Лабораторная работа № 13

ИССЛЕДОВАНИЕ МЕТОДОВ ТЕКСТОВОЙ СТЕГАНОГРАФИИ

Цель: изучение стеганографических методов встраивания/извлечения тайной информации с использованием электронного файла-контейнера текстового формата, приобретение практических навыков программной реализации методов (рассчитана на 4 часа аудиторных занятий: 2 часа – часть 1, 2 часа – часть 2).

Задачи:

1. Закрепить теоретические знания из области текстовой стеганографии, классификации, моделирования стеганосистем подобного вида и сущности основных методов.
2. Изучить основные алгоритмы встраивания/извлечения тайной информации на основе методов текстовой стеганографии, получить опыт практической реализации методов.
3. Разработать приложение для реализации алгоритмов встраивания/извлечения тайной информации на основе методов текстовой стеганографии.
4. Познакомиться с методиками оценки стеганографической стойкости методов.
5. Результаты выполнения лабораторной работы (отдельно по каждой из 2 частей) оформить в виде описания разработанного приложения (для части 2), методики выполнения экспериментов с использованием приложений и результатов экспериментов.

13.1. Теоретические сведения

13.1.1. Классификация, сущность и основные особенности базовых методов текстовой стеганографии

Мы отмечали, что к текстовой стеганографии относятся методы, предусматривающие использование в качестве контейнера файла-документа текстового типа.

Многообразие методов текстовой стеганографии подразделяется на *синтаксические* методы, которые не затрагивают семантику

текстового сообщения, и *лингвистические*, которые основаны на эквивалентной трансформации текстовых файлов-контейнеров, сохраняющей смысловое содержание текста, его семантику (см. [2, 52–56]).

Для понимания сущности некоторых из методов полезно познакомиться с важнейшими особенностями и параметрами использования стилей (в том числе пространственно-геометрическими параметрами шрифтов), на основе которых строится текстовый файл-контейнер. На рис. 13.1 показаны основные из параметров шрифта.

Рис. 13.1. Параметры шрифта (источник: http://indians.ru/a-font-sizes.htm)

К синтаксическим методам компьютерной стеганографии, которые характеризуются *сравнительно невысокой эффективностью* (с точки зрения объема встраиваемой информации), относятся следующие (такие методы мы отнесем к числу *базовых* синтаксических методов):

- *изменение расстояния между строками* электронного текста (*Line-Shift Coding*); называется методом *изменения межстрочных интервалов*; сущность заключается в том, что используется текст с различными межстрочными расстояниями: выделяется максимальное и минимальное расстояния между строками, позволяющее кодировать соответственно символы «1» и «0» осаждаемого сообщения;

• *изменение расстояния между словами* в одной строке электронного текста (*Word-Shift Coding*); суть метода состоит в том, что осаждение информации основано на модификации расстояния между словами текста-контейнера;

• *изменение количества пробелов между словами* (частный случай метода *Word-Shift Coding*); основан та том, что, например, чередование одинарного пробела и двойного (xx_xx__xx) кодирует «1», переход же с двойного пробела на одинарный кодирует «0» (xx__xx_xx);

• *на основе внесения специфических изменений в шрифты*, т. е.начертания отдельных букв (*Feature Coding*); заключается в изменении написания отдельных букв используемого стандартного шрифта: визуально заметны различные образы, соответствующие буквам с верхними (например, *l*, *t*, *d*) или нижними (например, *a*, *g*) выносными элементами (см. рис. 13.1); например, букву «А» можно модифицировать, незначительно укорачивая длинную нижнюю часть буквы (рис. 13.2);

а *б*

Рис. 13.2. Пример применения метода *Feature Coding:* *а* – пустой контейнер; *б* – заполненный контейнер

• *изменение интервала табуляции*; аналогичен вышеописанному методу изменения количества пробелов, только в этом случае меняется не количество пробелов, а соответственно расстояние между строками и интервал табуляции;

• *Null Chipper* (дословно – *несуществующий, нулевой лепет*); предполагает размещение тайной информации на установленных позициях слов или в определенных словах текста-контейнера, который, как правило, лишен логического смысла (как видно, действительно лепет);

• *увеличение длины строки*; предусматривает искусственное увеличение длины каждой строки за счет пробелов: например, нет пробела (определяется положением знака перехода на новую строку) – «0», один пробел – «1» (рис. 13.3);

• *использование регистра букв*; для обозначения бита секретного сообщения, представленного единицей, используется символ нижнего регистра, а нулем – верхнего (или наоборот);

• *использование невидимых символов*; знак «*пробел*» кодируется символом с кодом 32, но в тексте его можно заменить также символом, имеющим код 255 (или 0), который является «невидимым» и отображается как пробел.

как·видно,·действительно·лепет);¶
как·видно,·действительно·лепет);·¶

Рис. 13.3. Пример реализации метода увеличения длины строки

Рассмотренные базовые методы могут применяться независимо и совместно, сохраняют исходный смысл текста, а обеспечиваемые ими показатели плотности кодирования при совмещении складываются.

Еще одна важная особенность. Перечисленные методы работают успешно до тех пор, пока тексты представлены в коде ASCII.

! **Методы также легко применяются к любому тексту, независимо от его содержания, назначения и языка. Синтаксические системы стеганографии легко реализуются в программном коде, так как они полностью автоматические и не требуют вмешательства оператора. Однако синтаксические методы неустойчивы к форматированию текста (вспомним робастность систем на основе ЦВЗ), и поэтому информация может быть потеряна при простом применении иного стиля форматирования текста-контейнера, скрывающего в себе стегосообщение. К тому же с помощью синтаксических методов можно передать незначительное количество информации.**

Существуют также стеганографические методы, которые интерпретируют текст как двоичное изображение. Необходимо отметить, что данные методы нечувствительны к изменению масштаба документа, что обеспечивает им хорошую устойчивость к большинству искажений, которые могут иметь место при активных атаках.

