ОГЛАВЛЕНИЕ

[**ИСПОЛЬЗУЕМЫЕ ОПРЕДЕЛЕНИЯ** 4](#_Toc527465601)

[**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ** 5](#_Toc527465602)

[**ВВЕДЕНИЕ** 6](#_Toc527465603)

[**РАЗДЕЛ 1. ОБЩИЕ СВЕДЕНИЯ О СТЕГАНОГРАФИИ** 9](#_Toc527465604)

[1.1 Современные проблемы стеганографии 9](#_Toc527465605)

[1.2 Клиент-серверное взаимодействие 12](#_Toc527465606)

[1.3 Цифровая стеганография 14](#_Toc527465607)

[1.4 Постановка цели 16](#_Toc527465608)

[**РАЗДЕЛ 2. КЛАССИФИКАЦИЯ МЕТОДОВ** 16](#_Toc527465609)

[2.1 Классификация методов сокрытия информации 16](#_Toc527465610)

[2.2 Обзор методов цифровой стеганографии 17](#_Toc527465611)

[2.3 Методы встраивания в пространственные области изображений 19](#_Toc527465612)

[**РАЗДЕЛ 3. ОБОСНОВАНИЕ ВЫБРАННОГО МЕТОДА** 22](#_Toc527465613)

[3.1 Метод встраивания в область ДПФ 22](#_Toc527465614)

[3.2 Алгоритм встраивания ЦВЗ 24](#_Toc527465615)

[3.4 Дискретное косинус-преобразование 26](#_Toc527465616)

[3.5 Least Significant Bits (LSB) 27](#_Toc527465617)

[3.6 Обоснование выбранного формата. 29](#_Toc527465618)

[3.7 Математическая модель цифрового изображения JPEG 30](#_Toc527465619)

[**РАЗДЕЛ 4. АНАЛИЗ КРИТЕРИЕВ ВЫБОРА КОНТЕЙНЕРА** 35](#_Toc527465620)

[4.1 Общие критерии выбора контейнеров 35](#_Toc527465621)

[4.2 Классификация критериев выбора контейнера для LSB-метода 36](#_Toc527465622)

[4.4 Критерий эффективности в стеганографии изображений 38](#_Toc527465623)

[4.6 Атаки на стегосистему. 41](#_Toc527465624)

[**РАЗДЕЛ 5. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ** 45](#_Toc527465625)

[5.1 Назначение и структура ПО 45](#_Toc527465626)

[5.2 Выбор инструментов 45](#_Toc527465627)

[5.3 Описание основных компонентов программы. 46](#_Toc527465628)

[Перспективы 51](#_Toc527465629)

[**ЗАКЛЮЧЕНИЕ** 52](#_Toc527465630)

[**ЛИТЕРАТУРА** 53](#_Toc527465631)

[**ПРИЛОЖЕНИЕ** 54](#_Toc527465632)

[Исходный код lsb\_steganography.py: 54](#_Toc527465633)

[Исходный код lsb\_steganography\_GUI.pyw: 59](#_Toc527465634)

## **ИСПОЛЬЗУЕМЫЕ ОПРЕДЕЛЕНИЯ**

В выпускной квалификационной работе используются следующие термины с соответствующими определениями:

-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.

– браузер — программное обеспечение, основной задачей которого является

просмотр веб-страниц, документов, медиафайлов и других типов данных;

– сервер — аппаратно-программный комплекс, задачей которого является

обработка запросов от клиентов — устройств или программ, запрашивающих

определенную информацию или создающие задачи для обработки;

– аккаунт — учетная запись пользователя в системе; как правило, набор

информации о конкретном пользователе веб-ресурса;

– стеганография — наука о скрытой передаче данных; совокупность методов

по скрытой передаче одних данных в содержании других;

– модуляция — процесс преобразования оригинального сигнала в сигнал с

другими параметрами, пригодными для определенных условий использования;

– помехоустойчивое кодирование — совокупность методов, применяемых

при передаче или хранении информации, обеспечивающих сохранность этой

информации за счет наличия избыточных данных, используемых для

восстановления потерянных данных;

– битрейт — условное обозначение скорости записи данных, обозначающее

количество данных в единицу времени, необходимых для кодирования

определенной части информации;

– мессенджер — приложение для обмена мгновенными сообщениями,

основной функцией которого является передача сообщений и файлов, в том числе медиаданных, сфокусированное на скорости работы и широкой доступности для использования в различных условиях.

-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.

## **ПЕРЕЧЕНЬ СОКРАЩЕНИЙ**

LSB – Least Significant Bit

RGB – Red Green Blue

JPEG – Joint Photographic Experts Group

BMP – Bitmap Picture

RS – Regular-Singular, то есть «регулярно-сингулярный».

TIFF – Tagged Image File Format

PNG – Portable Network Graphics (формат хранения растровых графических изображений)

TGA – Truevision TGA (TGA – растровый графический формат)

ДКП - Дискретное косинус-преобразование

ЦВЗ – цифровой водяной знак

БЧХ – коды Боуза-Чоудхури-Хоквингема (оставляют один из больших классов линейных кодов, исправляющих ошибки)

НЗБ – наименее значащие биты

ПО — Программное Обеспечение

СМИ — Средства Массовой Информации

ДПФ — Дискретное Преобразование Фурье

БДПФ — Быстрое Дискретное Преобразование Фурье

ДЭВ — Дифференциальное Встраивание Энергии

AM — Amplitude Modulation

FM — Frequency Modulation

GUI — Graphical User Interface

ID — Identification; identity

FSK — Frequency Shift Keying

## **ВВЕДЕНИЕ**

Стеганография – это наука о скрытой передаче информации путём сохранения в тайне самого факта передачи. В отличие от криптографии, которая скрывает только содержимое секретного сообщения, стеганография скрывает факт его существования. Стеганография обычно используется в дополнение к методам криптографии.

