begin
```

```
   Blockread(F_in, buf, 1);
   Blockwrite(F_out, buf, 1);
  end;

//memo1.Lines.Add(inttostr(col) + '   = col    ' + inttostr(filesize(f_in)));
memo1.lines.add('bps = ' + inttostr(bps));
col:= -132 + (filesize(f_in) div (bps));   ///// УБРАЛ БПС*8!!!!


while col > 0 do

//while not eof(f_in) do


begin
  dec(col);

   begin
    j:= bps;                    //    Участок отвеачающий
    while j > 1 do              //    за работу с
    begin
     //if not eof(f_in) then                        //    различными
     Blockread(F_in, buf, 1);          //    bps wav
     Blockwrite(F_out, buf, 1);   //
     dec(j);                    //
    end;
    //if not eof(f_in) then

    blockread(F_in, buf, 1);
    df:= bytetobin(buf);
```

```
      writeln(f, buf);/////////

      ///////////
      df[6]:= random(2);

      df[7]:= random(2);          ////////////////////////////////////

      buf:= bintobyte(df);
      blockwrite(F_out, buf, 1);
    end;
  end;

  while not eof(f_in) do
   begin
    blockread(f_in, buf, 1);
   // memo1.Lines.Add('ola la');
    blockwrite(f_out, buf, 1);
   end;

  closefile(F_in);
  closefile(F_out);
  closefile(f);
end;

begin


begin
  if opendialog2.Execute then
```

```
  fn:= opendialog2.FileName;
memo1.Lines.Add(fn);
assignfile(f, fn);
reset(f);
r:= filesize(f) div strtoint(edit2.Text);
for ii := 1 to strtoint(Edit2.Text)-1 do
  begin
    assignfile(f1, 'c:\decrypt\slices\slice' + inttostr(ii) + '.wav');
    rewrite(f1);
    for j := 0 to r do
      begin
        read(f, s);
        write(f1, s);
      end;
    closefile(f1);
  end;
assignfile(f1, 'c:\decrypt\slices\slice' + Edit2.Text + '.wav');
rewrite(f1);
while not eof(f) do
begin
  read(f,s);
  write(f1,s);
end;
closefile(f1);
setlength(fx, 1);
for ii := 1 to strtoint(Edit2.Text) do
  begin
    compressfile('c:\decrypt\slices\slice' + inttostr(ii) + '.wav', 'c:\decrypt\comp.zip');
    assignfile(f1, 'c:\decrypt\comp.zip');
    reset(f1);
```

```
    assignfile(f2, 'c:\decrypt\slices\slice' + inttostr(ii) + '.wav');
    reset(f2);
    setlength(fx, length(fx) + 1);
    memo1.Lines.Add(inttostr(filesize(f1)));
    memo1.Lines.Add(inttostr(filesize(f2)));

    fx[ii]:= (filesize(f1) / filesize(f2));
    closefile(f1);
    closefile(f2);
    deletefile('c:\decrypt\comp.zip');
   end;
   closefile(f);
   ///////////////ПОВТОРЕНИЕ ОПЕРАЦИИ ДЛЯ ПОЛНОСТЬЮ ЗАПОЛНЕННОГО
ФАЙЛА!!!!!!///////////////

  hidewaverandom(fn, 'c:\decrypt\me.wav');

  fn:= 'c:\decrypt\me.wav';



  assignfile(f, fn);
  reset(f);
  r:= filesize(f) div strtoint(edit2.Text);
   for ii := 1 to strtoint(Edit2.Text)-1 do
    begin
     assignfile(f1, 'c:\decrypt\slices\filledslice' + inttostr(ii) + '.wav');
     rewrite(f1);
     for j := 0 to r do
       begin
         read(f, s);
         write(f1, s);
```

```
      end;
    closefile(f1);
  end;

assignfile(f1, 'c:\decrypt\slices\filledslice' + Edit2.Text + '.wav');
rewrite(f1);
while not eof(f) do
begin
  read(f,s);
  write(f1,s);
end;
closefile(f1);
setlength(fy, 1);
for ii := 1 to strtoint(Edit2.Text) do
  begin
    compressfile('c:\decrypt\slices\filledslice' + inttostr(ii) + '.wav', 'c:\decrypt\comp.zip');
    assignfile(f1, 'c:\decrypt\comp.zip');
    reset(f1);
    assignfile(f2, 'c:\decrypt\slices\filledslice' + inttostr(ii) + '.wav');
    reset(f2);
    setlength(fy, length(fy) + 1);
    fy[ii]:= (filesize(f1) / filesize(f2));
    closefile(f1);
    closefile(f2);
    deletefile('c:\decrypt\comp.zip');
  end;
  closefile(f);
memo1.Lines.Add(floattostr(sig));
```

```
  reresult:= 0;
  for ii := 1 to strtoint(Edit2.Text) do
    begin
      sig:= abs(fx[ii] - fy[ii]);
      memo1.Lines.Add(inttostr(ii) + ' f(x) = ' + floattostr(fx[ii]));
      memo1.Lines.Add(inttostr(ii) + ' f(y) = ' + floattostr(fy[ii]));
      memo1.Lines.Add('delta = ' + floattostr(sig));

      if sig < strtofloat(Edit1.Text) then
      begin
          reresult:= reresult + 1;
          memo1.Lines.Add('|f(X, ' + inttostr(ii) + ') - f(Y, ' + inttostr(ii) + ') < Sigma' );
        end
      else
        begin
          reresult:= reresult - 1;
          memo1.Lines.Add('|f(X, ' + inttostr(ii) + ') - f(Y, ' + inttostr(ii) + ') > Sigma' );
        end;
    end;

end;

end;

/////////////КОНЕЦ ОЧИМОВ!!!!!!!!!!!!!!!!!!!!!!!!!!!!////////////////

////////Конец Стегоанализа

procedure TForm5.FormCreate(Sender: TObject);
```

```
begin


end;

//Keyfile
procedure TForm5.BitBtn1Click(Sender: TObject);
begin
  if OpenDialog1.Execute then
    Ideal:= OpenDialog1.FileName;
end;
//WaveHide
procedure TForm5.Button1Click(Sender: TObject);
var
  steg, fin: string;
begin
  if OpenDialog1.Execute then
    steg:= OpenDialog1.FileName;

  if SaveDialog1.Execute then
    fin:= SaveDialog1.FileName;
  HideWave(Container, Fin, Steg);
end;
end.
```