К числу основных лингвистических методов относятся [2, 52]:

• *метод синонимов*; в качестве примера приведем подмножество синонимов: {«*тайный*», «*секретный*», «*конфиденциальный*», «*доверительный*»}. В приведенном подмножестве каждое слово имеет единственное одинаковое смысловое значение, что позволяет закодировать каждое слово своим уникальным кодом

(т. е. выполнить операцию осаждения), например, «*доверительный*» – 00, «*конфиденциальный*» – 01, «*секретный*» – 10, «*тайный*» – 11. Подобное кодирование позволяет выбирать одно из четырех слов (как видим, они для удобства расположены по алфавиту) в зависимости от двух битов секретного сообщения. Отметим, что при этом, независимо какое из четырех слов будет выбрано, семантика сообщения не изменится. Очевидно, что при этом количество символов, соответствующих одному из синонимов используемого подмножества, зависит от общего числа элементов в подмножестве. Кроме того, обеим сторонам стеганосистемы должен быть известен общий алгоритм кодирования, т. е. один из ключей системы. Следует отметить, что в каждом подмножестве синонимов их упорядочивание должно выполняться по одному и тому же алгоритму и у отправителя сообщения, и у его получателя. В случае наличия слов с несколькими смысловыми значениями подобное кодирование оказывается невозможным. Также невозможно кодирование, если один из синонимов состоит из двух (или более) разделенных пробелом слов;

- *метод переменной длины слова*; основан на том, что длина слов в сообщении зависит от содержания секретного сообщения и способа кодирования слов: обычно одно слово текста-контейнера определенной длины кодирует два бита информации из стеганосообщения; например, слова текста длиной в 4 и 8 символов могут означать комбинацию битов «00», длиной в 5 и 9 – «01», 6 и 10 – «10», 7 и 11 букв – «11»; слова короче 4 и длиннее 11 букв можно вставлять где угодно для лексической и грамматической связки слов в предложении – программное приложение, которое декодирует принятое сообщение (извлекает сообщение из стеганоконтейнера), будет просто игнорировать их;
- *метод первой буквы* – программа-помощник в этом методе накладывает ограничение уже не на длину слова, а на первую (можно на вторую) букву; обычно одну и ту же комбинацию могут кодировать несколько букв, например, комбинацию «101» означают слова, начинающиеся с «А», «Г» или «Т;
- *мимикрия*; мимикрия генерирует осмысленный текст, используя синтаксис, описанный в *Context Free Grammar* (CFG), и встраивает информацию, выбирая из CFG определенные фразы и слова; грамматика CFG – это один из способов описания языка, который состоит из статических слов и фраз языка, а также узлов.

13.1.2. Методы текстовой стеганографии на основе модификации цветовых и пространственно-геометрических параметров символов текста-контейнера

Поддерживаемые форматы документов-контейнеров, которые мы хотим рассматривать как объекты для скрытия тайной информации – любые, способные хранить цвет и иные указанные параметры символов и которые можно открыть с помощью процессора MS Office Word: (*.doc, *.docx), *.rtf (межплатформенный формат хранения размеченных текстовых документов), *.odt (открытый формат документов для офисных приложений).

Что касается реализации метода на основе модификации цвета символов текста-контейнера [55, 56, 67, 68], по сути своей он схож с классическим методом наименее значащих битов (см. лабораторную работу № 12) и опирается на использовании цветовой модели RGB (см. рис. 12.2). Подробный алгоритм реализации метода можно найти, например, в [67].

К основным пространственно-геометрическим параметрам символов (и текста в целом), которые могут быть использованы как инструменты для стеганографического преобразования, относятся апрош (обозначен цифрами 7 и 8 на рис. 13.1) и кернинг.

13.1.2.1. Метод на основе апроша

Апрош определяет расстояние между соседними символами текста. Фактически апрош состоит из двух таких расстояний – полуапрошей, являющихся как бы пространством, прилегающим к каждому из символов-соседей (см. рис. 13.1 и 13.4). Мы далее будем обращаться только к апрошу. Согласно существующим техническим правилам набора нормальный апрош должен быть равен половине кегля (размера) шрифта.

Идея метода [69, 70] заключается в следующем. Встраивание сообщения в контейнер может быть основано на модификации базового (устанавливаемого текстовым процессором по умолчанию) значения апроша a_o, его изменением от базового до некоторого максимального $a_{\max}$ (или минимального $a_{\min}$), которое зрительно не должно отличаться от стандартного. Такое изменение производится с определенным шагом (дискретно) Δa_i, каждому значению которого присваивается определенный бит или определенная комбинация битов.

Рис. 13.4. Измерение апроша

Изменение величины апроша между двумя определенными символами текста относительно базового значения a_o на небольшое расстояние (пункты (пт) или доли пункта) формально можно представить в следующем виде:

$$a_t = a_o + \Delta a_t \,. \qquad (13.1)$$

Такое изменение не должно вызывать визуально заметного уплотнения ($\Delta a_t < 0$) или разрежения ($\Delta a_t > 0$) групп символов. В текстовом процессоре MS Word апрош может принимать значения в диапазоне от 0 до 1584 пунктов.

Для примера и визуального представления об особенностях установки данного пространственно-геометрического параметра шрифта на рис. 13.5 приведена строка текста с различными параметрами апроша.

При использовании метода осаждение секретного сообщения
При использовании метода осаждение секретного сообщения
При использовании методаосаждениесекретногосообщения
При использовании метода осаждение секретного сообщения
При использовании метода осаждение секретного сообщения
При использовании метода осаждение секретного сообщения

Рис. 13.5. Примеры использования различного апроша

На этом рисунке вторая строка оформлена с использованием стандартного (обычного) апроша. В первой строке применено уплотнение во всех словах, кроме первого: во втором слове – на 0,1 пт, в третьем – на 0,2 пт, в четвертом – на 0,3 пт, в пятом – на 0,4 пт, в шестом – на 0,5 пт; в третьей строке слова со второго по шестое оформлены с изменением Δa в противоположную сторону, т. е. с разрежением. В четвертой строке применялось уплотнение (отрицательный Δa) или разрежение (положительный Δa) лишь к отдельным символам первого («При») и второго («использовании») слов: для «П» – Δa = –0,1 пт; «ри» – Δa = –0,2; «и» –Δa = 0;

«с» – $\Delta a = 0{,}1$; «п» – $\Delta a = 0$; «о» – $\Delta a = -0{,}1$; «л» – $\Delta a = -0{,}2$; «ь» – $\Delta a = -0{,}3$; «з» – $\Delta a = 0{,}4$; «ов» – $\Delta a = -0{,}3$; «ан» – $\Delta a = 0{,}2$; «ии» – $\Delta a = 0$. Пятая строка целиком отформатирована при $\Delta a = 1$ пт, а шестая – при $\Delta a = -1$ пт.