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.

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.

Основной областью применения компьютерной стеганографии является сокрытие сообщений в цифровых данных, как правило, имеющих аналоговую природу (изображения, аудио- и видеозаписи). Также в качестве контейнеров используют текстовые и исполняемые файлы. В данной работе информация скрывалась в графических контейнерах.

В настоящее время в связи с бурным развитием вычислительной техники и новых каналов передачи информации появились новые стеганографические методы, в основе которых лежат особенности представления информации в компьютерных файлах, вычислительных сетях и т. п. Это дает нам возможность говорить о становлении нового направления - компьютерной стеганографии.

Цель выпускной квалификационной работы – реализация некоторых алгоритмов скрытия данных в графических изображениях формата JPEG, исследование их на устойчивость к атакам и изменение заполненного контейнера.

Задача данной выпускной квалификационной работы заключается в ознакомлении с общими принципами стеганографических методов, реализация метода наименьшего значащего бита (LSB) для скрытия информации в графических изображениях формата JPEG.

Данный проект реализует следующие цели:

* Ознакомление с теорией стеганографии и стегоанализа
* Анализ различных методов стеганографического сокрытия данных
* Реализация метода скрытия в файле JPEG
* Анализ метода скрытия в файле JPEG
* Реализация метода скрытия данных методом LSB для JPEG
* Анализ метода скрытия данных методом LSB для JPEG на устойчивость к пассивным атакам и искажениям
* Анализ изменения параметров контейнера при скрытии в нем данных

Методы стеганографии применяются не только для скрытной передачи сообщений, но и используют для защиты авторских или имущественных прав на цифровое изображение, фотографии или другие оцифрованные произведения искусства. Преимущества, которые дают представление и передача сообщений в цифровом виде, могут оказаться перечеркнутыми легкостью, с которой возможно их воровство или модификация. Поэтому разрабатываются различные меры защиты информации, организационного и технического характера. Один из наиболее эффективных технических средств защиты мультимедийной информации заключается во встраивании в защищаемый объект невидимых меток – цифровых водяных знаков. Они могут содержать много полезной информации: когда создан файл, кто владеет авторскими правами, как вступить в контакт с автором и т.д. Все внесенные данные могут рассматриваться как веские доказательства при рассмотрении вопросов и судебных разбирательств об авторстве или для доказательства факта нелегального копирования и часто имеют решающее значение.

Наиболее распространенным на сегодняшний день методом цифровой стеганографии является метод, заключающейся во вложении скрываемого сообщения в изображение путем модификации наименее значимых бит (LSB). Цифровые изображения представляют собой матрицу пикселов. Пиксел – это единичный элемент изображения. Он имеет фиксированную разрядность двоичного представления.

Например, в простейшей черно-белой картинке каждый пиксель описывается одним байтом, который кодирует яркость пикселя: ноль — черный, 255 — белый, все остальное — градации серого. Если изменить любой байт такого файла или, что тоже самое, отдельные биты этого байта, то соответствующий ему пиксель изменит яркость. При этом изменение разных битов влияет на яркость пикселя по-разному: первый очень сильно, второй слабее, а последний, восьмой бит может добавить байту (а значит, и пикселю) только единицу. Нормальный человек не заметит изменение яркости точки на одну (1/255) градацию серого. А значит, абсолютно не важно, каковы последние биты каждого байта. И их можно обнулять, переставлять, заменять, — картинка при этом будет казаться одинаковой. [1]

Практическая значимость исследования заключается в разработке программного средства, готового для внедрения в существующие системы, выполняющие распространение контента по модели ограниченного доступа к данным, для защиты распространяемых изображений. Разработанный алгоритм легко расширяется до аналогичного, пригодного для использования в любых медиафайлах.

## **РАЗДЕЛ 1. ОБЩИЕ СВЕДЕНИЯ О СТЕГАНОГРАФИИ**

# 1.1 Современные проблемы стеганографии

Опишем сценарий использования разрабатываемого алгоритма для

обоснования практической значимости работы. Существуют компании, бизнес которых сосредоточен исключительно на предоставлении информационных услуг. Интернет для таких компаний является основным каналом связи с клиентом и основным каналом распространения своей продукции. Например, среди таких можно отметить электронные СМИ, форумы, блоги, образовательные ресурсы, новостные подписки и другие.

Для удобства, остановимся на примере образовательных ресурсов. Среди пользователей глобальной сети Интернет все более популярным становится явление сетевого обучения — способ, при котором вы сможете получить полезные прикладные умения или теоретические знания, не выходя из дома. Для этого требуется быть участником какого-либо образовательного ресурса, публикующего курсы по интересующим вас темам. Такие курсы создаются силами частных компаний и распространяются по модели платных подписок или единовременной оплате доступа к курсу.

Таким образом, авторы передают свои знания и опыт студентам в обмен на вознаграждение. Это очень хороший способ научиться чему-то новому, что не входит стандартные академические программы или появилось не так давно, а рынок уже требует специалистов соответствующего профиля. К примеру, это очень актуально в сфере веб-разработки — back-end и front-end технологии развиваются сейчас очень стремительно, не без причины широкого распространения облачных технологий, и появляются новые фреймворки, библиотеки, программное обеспечение, работе с которыми не научит ни один университет, а компаниям уже сейчас нужны специалисты, способные внедрять самые последние разработки.

Еще в качестве примера можно привести то, что существуют компании, предлагающие свои уникальные методики самостоятельного изучения английского языка посредством прохождения авторских онлайн-курсов. Это может быть полезно людям в удаленных уголках страны; людям, у которых нет достаточного количества времени для очного обучения в группах, и просто всем тем, кому такой способ получения информации кажется более комфортным и привлекательным.

На этой почве и появляются различные компании-поставщики образовательного контента, бизнес-модель которых строится на продаже доступа к курсам большому количеству студентов из разных уголков планеты.

Здесь и проявляется проблематика — защита авторского контента. В интересах компаний сделать так, чтобы курс, проданный одному студенту, был доступен именно этому студенту. Если же он распространяет материалы курса (например, видеоуроки) среди своих знакомых или просто среди всех желающих, то компания терпит убытки в виде упущенной выгоды, ведь все те, кто получил материалы курса от одного студента могли стать клиентами компании. Явление несанкционированного распространения контента, защищенного авторскими правами, называется компьютерным пиратством.

Компьютерное пиратство — незаконное распространение или использование материалов, защищенных авторским правом, без разрешения автора или с нарушением договора о использовании таких материалов. Такие действия являются правонарушением и преследуются по закону. Однако, нежелание некоторых пользователей платить за труд автора приводит к тому, что защищенные авторским правом материалы курсов становятся публично доступными неограниченному кругу лиц. Данная работа не приводит способы защиты интернет-ресурса от таких действий, но рассматривает способ сократить издержки через быструю локализацию источника утечки, а именно аккаунта пользователя, который стал распространять медиаданные купленного курса.

Чтобы понимать, каким образом медиаданные могут быть скопированы и распространены, следует немного углубиться в специфику клиент-серверного взаимодействия — то есть процесса взаимодействия устройства пользователя (студента) и сервера (онлайн-ресурса), где студенту предоставлен доступ к купленному образовательному курсу.

Таким образом, в настоящее время продолжается развитие методов стеганографии в связи с необходимостью решения следующих задач:

1) Защита авторских прав с помощью цифровых подписей и водяных знаков

В копию защищаемого файла внедряется дополнительная информация, позволяющая определить автора.

2) Скрытая передача данных

Передача данных со скрытием самого факта передачи.

3) Скрытое хранение данных

Данная задача схожа с предыдущей. Скрывается сам факт хранения информации. Для решения задачи чаще всего используется избыточность носителей информации.

4) Подтверждение подлинности документов

В данной задаче используются методы, схожие с используемыми в задаче защиты авторских прав. Настоящим считается документ, содержащий цифровую подпись.

5) Скрытая передача управляющего сигнала

Управляющие сигналы применяются для изменения состояния какого-либо устройства. При этом может возникнуть необходимость сокрытия факта изменения состояния. Для этого используются методы сокрытия передачи управляющего сигнала.

6) Подтверждение достоверности переданной информации

В этой задаче сообщение посылается с дополнительной информацией, подтверждающей подлинность/неизменность информации. Обычно это бывает хеш-функция или контрольная сумма.

7) Стеганографическое отслеживание утечек информации

Используется в случае необходимости отследить пути распространения определённого файла. Например, необходимость отслеживания может понадобиться при передаче сотрудникам какой-либо фирмы информации конкурентам. В этом случае разные копии данного помечаются разными знаками, что позволяет определить злоумышленника. [2]

Сама стеганосистема представляет собой систему, осуществляющую встраивание и выделение одной битовой последовательности из другой. Последовательность, подлежащая скрытию, называется сообщением. Последовательность, в которую осуществляется встраивание, называется контейнером. Если в контейнер не встраивалось сообщение, то он называется пустым, иначе – заполненным. Как правило, в составе стеганосистемы дополнительно выделяют подсистемы, такие как прекодер, стеганокодер, стеганодетектор, декодер [3]. Сравнительно недавно была разработана математическая модель стеганосистемы.

В любой стеганосистеме важную роль играет стеганографический протокол – порядок действий, к которым прибегают две или более сторон, с целью решения определенных задач [4].

Цифровой водяной знак (ЦВЗ) – внедренная в мультимедийный сигнал информация, назначение которой – аутентификация содержимого, охрана прав собственника, защита от копирования и т.п.

Стеганосистема образует стеганоканал, по которому передается заполненный контейнер. Этот канал считается подверженным воздействиям со стороны нарушителей. Следуя [8], в стеганографии обычно рассматривается постановка задачи в виде «проблема заключенных», желающих тайно обмениваться сообщениями посредством передачи их в скрытом. Пассивный нарушитель может лишь обнаружить факт наличия стеганоканала и (возможно) читать сообщения. Диапазон действий активного нарушителя значительно шире. Скрытое сообщение может быть им удалено или разрушено. В этом случае передающая и, возможно, принимающая сторона узнают о факте вмешательства. Действия злоумышленного нарушителя наиболее опасны. Он способен не только разрушать, но и создавать ложные сообщения.

При построении стеганосистемы должны учитываться следующие положения, многие из которых лежат в основе критериев эффективности стеганографических алгоритмов изображений:

* стеганосистема должна иметь приемлемую вычислительную сложность реализации;
* заполненный контейнер должен быть визуально неотличим от незаполненного;
* должна обеспечиваться необходимая пропускная способность (что особенно актуально для стеганосистем скрытой передачи данных);
* методы скрытия должны обеспечивать аутентичность и целостность секретной информации для авторизованного лица;
* потенциальный нарушитель имеет полное представление о стеганосистеме и детали её реализации, единственное, что ему неизвестно, – это ключ, с помощью которого только его обладатель может установить факт наличия и содержание скрытого сообщения;
* если факт существования скрытого сообщения становится известным нарушителю, это не должно позволить последнему извлечь его до тех пор, пока ключ сохраняется в тайне;
* нарушитель должен быть лишен любых технических и других преимуществ в распознании или, по крайней мере, раскрытии содержания секретных сообщений [3].

# 1.2 Клиент-серверное взаимодействие

Клиент — аппаратный или программный компонент вычислительной системы, посылающий запросы к серверу. Клиент может запрашивать какие-либо данные с сервера, манипулировать ими непосредственно на сервере, запускать на нем новые процессы. Полученные данные клиент может отображать пользователю или использовать иным образом, в зависимости от его предназначения.