Особенностью рассматриваемого метода является возможность одноразового размещения (в апроше одного символа) числа битов, определяемого дискретной разницей между минимальным и максимальным значениями Δa. Например, если отсчет вести от $\Delta a_{\min}$ до установленного интервала Δa_t в виде параметра $0{,}1n_д$ (пт), то количество условных дискретных единиц $n_д$, представленное в бинарном виде, определяет число битов, которые можно таким образом разместить; например, $\Delta a_{\min} = -0{,}5$ пт, а $\Delta a_t = 0{,}3$ пт. Разница между этими величинами составляет 0,8 пт: $8 \cdot 0{,}1$ или $n_{дt} = 8$ (в двоичном виде – 1000; в первом приближении именно такую бинарную комбинацию можно разместить (осадить) путем модификации конкретного апроша). На этой основе могут быть разработаны различные варианты кодировки осаждаемых комбинаций.

13.1.2.2. Метод на основе кернинга

В текстовых документах встречаются такие сочетания знаков, которые образовывают визуальные «дыры» либо «сгущения». Например, в текстах на основе кириллицы – это такие сочетания: «ГА», «ТА», «АТА», «ЬТ» и т. п., на основе латиницы – «AY», «AV», «T;», «ff», а на основе греческого алфавита – «ΘA», «ΔO», «λк» и др. Такие сочетания называются *кернинговыми парами*. Особенности «кернингования» приведены на рис. 13.6.

Под *кернингом* обычно понимается процесс изменения межсимвольного расстояние между отдельными парами символов или кернинговыми парами (именно фактор парности отличает кернинг от апроша).

Рис. 13.6. Пояснение к понятию кернинга

Таким образом, технология кернинга, появившаяся в полиграфии после внедрения фотонабора (а затем и компьютерного набора), включает подбор межбуквенных интервалов для конкретных пар букв с целью улучшения внешнего вида и удобочитаемости текста. Такой избирательный подбор позволяет компенсировать неравномерности визуальной плотности текста, получаемой при использовании стандартных апрошей для каждой буквы.

Очевидно, что промежуток между «A» и «V» в первой строке на рис. 13.6 гораздо больше, чем во втором, хотя формально они одинаковы. В данном случае сочетание «AV» как раз и является кернинговой парой. После применения кернинга визуальное восприятие текста улучшилось.

С появлением цифрового фотонабора стало возможным хранить такие критические сочетания знаков (кернинговые пары) для некоторого условного шрифта, общее число которых мы обозначим N_k, в памяти компьютера с указанием величины (σ), на которую необходимо сдвинуть символы, чтобы визуально выровнять буквенные просветы.

Как правило, текстовые редакторы или процессоры содержат встроенные средства настройки кернинга, который определяет стандартный межбуквенный интервал для того или иного шрифта. При этом σ устанавливается в соответствии со значениями из таблицы кернинговых пар, встроенной в вышеуказанный файл со шрифтом. Такая настройка позволяет выровнять шрифт и является стандартной. В некоторых шрифтах сейчас количество пар доходит до нескольких тысяч.

Значение кернинга может быть как положительным (когда знаки раздвигаются, $\sigma > 0$), так и отрицательным (когда сдвигаются, $\sigma < 0$). Эта величина в программах верстки устанавливается в процентах от ширины символа пробела используемого шрифта.

Следуя общепринятой методике, кернинг, как параметр, будем измерять в тысячных долях круглой шпации (*Em*) – единицы измерения, которая определяется относительно текущего размера шрифта и равна ширине символа «M». Например, для шрифта размером 6 пунктов 1 круглая шпация равна 6 пунктам, а для шрифта размером 10 пунктов – 10 пунктам. Таким образом, размер кернинга σ строго пропорционален текущему размеру шрифта. Сдвиги букв относительно автоматически установленного межсимвольного расстояния (измеряемого, например, апрошем) можно производить

с различным шагом: от 0,01 до 0,04 величины *Em*, в зависимости от нужной точности.

Как следует из анализа общих принципов задания размера кернинга и управления этим размером, предлагаемый метод основывается на принудительном применении кернинга, не зависящем от установок параметров текста-контейнера, созданных средствами текстового процессора или иного специализированного текстового редактора. Здесь есть одна важная особенность. Изменять значения кернинга можно лишь в программах верстки (например, InDesign) или в программах, предназначенных для работы с векторной или растровой графикой (например, CorelDraw, Photoshop). В текстовом процессоре MS Word значения кернинга автоматически установлены в таблицах кернинговых пар каждого шрифта, доступа к которым нет.

Здесь возможности пользователя практически связаны лишь с указанием минимального размера шрифта, для которого можно применять кернинг. Это означает, что если текст набран на основе шрифта размером, например, 14 пт, а мы в специальной опции (*Главная/Шрифт*) установили минимальный размер в 15 пт, при котором будет выполняться кернинг (см. поясняющую иллюстрацию на рис. 13.7), то для нашего текста эта процедура (иногда ее называют «кернингованием») не будет выполнена.

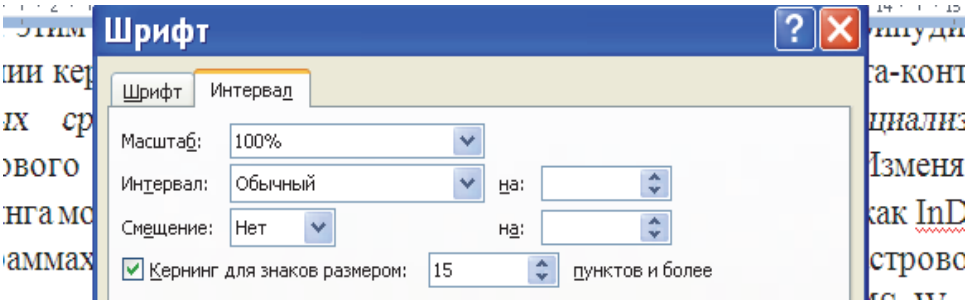

Рис. 13.7. Пример установки параметров для применения кернинга

Таким образом, рассматриваемый метод может быть интерпретирован так, что само значение кернинга мы программно не изменяем, а изменяем лишь размер символов, к которым будет применен кернинг в результате осаждения секретной информации.