Сервер — программный компонент вычислительной системы, выполняющие сервисные (обслуживающие) функции по запросу клиента. Предоставляет доступ к определенным ресурсам или выполняет вычисления.

В нашем случае сервером можно считать:

– само физическое устройство, на которое в конечном счете через интернет

приходит запрос от клиента;

– веб-сервер, как программное обеспечение, все время запущенное и

ожидающее обращения от клиента, чтобы сразу же обработать запрос.

Клиентом в таком случае выступает:

– сам пользователь, как клиент компании;

– компьютер или мобильное устройство пользователя, как физическое

устройство;

– браузер пользователя, как программа, непосредственно формирующая

запрос на сервер и обрабатывающая его ответ.

Браузер — прикладное программное обеспечение для просмотра веб- страниц, содержания веб-документов, компьютерных файлов и их каталогов; управления веб-приложениями; а также для решения других задач.

Пользователь может не только просмотреть контент курса в своем браузере, но и сохранить его себе, чтобы затем распространять его на других, не подконтрольных компании площадках. Будем считать, что пользователь уже оплатил курс и получил необходимые данные для доступа к нему. Эти данные он указывает в своем браузере, происходит т.н. процесс «аутентификации» — проверка подлинности предоставленных пользователем данных для его авторизации.

Авторизация — это получение прав доступа к ресурсам. Таким образом, теперь пользователь может в своем браузере открывать страницы курса и относящихся к нему уроков, а также просматривать весь необходимый контент.

При получении непосредственно контента взаимодействие складывается следующим образом:

1) пользователь открывает в браузере страницу конкретного урока какого-

либо образовательного курса;

2) браузер получает html-страницу урока в качестве ответа на запрос;

3) страница содержит в себе сам текст урока и множество ссылок на

контент: изображения, видео, аудио;

4) по найденным ссылкам браузер автоматически снова посылает запрос на

сервер для получения самих файлов;

5) производится загрузка файлов и их отображение на странице, в

зависимости от типа файла; например, изображение отображаются «как есть», а файлы встраиваются в «плеер», который позволяет управлять процессом

воспроизведения видео.

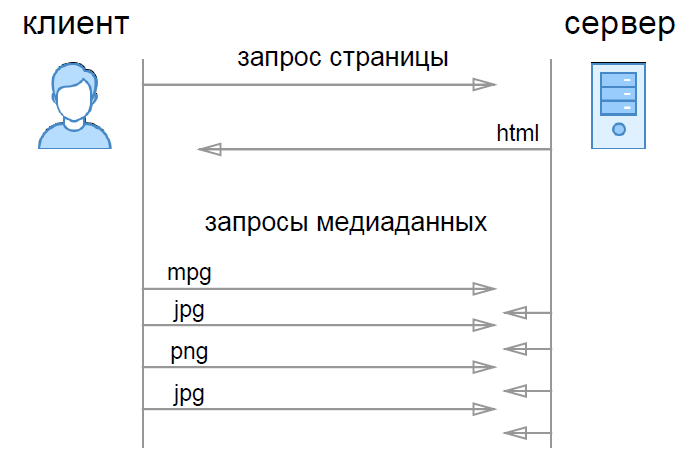


Рисунок 1. Схема клиент-серверного взаимодействия при запросе страницы, содержащей медиаданные.

Данные могут представляться в качестве любой информации: текст, сообщение, изображение и т.п. Информация, предназначенная для сокрытия тайных сообщений, называется контейнером. В данной работе контейнер представляет собой изображение. Стегоконтейнером является изображение с уже встроенным сообщением.

Если пользователь немного технически подкован в этом вопросе, то понимает, что эти файлы, отображаемые на странице браузера, он может сохранить отдельно и затем распространять их. Особенно остро эта проблема проявляется для курсов, построенных в формате видео уроков.

Если каждая копия медиаданных каким-либо образом помечена и однозначно указывает на того пользователя, кому она была предназначена, то не составляет труда, пользуясь знанием о том, как эту метку извлечь, найти этого пользователя в системе образовательного ресурса и ограничить ему доступ к курсам, а также получить о нем дополнительную информацию, которая может быть полезна правоохранительным органам.

Если вспомнить, каким образом контент попадает на компьютер пользователя и отображается в браузере, мы приходим к выводу, что метка должна быть непосредственно в загружаемых файлах.

Это возможно, если использовать принципы цифровой стеганографии.

# 1.3 Цифровая стеганография

Формальное определение стеганографии гласит о том, что это наука о

скрытой передаче информации путём сохранения в тайне самого факта передачи. Развитие цифровых технологий способствовало развитию специального отделения этой науки — цифровой стеганографии, в рамках которого исследуются способы записи одних битовых последовательностей в другие, как правило, имеющие аналоговую природу. Т.е. необходимые данные записываются в оцифрованный непрерывный сигнал, который могут представлять собой разного рода медиаданные: речь, аудиозаписи, изображения, видео. [3]

По области применения ЦС можно классифицировать следующим образом:

– встраивание информации с целью ее скрытой передачи;

– встраивание цифровых водяных знаков (ЦВЗ) (watermarking);

– встраивание идентификационных номеров (fingerprinting);

– встраивание заголовков (captioning).

Стеганографическое преобразование для ЦВЗ и для идентификационных номеров во многом похоже, с единственной разницей в том, что во втором случае для каждой конкретной копии медиаданных для конкретного пользователя используется индивидуальный ЦВЗ, позволяющий идентифицировать пользователя после выделения ЦВЗ.

Сам ЦВЗ представляет собой некоторое скрываемое сообщение (последовательность бит), которое записывается в контейнер (например, статичное изображение или видеоряд). Стеганографическое преобразование должно быть выполнено таким образом, чтобы исходный и помеченный сигналы были неотличимы по определенным критериям восприятия, т.е. пустой и заполненный контейнеры в некотором роде были равны друг другу.

Основное требование при этом - контейнер-результат не должен быть визуально отличим от контейнера-оригинала. Контейнер может быть фиксированным и потоковым. Примером фиксированного является статичное изображение, размеры которого и, как следствие, потенциал для хранения сообщений нам заранее известны. Потоковый контейнер представляет собой непрерывно следующую последовательность бит — это, например, видеоряд или аудиозапись. Сообщение в такой контейнер вкладывается в реальном масштабе времени, поэтому заранее неизвестно, хватит ли кодеру места в контейнере, чтобы вместить нужные сообщения.

Кроме этого существует еще одна сложность, заключающаяся в синхронизации записи и чтения сообщений. Решается это при помощи использования заголовков или битов синхронизации. В фиксированном контейнере размеры и характеристики последнего заранее известны. Это позволяет выполнять вложение данных оптимальным (в определенном смысле) образом. Далее будем рассматривать фиксированные контейнеры.

Упаковка сообщения в контейнер выполняется с помощью стеганокодера. Вложение происходит, например, путем модификации наименьших значащих битов контейнера.

Вообще, именно алгоритм (стратегия) внесения элементов сообщения в контейнер определяет методы стеганографии, которые в свою очередь делятся на определенные группы, например, в зависимости от того, файл какого формата был выбран в качестве контейнера.

В большинстве стеганосистем для упаковки и извлечения сообщений используется ключ, который предопределяет секретный алгоритм, определяющий порядок внесения сообщения в контейнер.

Встраивание и выделение сообщений из медиаданных осуществляет стегосистема, структурная схема которой описана на рисунке 2. [7]



Рисунок 2. — Схема работы стегосистемы.

По типу ЦВЗ могут быть хрупкими, полухрупкими и робастными. Робастность — устойчивость ЦВЗ к разного рода помехам и искажениям. В хрупких ЦВЗ сообщение разрушается при незначительной модификации контейнера. Это бывает полезно для применения аутентификации сигналов, т.е. чтобы однозначно определить подлинность сигнала. Полухрупкие ЦВЗ разрешают одни виды изменений и запрещают другие. Например, для изображений может быть разрешено сжатие, но замена фрагментов уже повлечет разрушение ЦВЗ.

# 1.4 Постановка цели

Таким образом, защита авторского контента при помощи применения цифровой стеганографии возможна. Достигается это следованием определенной последовательности действий:

1) генерация индивидуального для каждого пользователя ЦВЗ («fingerprint»,

отпечаток пальца);

2) встраивание ЦВЗ в каждую копию медиаданных, запрашиваемых

пользователем (например, из веб-интерфейса платформы обучения);

3) встраивание производится либо в заранее подготовленные копии, либо в

режиме реального времени — сервер производит транскодирование

медиаданных «на лету».

Принцип защиты срабатывает в тот момент, когда пользователь производит нелегальное сохранение авторского контента и последующее его распространение. Встроенные в клиентские копии контента водяные знаки позволяют идентифицировать его и принять сопутствующие меры.

Реализацию идеи защиты можно разделить на несколько крупных блоков:

– платформа для обучения (она же — система доставки контента; сайт, на

котором непосредственно находятся студенты во время изучения курсов);

– подсистема защиты медиаданных, отсылаемых на клиентскую сторону (т.е.

студенту, просматривающему в данный момент курс из браузера);

– подсистема определения источника «утечки» — выявление метки из

медиафайлов и ее декодирование.

Сама по себе работа заключается в разработке алгоритма, который сможет выполнить возложенные на него задачи по скрытой записи данных.

## **РАЗДЕЛ 2. КЛАССИФИКАЦИЯ МЕТОДОВ**

# 2.1 Классификация методов сокрытия информации

Одними из самых распространенных типов контейнеров в компьютерной стеганографии на данный момент являются изображения, аудиоданные, представленные в цифровой форме, и видео-последовательности. Это объясняется тем, что подобные контейнеры имеют шумовую составляющую, обусловленную их образованием, которая способна замаскировать встраиваемое сообщение.

Все методы, предназначенные для сокрытия данных, можно разделить по принципам, лежащим в их основе:

1) Форматные методы сокрытия ‒ это такие методы, которые основываются на особенностях формата хранения графических данных. Разработка таких методов сводится к анализу формата с целью поиска полей формата, изменение которых в конкретных условиях не скажется на работе с графическим изображением.

2) Неформатные методы ‒ это методы, использующие непосредственно сами данные, которыми изображение представлено в этом формате. Применение неформатных методов неизбежно приводит к появлению искажений, вносимых стеганографической системой, однако при этом они являются более стойкими к атакам как пассивных, так и активных противников.

Рассмотрим классификацию методов более подробно.

1. Неформатные методы сокрытия

1.1. Неформатные методы сокрытия в JPEG

1.1.1. Метод сокрытия в исходных данных изображения

1.1.2. Метод сокрытия с использованием таблиц квантования

1.1.3. Метод использования ложных таблиц квантования

1.1.4. Метод сокрытия в спектре изображения после квантования

1.2. Методы сокрытия в графических изображениях с палитрой цветов

1.3. Метод сокрытия с использованием младших битов данных изображения

1.4. Метод сокрытия с использованием младших битов элементов палитры

1.5. Метод сокрытия, основанный на наличии одинаковых элементов палитры

1.6. Метод сокрытия путём перестановки элементов палитры

2. Форматные методы сокрытия в графических изображениях

2.1. Форматные методы сокрытия в файлах BMP

2.2. Форматные методы сокрытия в JPEG

2.2.1. Дописывание данных в конец JPEG файла

2.2.2. Метод сокрытия в косвенных данных

2.2.3. Метод сокрытия с использованием маркеров комментариев

2.2.4. Метод сокрытия с использованием уменьшенного изображения.