Следует принять во внимание также следующую важную особенность. Текстовый процессор MS Word позволяет изменять параметры кернинга для знаков определенного размера. Размер символов, к которым может быть применим кернинг, должен быть

в диапазоне между 1 и 1638 пт; в частном случае этот размер может быть указан в последней строке окна на рис. 13.7 (а сам размер шрифта можно устанавливать с точностью до 0,5 пт).

В [69] в общем виде даны алгоритмы встраивания/извлечения сообщений на основе кернинга.

Заметим, что выбор символов текста для встраивания сообщения на основе модификации цветовых и пространственно-геометрических параметров может выполняться на основе ранее описанных принципов в соответствии с одним из элементов ключевой информации:

- локальный разброс;
- случайный (псевдослучайный) разброс;
- случайный (псевдослучайный) разброс с памятью;
- подряд.

13.1.3. Особенности программной реализации методов

Общий принцип работы некоторого условного приложения проиллюстрирован на диаграмме деятельности (рис. 13.8).

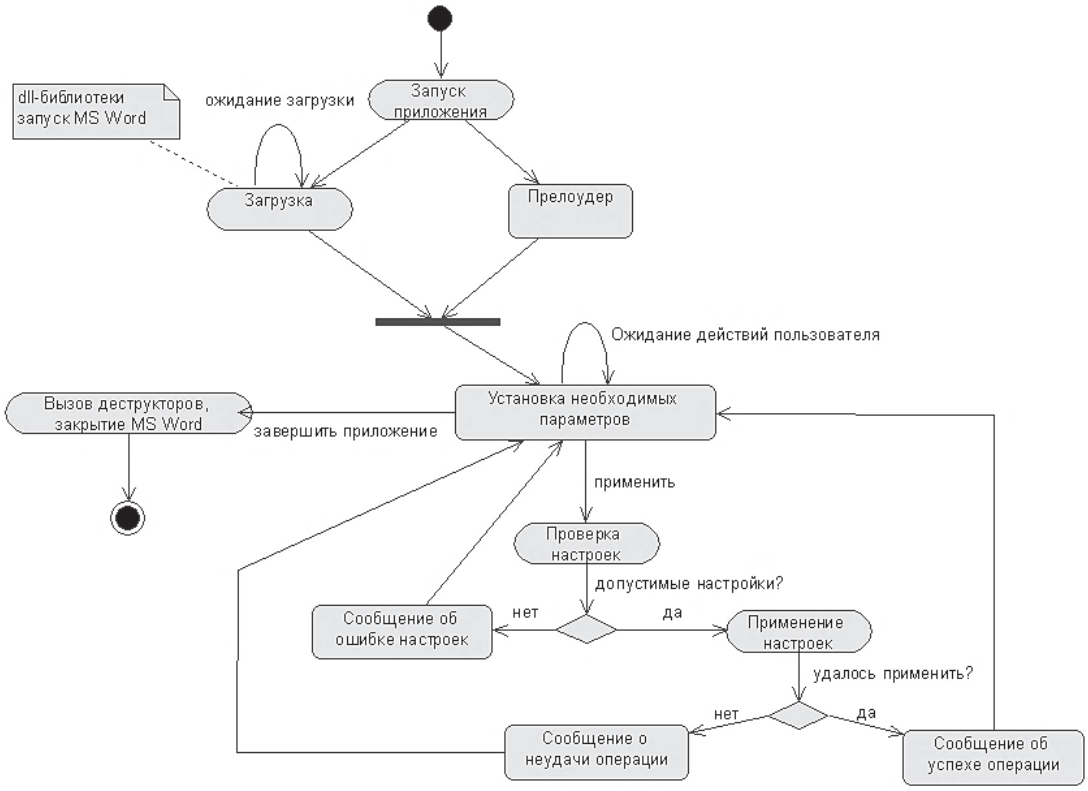

Рис. 13.8. Диаграмма деятельности программного средства, реализующего методы текстовой стеганографии на основе MS Word

Некоторые особенности программной реализации рассмотренных методов проанализируем на конкретном примере.

Пусть интерфейс программного средства включает кнопки «*Анализировать*», «*Встроить*», «*Получить*» и «*Справка*».

Пример общей блок-схемы алгоритма функционирования приложения показан на рис. 13.9.

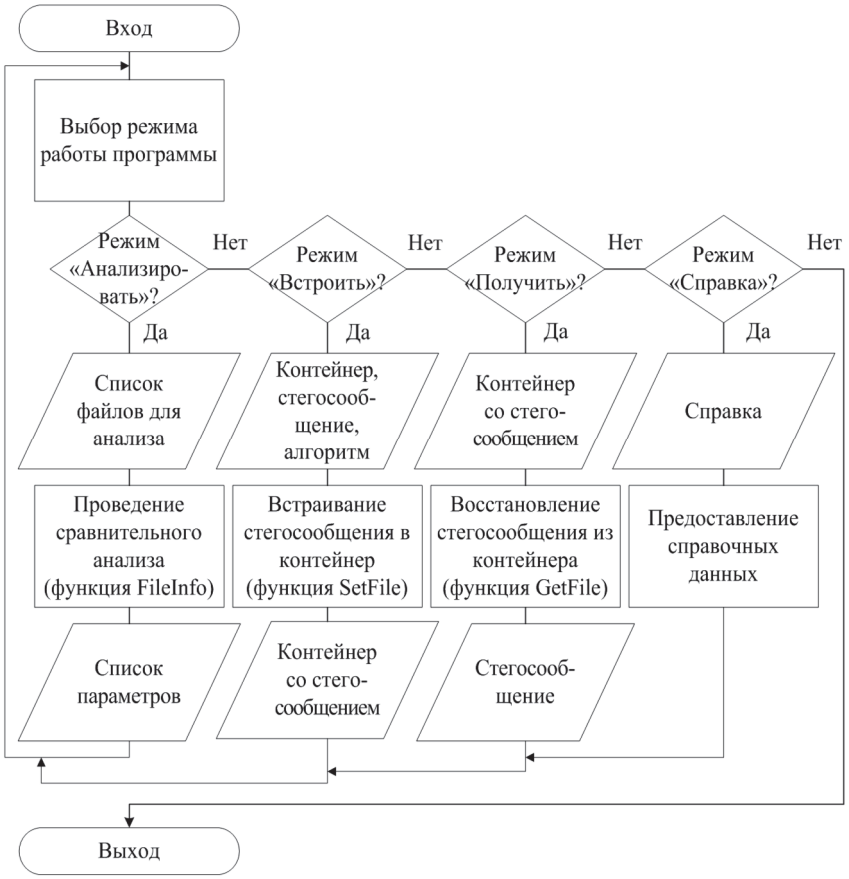

Рис. 13.9. Пример общей блок-схемы функционирования программного средства, реализующего методы текстовой стеганографии