# 2.2 Обзор методов цифровой стеганографии

Для выбора подходящего метода для дальнейшей реализации алгоритма

проведем сравнительный анализ существующих методов цифровой стеганографии.

Метод Куттера – Джордана – Боссена, состоит в том, что бит сообщения, закодированного данным методом, встраивается в канал синего цвета путем модификации яркости выбранного пикселя в соответствии с формулой (1.1).

, (1.1)

где (1.2)

Извлечение происходит по следующей формуле (1.3):

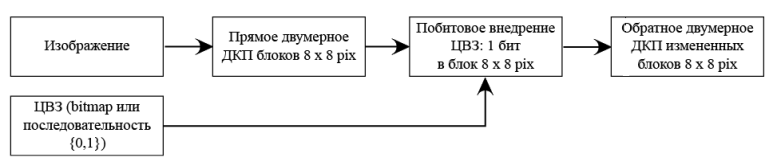
(1.3)

где .

Главная проблема данного метода – вероятностный характер извлечения яркости цвета.

**Метод Коха и Жао**

Этот метод на сегодняшний день является достаточно востребованным. Схема внедрения данных по этому методу показана на рисунке 3.

Рисунок 3. Схема внедрения цифровых водяных знаков в изображение

Водяные знаки, предназначенные для защитной маркировки изображений, могут представлять собой как монохромное (bitmap) изображение, так и числовую последовательность {0,1} произвольной длины. Внедрение бит ЦВЗ по методу Коха и Жао осуществляется в блоки пикселей растрового изображения, которое может быть сохранено в любом из рассмотренных в первой главе графических форматах, а также в любой из рассмотренных цветовых моделей.

Защищаемое растровое изображение разбивается на N блоков (матриц) fa размером 8 х 8 пикселей, которые подвергаются двумерному ДКП. При этом каждая целочисленная матрица пикселей изображения преобразуется в матрицу частотных коэффициентов. Перед началом маркировки необходимо оценить соответствие размеров внедряемого ЦВЗ количеству полученных блоков изображения. Далее в полученные частотные матрицы DCTa производят побитовое встраивание водяных знаков, при этом каждый бит водяного знака внедряется в блок 8 х 8 пикселей в результате относительной замены двух или трех элементов матрицы DCTa.

**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 кодировании нуля и единицы. Обе эти задержки должны быть меньше, чем порог чувствительности уха слушателя к получаемому эху.

# 2.3 Методы встраивания в пространственные области изображений

**Метод Kutter**

Пусть изображение имеет RGB-кодировку. Встраивание выполняется в канал синего цвета, так как к синему цвету система человеческого зрения наименее чувствительна. Пусть si – встраиваемый бит, I = {R,G,B} – контейнер, *p = (x,y)* – псевдослучайная позиция, в которой выполняется вложение. Секретный бит встраивается в канал синего цвета путем модификации яркости

(2.9)

(2.10)

где q – константа, определяющая энергию встраиваемого сигнала. Ее величина зависит от предназначения схемы. Чем больше q, тем выше робастность вложения, но тем сильнее его заметность. Максимальное отклонение синей цветовой составляющей при условии неизменности двух других цветов составляет 9–26%. Цветовая компонента каждого пикселя описывается одним байтом. Изменение происходит по маске 11100011, то есть модификации подлежат 4, 5 или 6 биты. Отклонение интенсивности цвета в данном случае не превышает 6,3%, а общее изменение яркости пикселя не превышает 1% [6].

**Метод Bruyndonckx**

ЦВЗ представляет собой строку бит. Для повышения помехоустойчивости применяется код БЧХ. Внедрение осуществляется за счет модификации яркости блока 8×8 пикселов. Процесс встраивания осуществляется в три этапа:

 классификация, или разделение пикселов внутри блока на две группы с примерно однородными яркостями.

 разбиение каждой группы на категории. Для этого на блоки накладываются маски, разные для каждой группы и каждого блока.

Назначение масок состоит в обеспечении секретности внедрения

 модификация средних значений яркости каждой категории в каждой группе.

**Метод Langelaar**

Алгоритм работает с блоками 8×8. Вначале создается псевдослучайная маска нулей и единиц такого же размера pat (x,y) {0,1}. Далее каждый блок B делится на два субблока B0 и B1, в зависимости от значения маски. Для каждого субблока вычисляется среднее значение яркости l0 и l1. Далее выбирается некоторый порог α, и бит ЦВЗ встраивается следующим образом:

(3.1)

Если это условие не выполняется, мы изменяем значение яркости пикселов субблока B1. Для извлечения бита ЦВЗ вычисляются средние значения яркости субблоков – и . Разница между ними позволяет определить искомый бит:

(3.2)

**Метод Rongen**

Также, как и в предыдущем алгоритме, ЦВЗ представляет собой двумерную матрицу единиц и нулей с примерно равным их количеством. Пиксели, в которые можно внедрять единицы (то есть робастные к искажениям), определяются на основе некоторой характеристической функции (характеристические пиксели). Эта функция вычисляется локально, на основе анализа соседних пикселов. Характеристические пиксели составляют примерно от общего числа, так что не все единицы ЦВЗ встраиваются именно в эти позиции. Для повышения количества характеристических пикселов в случае необходимости предлагается осуществлять небольшое предыскажение изображения. Детектор находит значения характеристических пикселов и сравнивает с имеющимся у него ЦВЗ. Если в изображении ЦВЗ не содержится, то в характеристических пикселах количество единиц и нулей будет примерно поровну.

**Метод Patchwork**

В основе алгоритма Patchwork лежит статистический подход. Вначале псевдослучайным образом на основе ключа выбираются два пикселя изображения. Затем значение яркости одного из них увеличивается на некоторое значение (от 1 до 5), значение яркости другого уменьшается на то же значение. Далее этот процесс повторяется большое число раз (~10000) и находится сумма значений всех разностей. По значению этой суммы судят о наличии или отсутствии ЦВЗ в изображении. Авторами также предложены улучшения основного алгоритма для повышения его робастности. Вместо отдельных пикселов предлагается использовать блоки, или patches. Отсюда и название алгоритма. Алгоритм Patchwork является достаточно стойким к операциям сжатия изображения, его усечения, изменения контрастности. Основным недостатком алгоритма является его неустойчивость к аффинным преобразованиям, то есть поворотам, сдвигу, масштабированию. Другой недостаток заключается в малой пропускной способности. Так, в базовой версии алгоритма для передачи 1 бита скрытого сообщения требуется 20000 пикселов.

**Метод Bender**

Алгоритм, основанный на копировании блоков из случайно выбранной текстурной области в другую, имеющую сходные статистические характеристики. Это приводит к появлению в изображении полностью одинаковых блоков. Эти блоки могут быть обнаружены следующим образом:

 анализ функции автокорреляции стегоизображения и нахождение ее пиков;

 сдвиг изображения в соответствии с этими пиками и вычитание изображения из его сдвинутой копии;

 разница в местоположениях копированных блоков должна быть близка к нулю. Поэтому можно выбрать некоторый порог и значения, меньшие этого порога по абсолютной величине, считать искомыми блоками.

Так как копии блоков идентичны, то они изменяются одинаково при преобразованиях всего изображения.

Если сделать размер блоков достаточно большим, то алгоритм будет устойчивым по отношению к большинству из негеометрических искажений. В проведенных экспериментах показана робастность алгоритма к фильтрации, сжатию, поворотам изображения. Основным недостатком алгоритма является исключительная сложность нахождения областей, блоки из которых могут быть заменены без заметного ухудшения качества изображения. Кроме того, в данном алгоритме в качестве контейнера могут использоваться только достаточно текстурные изображения.

Сравнение некоторых других методов приводится из работы Кохановича [1]:



Рисунок 6 — Сравнение стеганографических методов.

Некоторые методы сложны в реализации и выполняются достаточно долго, т.к. обеспечивают высокую скрытность встраивания, некоторые не подходят по другим ранее определенным требованиям. Например, необходимо наличие оригинального изображения или объем встраиваемых данных слишком мал.

## **РАЗДЕЛ 3. ОБОСНОВАНИЕ ВЫБРАННОГО МЕТОДА**

# 3.1 Метод встраивания в область ДПФ

Дискретное преобразование Фурье (ДПФ) — одно из фундаментальных понятий в области обработки сигналов вообще и цифровой обработки изображений в частности.

Прямое преобразование Фурье (Фурье-образ) *F(u)* непрерывной функции

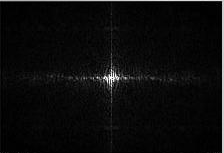
одной переменной *f(x)* определяется равенством (2.1)

где *i* — мнимая единица. По заданному Фурье-преобразованию *F(u)* можно получить исходную функцию *f(x)* при помощи обратного преобразования Фурье: (2.2)

Но в данном случае нас интересуют дискретные функции. Фурье- преобразование дискретной функции одной переменной *f(x), x = 0, 1, 2 ,…, M-1*, задается равенством: (2.3)

Это прямое дискретное преобразование Фурье (ДПФ). Аналогично, можно восстановить исходную функцию при помощи обратного ДПФ.

Для изображений, как для двумерных массивов данных, актуально использование двухмерного ДПФ, т.е. выполнение ДПФ по строкам, а затем выполнение ДПФ по результирующим столбцам. При этом желательно предварительно затем выполнить центрирование образа. В результате, практически вся энергия концентрируется в области низких частот (визуально — центр), что полезно при высокочастотной фильтрации изображений.



(а) (б)

Рисунок 4. — Изображение (а) и его Фурье-образ (б).

Спектр Фурье представлен после применения логарифмического преобразования. Запись ЦВЗ методом ДПФ предполагает работу в частотном спектре изображения и модификацию значений таким образом, чтобы при обратном преобразовании они не привели к заметным искажениям. Сложность такого метода заключается в том, что в Фурье-образ возможно записать только полу-хрупкий или робастный ЦВЗ. При попытке записи хрупкого ЦВЗ обратное преобразование с последующим округлением значений может привести к небольшим, но достаточным изменениям ЦВЗ, что повлечет разрушение неробастной метки.

Преимуществом метода является возможность использования Быстрого

Дискретного Преобразования Фурье (БДПФ). Это очень быстрый алгоритм вычисления ДПФ, выполняющийся за O (N log(N)).

Это возможно благодаря «прореживанию» — делению процесса ДПФ на более мелкие операции. Ограничением при этом является размер ряда, по которому выполняется преобразование. Размер должен быть кратным степени двойки 2k, где k — целое число. Однако, в случае невыполнения этого условия можно дополнить ряд нулями, а затем учесть это в результате.

Работа в спектральной области изображения позволяет гибко управлять влиянием метки на результирующее изображение и записывать робастные ЦВЗ в области высоких частот, т.к. в этой области концентрируется минимум энергии (<10%). Это, в свою очередь, означает то, что для человека визуально оригинал и результат не будут отличаться — основная информация об изображении концентрируется в области низких частот (максимум энергии).

# 3.2 Алгоритм встраивания ЦВЗ

Был предложен следующий алгоритм встраивания ЦВЗ:

1) прямое преобразование Фурье для изображения;

2) встраивание сформированной метки (ЦВЗ) в область высоких частот спектральной области изображения;

3) обратное преобразование Фурье.

Такой несложный алгоритм гарантирует быструю скорость работы ПО,

обеспечивающего запись ЦВЗ в режиме реального времени.