Вкладка «*Анализировать*» позволяет получить количественные характеристики методов *изменения регистра буквы*, *изменения цвета символов*, *изменения масштаба символов*, *изменения апроша*, *изменения кернинга*, а также методов *Line-Shift Coding*, *Word-Shift Coding*, *Feature Coding* при использовании в качестве контейнера выбранных текстовых файлов. По полученным результатам можно произвести сравнение эффективности использования конкретного метода по сравнению с другими.

Вкладка «*Встроить*» позволяет осуществить встраивание (осаждение) сообщения в контейнер по выбранному алгоритму.

По умолчанию каждый метод производит встраивание 1 бита файла-сообщения в соответствующем параметре каждого символа файла-контейнера. В [71] приведены коды основных функций анализируемого приложения, реализованные в C#.

13.1.4. Краткое описание специализированных программных средств

Для выполнения основного задания в данной лабораторной работе целесообразно ознакомиться (в качестве примера) со структурой, интерфейсом, особенностями функционирования и программной реализации доступных инструментальных средств, реализующих методы текстовой стеганографии.

13.1.4.1. Программное средство «Sword»

Функционал программы построен на основе диаграммы, показанной на рис. 13.8. Основное окно средства показано на рис. 13.10.

Как видно из рис. 13.10, интерфейс программы имеет несколько блоков установки. В блоке «*Текст для скрытия*» пользователь может ввести вручную с клавиатуры или вставить из буфера обмена секретное сообщение, которое необходимо скрыть в документе-контейнере. Имеется возможность использовать в качестве сообщения уже существующий электронный текстовый документ (кнопка «*Взять из файла*»). При выборе необходимого документа предоставляются три фильтра, а также возможность выбора любого файла (рис. 13.11).

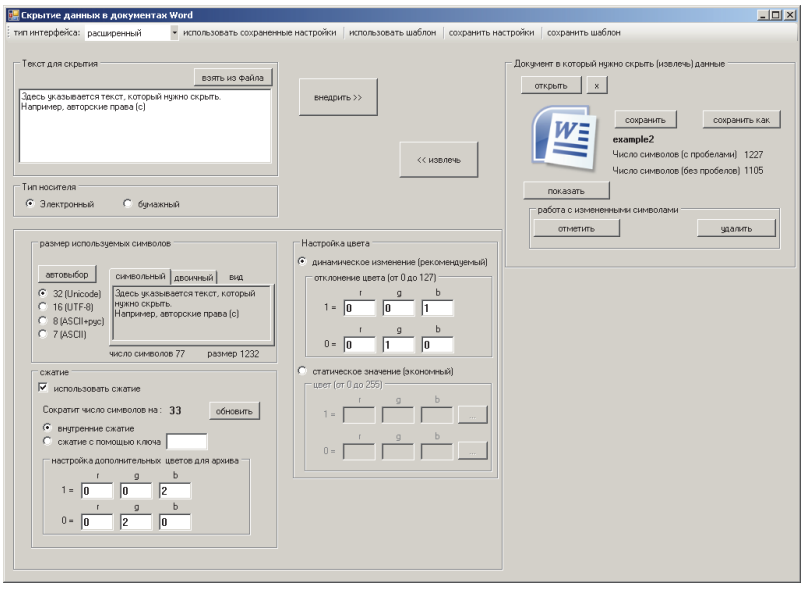

Рис. 13.10. Основное диалоговое окно программного средства «Sword»

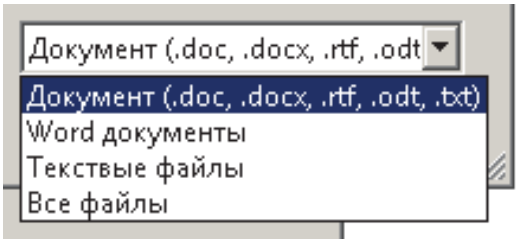

Рис. 13.11. Выбор формата используемого документа

В области «*Тип носителя*» можно выбрать вид текстового документа-контейнера – электронный или бумажный. Разница заключается в том, что при выборе типа «бумажный» информация будет встраиваться лишь в символы, тогда как в типе «электронный» осаждение будет осуществляться и в пробельные элементы.

С помощью переключателей в блоке «*Размер используемых символов*» можно посмотреть, как выбранный текст будет выглядеть в двоичном виде в выбранной кодировке. В программе используется четыре основных вида кодировки: ASCII, русская таблица кодов ASCII, UTF-8 и Unicode. При смене кодировки исходный текст не теряется. Поэтому, если в одной кодировке символы поменялись, можно выбрать другую – и они восстановятся. Также в данном блоке отображаются сведения о количестве встраиваемых символов (параметр необходимо учитывать при выборе документа-контейнера), о необходимом минимальном размере файла, в который будет происходить осаждение сообщения.

Использование 16-ричной и 32-ричной систем представления обусловлено тем, что при такой кодировке стегосообщение можно скрыть в меньшем объеме текста. Например, нам необходимо зашифровать секретное сообщение «Слово» в текстовом документе. При этом число символов в документе, необходимых для скрытия, в двоичной системе составляет 79, а в 32-ричной – 15.

С помощью кнопки «*Настройки цвета*» можно выбрать значение отклонения исходного цвета символа, т. е., иначе говоря, в данной области выбирается значение соответствующей ключевой информации. Используемый ключ можно сохранить для последующего его использования при извлечении секретной информации. Эти файлы являются обычными текстовыми документами, в которых хранятся настройки, структурированные специальным способом. Для того чтобы отличить файлы с такими настройками от других текстовых документов, первые сохраняются с расширением **.sword*.

В данной области предусмотрено два типа изменения цвета: динамическое и статическое. При динамическом встраивании за основу берется один какой-либо символ (символ-образец с индексом *t*), считывается его цвет. Необходимое секретное сообщение преобразуется в двоичный вид. Программа «распределяет» секретное сообщение внутри текста-контейнера в соответствии с ключом.

Если выбрать «*Статическое изменение цвета*», то для «0» и «1» необходимо указать конкретные значения трех цветовых каналов (*R*, *G*, *B*) либо выбрать конкретный цвет на вкладке (рис. 13.12). Программа создает переменные, которым присваивает значения для цветовых каналов. Однако наиболее эффективным и наименее заметным является динамическое изменение цвета.

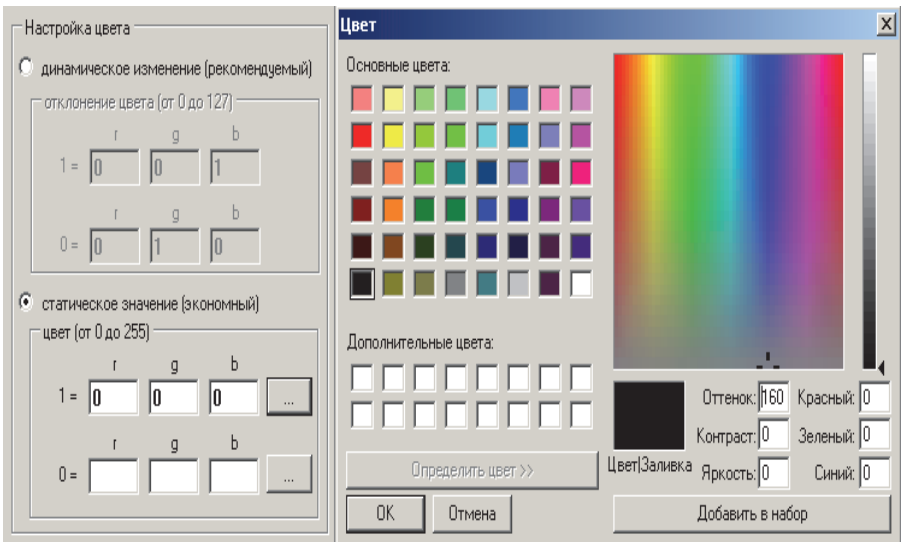

Рис. 13.12. Интерфейс для настройки цвета скрываемых символов в приложении *Sword*

И, наконец, в правой части главного окна можно выбрать электронный текстовый документ, в который необходимо произвести скрытие (контейнер). После осаждения сообщения в документ-контейнер можно просмотреть позиции, в которых скрыта информация. Соответствующие символы и пробельные элементы будут выделены синим маркером (рис. 13.13).

Рис. 13.13. Фрагмент документа-контейнера с выделением модифицированных символов

После инициализации документа с его содержимым можно производить любые действия. При внедрении исследуется именно текущее содержание документа, а не то, которое было на момент открытия. После извлечения текст будет помещен в поле «*Текст для скрытия*».

С программным кодом средства можно познакомиться по ссылке [72].

13.1.4.2. Программное средство «QuatesEmbed»

Здесь реализованы стеганографические методы для контейнеров DOCX-формата на основе модификации такого символа, как кавычка, которая может быть одинарной или двойной. Использование кода (0 или 1), соответствующего виду кавычки, позволяет осаждать информацию.

Программный код приложения можно найти по ссылке [73]. Для работы с приложением необходимо, чтобы на компьютере пользователя был установлен *.NETFramework*.

Для хранения информации о документе-контейнере при выполнении осаждения создан вспомогательный класс *XMLFile*, являющийся абстракцией над реальными документами *XML*, описание которого представлено в листинге 13.1.

```
class XMLFile
{
      public StringBuilder File { get; set; }

     public decimal GetContainerCapacity(int bitsPerSymbol)
     {
          int containerCapacity = 0;
          try
          {
               for (int i = 0; i< File.Lenght; i++)
               {
                     if (File[i] == '\' ' || File[i] == '\'' ')
                           {
                              containerCapacity++;

                              if (File[i] == '\' ')
                              {
                                    while (File[++i] != '\' ');
                              }
                              else if (File[i] == '\'' ')
                              {
```

```
                                        while (File[++i] != '\''
                                    ');
                                    }
                                }
                            }
                        }
                catch (IndexOutOfRangeException e)
                {
                        return containerCapacity;
                }
        return Math.Floor((decimal) containerCapacity / bitsPerSymbol;
    }
}
```

Листинг 13.1. Описание класса *XMLFile*

Метод *GetContainerCapacity* позволяет получить информацию о размере загруженного в приложение контейнера. На основании подсчета количества пар двойных и одинарных кавычек в документе, учитывая язык внедряемого сообщения, он возвращает максимальное число символов, из которых может состоять внедряемое сообщение. Класс *XMLFile* имеет свойство *File*, содержащее текст считываемого XML-документа. Данное свойство имеет тип *StringBuilder*, позволяющее изменять содержимое свойства, не создавая при этом лишних объектов. Это позволяет обрабатывать большие файлы, при этом эффективно используя оперативную память.

Для манипуляции файлами создан класс *FileManager*, содержащий набор методов для перемещения, а также изменения содержимого документов. В листинге 13.2 представлены методы *ReadXMLFile* и *WriteXMLFile*, позволяющие считывать/изменять содержимое реального XML-файла и сохранять его состояние в объект класса *XMLFile*.

```
public static string ReadXMLFile(string path)
{
      return File.ReadAllText(path);
}

public static void WriteXMLFile(XMLFile XMLFile, string path)
{
      File.WriteAllText(path, XMLFile.File.ToString());
}
```

Листинг 13.2. Методы, изменяющие содержимое XML-файлов

Для того чтобы извлечь XML-документ из DOCX-файла, необходимо изменить расширение с DOCX на ZIP. Методы, реализующие смену расширения файла, представлены в листинге 13.3.

```
publicstaticvoidWriteXMLFile(XMLFileXMLFile, stringpath)
{
      File.WriteAllText(path, XMLFile.File.ToString());
}

public static void CopyFileAndChangeExtentionToZip(string
file)
{
      DeleteTempArchive(tempArchiveStorage);
      File.Copy(file,    Path.ChangeExtention(tempArchive
      Storage, ".zip"));
}
```

Листинг 13.3. Методы, реализующие смену расширения файлов

Перед внедрением сообщения в XML-документ его необходимо конвертировать в бинарный вид при помощи метода *Make BinaryString*.

После стеганографического преобразования измененный XML-документ помещается в архив при помощи метода *AddStego ContainerToArchive* класса *Embedder*. Для извлечения содержимого XML-документа из ZIP-архива реализован метод *ReadDocument FromZipFile*. Метод принимает на вход путь, по которому можно обратиться к XML-документу, и, используя метод *Open* класса *ZipFile* из пространства имен *System.IO.Compression*, считывает текст XML-файла.

Для внедрения сообщения в XML-документ используется метод *EmbedMessage*. Представленный метод заменяет в документе двойные и одинарные кавычки в соответствии с бинарной последовательностью, полученной из осаждаемого сообщения. Данный метод модифицирует принимаемый XML-документ, который при помощи метода *AddStegoContainerToArchive* размещается в ZIP-архиве.

Извлечение бинарной последовательности производится при помощи метода *ExtractMessage*. На вход метод принимает объект класса *XMLFile*, в котором находится строковый объект, содержащий разметку XML-документа. Для оптимизации скорости работы алгоритма в качестве переменной, в которую последовательно записывается извлекаемое сообщение, используется класс *StringBuilder*.

После завершения считывания объект данного класса конвертируется в строку, с которой производятся дальнейшие операции согласно алгоритму извлечения. Представленный метод проходится последовательно по передаваемому на вход документу и считывает все кавычки. В процессе считывания одинарной кавычке ставится в соответствие двоичный 0, а двойной – двоичная 1. Считываемая бинарная последовательность сохраняется как переменная *ExtractMessage*.

Главное окно приложения показано на рис. 13.14. Назначение элементов интерфейса не требует пояснений.

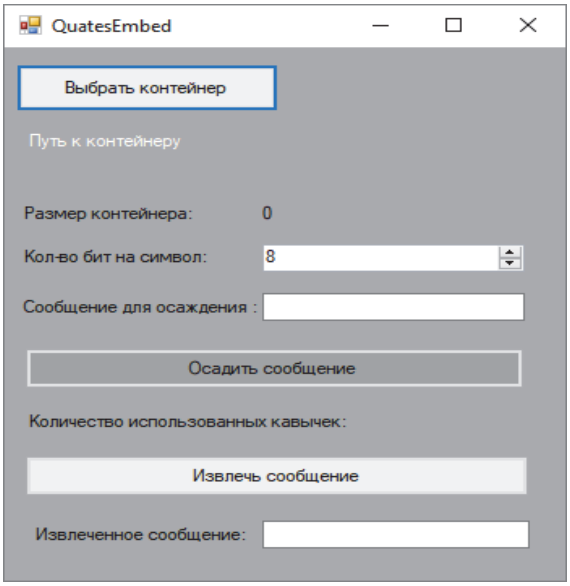

Рис. 13.14. Главное окно приложения «*QuatesEmbed*»

Как видно из данных на рис. 13.15, для размещения сообщения (information) «использовались» 88 пар ковычек.

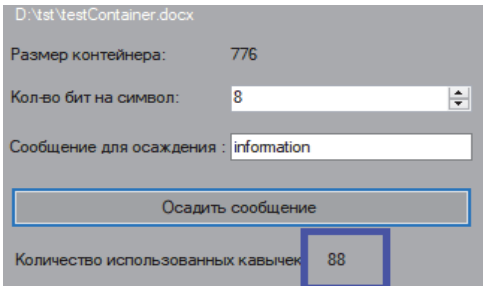

Рис. 13.15. Одно из информативных окон приложения

Мы рассмотрели некоторые особенности программной реализации стеганометодов и использования готовых средств для лучшего понимания сущности основного лабораторного задания и более осознанного его выполнения.

13.2. Практическое задание

1. Перед выполнением основного задания целесообразно познакомится со структурой, интерфейсом и функциональными особенностями приложений «*Sword*» и «*QuatesEmbed*». Для этого необходимо: инициировать приложения, выбрав в указанном преподавателем каталоге соответствующий исполнительный (*.exe*) файл.

1.1. Используя программное средство «*Sword*», встроить (секретное) сообщение в текстовый документ-контейнер, который необходимо выбрать, и произвести операцию извлечения осажденного сообщения:

- выбрать ключ («*Настройка цвета*»); ключ определяет, на сколько единиц будет изменено значение цвета символа, т. е. его отклонение от исходного; значение отклонения задается для трех базовых цветов: красного (r), зеленого (g), синего (b);
- выбрать встраивамое сообщение («*Текст для скрытия*»); сообщение можно вводить с клавиатуры либо загружать из файла; стеганосообщение (тайное сообщение) в различных системах счисления можно просмотреть в области «*Размер используемых символов*»);
- выполнить операцию встраивания (нажать кнопку «*Внедрить*»); после осаждения информации можно просмотреть используемые для этого символы контейнера, выделив их специальным маркером (кнопка «*Отметить*»);
- последовательно выделить каждый из модифицированных символов и проанализировать его цветовые характеристики (используйте кнопку «*Цвет текста*»); определить визуально, какое максимальное отклонение цвета от исходного является незаметным;
- извлечь секретное сообщение из стеганоконтейнера, используя известный ключ.

Указанные в п. 1.1 операции выполнить:

- для различных типов кодировки;
- для различных типов носителя;
- для различных режимов настройки цвета.

Оценить эффективность различных режимов встраивания (отношение максимального объема осаждаемой информации к объему контейнера, при которых визуальные изменения контейнера практически отсутствуют).

1.2. Используя программное средство «*QuatesEmbed*», встроить (секретное) сообщение в документ-контейнер, который необходимо выбрать, и произвести операцию извлечения осажденного сообщения:

• для выбора контейнера необходимо нажать кнопку «*Выбрать контейнер*» на главной странице приложения – появится окно, позволяющее указать файл, хранящийся на жестком диске пользователя (рис. 13.16);

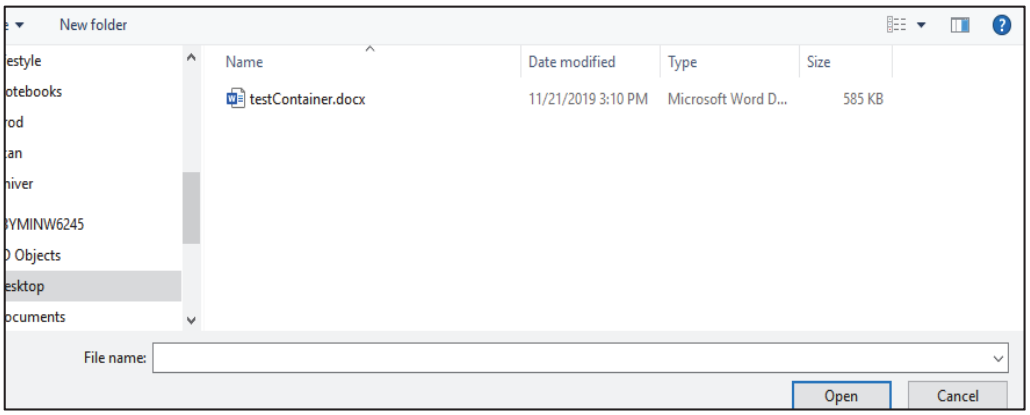

Рис. 13.16. Модальное окно выбора контейнера

• нажать на кнопку «*Open*», после чего файл откроется в приложении; приложение анализирует файл-контейнер на предмет возможности внедрения информации (путем подсчета кавычек в XML-документе); после этого информация о размере контейнера отображается в окне приложения (пример на рис. 13.17);

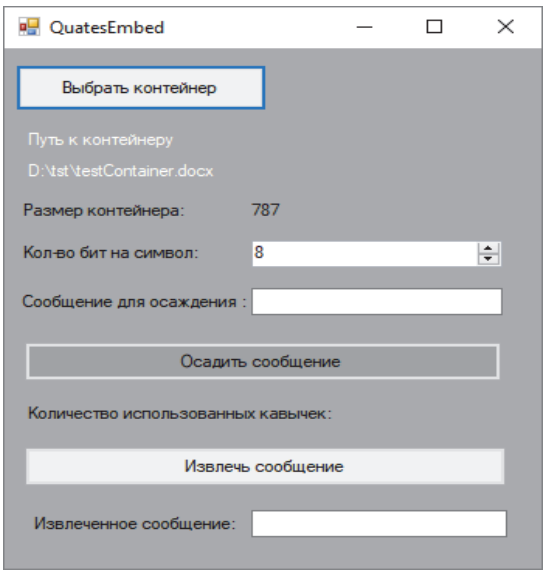

Рис. 13.17. Результаты анализа пустого контейнера

• указать количество битов, отводимое для осаждения одного символа сообщения (на рис. 13.17 указано 8 битов); отметим, что если внедряемое сообщение – на русском языке, то на один символ сообщения будет отводиться 11 битов в контейнере, в то время как для сообщения на английском языке можно использовать всего лишь 8 битов;

• указать внедряемое сообщение;

• выполнить операцию внедрения; после выполнения всех операций при помощи всплывающих окон (после сохранения стеганоконтейнера) пользователю будет доступной информация о том, сколько пар кавычек было использовано в процессе внедрения сообщения.

Для осуществления процедуры извлечения информации из стеганоконтейнера необходимо:

• установить количество битов в поле «*Кол-во бит на символ*», равное значению, которое использовалось для внедрения сообщения;

• нажать на кнопку «*Извлечь сообщение*» и выбрать стеганоконтейнер.

Выполнить указанные в п. 1.2 операции над контейнерами различного объема, содержащими разнотипную информацию. Оценить достигаемый в каждом случае эффект.

Сравнить эффективность методов, реализованных в обоих программных средствах, которые исследовались в части 1 практического задания.

2. Разработать авторское приложение, реализующее один из методов текстовой стеганографии на основе модификации пространственно-геометрических параметров текста-контейнера. Варианты заданий приведены в таблице. Дополнительные параметры согласуются с преподавателем.

Варианты заданий

Вариант	Реализумые методы
1	Модификация расстояния между строками; модификация апроша
2	Модификация расстояния между словами; модификация цвета
3	Модификация числа пробелов; модификация кернинга
4	Изменение длины строки; модификация апроша
5	Модификация расстояния между строками; модификация цвета
6	Модификация расстояния между словами; модификация апроша
7	Модификация числа пробелов; модификация апроша
8	Модификация числа пробелов; модификация цвета

Окончание таблицы

Вариант	Реализумые методы
9	Модификация числа пробелов; модификация кернинга
10	Модификация расстояния между строками; модификация кернинга
11	Изменение длины строки; модификация кернинга
12	Изменение длины строки; модификация цвета
13	Модификация расстояния между словами; модификация кернинга
14	Метод переменной длины слов; модификация апроша
15	Метод переменной длины слов; модификация цвета
16	Метод переменной длины слов; модификация кернинга

Результаты выполнения каждой части практического задания оформить в виде отчета по установленной форме.

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. В чем состоит сущность методов текстовой стеганографии?
2. Охарактеризовать методы синтаксической текстовой стеганографии. Привести примеры конкретной реализации методов.
3. Охарактеризовать методы лингвистической текстовой стеганографии. Привести примеры конкретной реализации методов.
4. Дать оценку стеганографической стойкости методов текстовой стеганографии при конвертации текста-контейнера в иной текстовый формат.
5. Дать оценку стеганографической стойкости методов текстовой стеганографии при визуальном стеганоанализе текста-контейнера.
6. Дать общую характеристику стеганоанализу в области текстовой стеганографии на основе метода «χ-квадрат».
7. Что такое апрош? В чем состоит сущность стеганометода на основе модификации апроша?
8. Что такое кернинг? В чем состоит сущность стеганометода на основе модификации кернинга?
9. Дать сравнительную оценку методов на основе модификации пространственно-геометрических и цветовых параметров символов текста-контейнера (критерий: отношение оправданного объема осаждаемой информации к объему контейнера).
10. Какие новые методы текстовой стеганографии вы можете предложить?