Дополнительно для реализации этого алгоритма требуется сформировать метку (ЦВЗ), пригодную для использования в спектральной области и устойчивую к помехам и разного рода атакам, в том числе сжатию в формате JPEG.

Дополнительно стоит отметить ограничения, накладываемые этим методом. Прежде всего, в нашем распоряжении есть не более 80% контейнера для вложения скрываемого сообщения, т.к. остальные 20% содержать максимум энергии, достаточной для отрисовки изображения с минимумом визуальных помех. Образ Фурье представляет из себя массив комплексных чисел, содержащий в себе информацию об амплитудном и фазовом спектрах. Как показывает практика, фазовый спектр несет очень мало информации о конечном виде изображения и может быть легко изменен при модификации контейнера, соответственно, для записи ЦВЗ мы можем использовать только амплитудный, чтобы гарантировать робастность записи. Таким образом, остается не более 30-40% исходной двумерной площади для записи знака. Соответственно, знак должен быть компактным, но при этом сохранять свойство устойчивости к помехам.

Накладываются также и ограничения, касающиеся размера изображения, пригодного для записи ЦВЗ в него. Исходя из размера метки минимальный изображения может меняться.

Для удобства здесь и далее будем рассматривать в качестве ЦВЗ 32-битное число, представляющее собой ID пользователя в системе, которому предназначалась данная копия медиаконтента.

Существует прямая взаимосвязь между размером секретного сообщения и размером метки (ЦВЗ), которой оно представлено пригодным для записи в контейнер.

Определим минимальный размер контейнера через определенные ранее

условия и информацию о размере метки:

*(m x n) \* 30% < Nцвз* , (2.4)

где *m* и *n* — размерность изображения, а *Nцвз* — размер метки.

**3.3 Алгоритм формирования метки**

32 бит достаточно для записи числа, представляющего собой более 4 млрд. Если использовать только 16 бит из предложенных 32, то можно записать максимальное число 65535, что зачастую для образовательных онлайн-ресурсов является достаточным в качестве обозначения идентификатора пользователя. Оставшиеся 16 бит можно использоваться для помехоустойчивого кодирования, что будет полезно в условиях требования устойчивости к внешним изменениям.

Т.к. изображение — это набор пикселей, каждый из которых кодируется по 1 байт на канал, то в этом случае мы имеем дело с дискретными значениями, причем их набор достаточно ограничен. В случае работы с Фурье-образом, мы имеем дело с комплексными числами, что требует точность вычислений сравнимую с вычислениями значений с плавающей запятой. Однако, при обратном преобразовании обязательно пропадают некоторые незначительные данные в процессе округления чисел с плавающей запятой. Соответственно, из-за проблем округления сложно гарантировать, что прямое преобразование Фурье, подвергшееся затем обратному преобразованию в изображение и снова прямому будет абсолютно идентично.

Таким образом, подразумеваем, что встроенная в спектральную область

метка гарантированно будет модифицирована даже обратным преобразованием в изображение, т.к. это обусловлено таким частным случаем преобразования. При этом не проводилось никаких изменений формата изображения или предумышленных атак на стего.

Далее следует говорить о ЦВЗ и контейнере для ЦВЗ в терминах сигналов, т.к. изображение — это сигнал аналоговой природы для глаза человека и работаем мы с этим сигналом ровно так же, как с радиосигналом, сигналом от эхолокатора или любым другим, где преобразования Фурье так же являются одним из основных инструментов для анализа и фильтрации данных.

Итак, спектральная область изображения представляет собой набор коэффициентов, характеризующих амплитудные и фазовые характеристики сигнала. Для разнородного изображения (не монотонного) образ представляет собой сигнал, похожий, в том числе статистически, на случайные данные.

Таким образом для записываемого ЦВЗ такую среду, в которую мы его

помещаем, можно трактовать, как среда с высокочастотным шумом. Устойчивым к таким условиям является периодический сигнал с правильно подобранными параметрами амплитуды и частоты. Выбор амплитуды важен для того, чтобы сильно не повредить изображение и не проявить метку при обратном преобразовании.

Выбор частоты позволяет сделать сигнал устойчивым к прямым/обратным преобразованиям Фурье. Как правило, сигнал высокой частоты проявляет себя лучше всего. Примером тому может являться множество технологий беспроводной передачи данных, начиная от цифрового радио, заканчивая 3G, LTE, WiMAX сетями, где частота измеряется мегагерцами.

Однако, в наших условиях мы очень ограничены размером контейнера, а потому для очень высокочастотного сигнала его попросту не хватит, т.к. здесь мы сталкиваемся с еще одним фундаментальным понятиям в области обработки сигналов — теоремой Котельникова. Она гласит о том, что любой аналоговый сигнал можно восстановить из дискретного, если частота дискретизации вдвое больше максимальной частоты сигнала (2.5).

f > 2fc (2.5)

# 3.4 Дискретное косинус-преобразование

Известно, что пиксели на изображении коррелируют со своими соседями, потому что значение конкретного пикселя можно предсказать по его соседям. Дискретное косинусное преобразование (ДКП) уменьшает эту избыточность между пикселями [12]. Оно преобразует исходную матрицу данных в матрицу некоррелированных величин (2.6), используя суммы косинусов на разных частотах. ДКП имеет обратное преобразование (2.7).

где – исходная матрица, – матрица коэффициентов ДКП, – размер матриц и ,

, (2.8)

*=* , или  *=* .

Преобразование происходит таким образом, что коэффициенты матрицы ДКП получаются упорядоченными по частоте. Сначала следуют низкочастотные коэффициенты, затем среднечастотные и высокочастотные (рис.5). Низкочастотные коэффициенты содержат самую важную информацию для восстановления исходных данных, и их изменение приведёт к сильному искажению данных после применения обратного преобразования. Высокочастотные коэффициенты можно отрезать (занулить) без сильного воздействия на данные после применения обратного преобразования, что и происходит на этапе квантования.

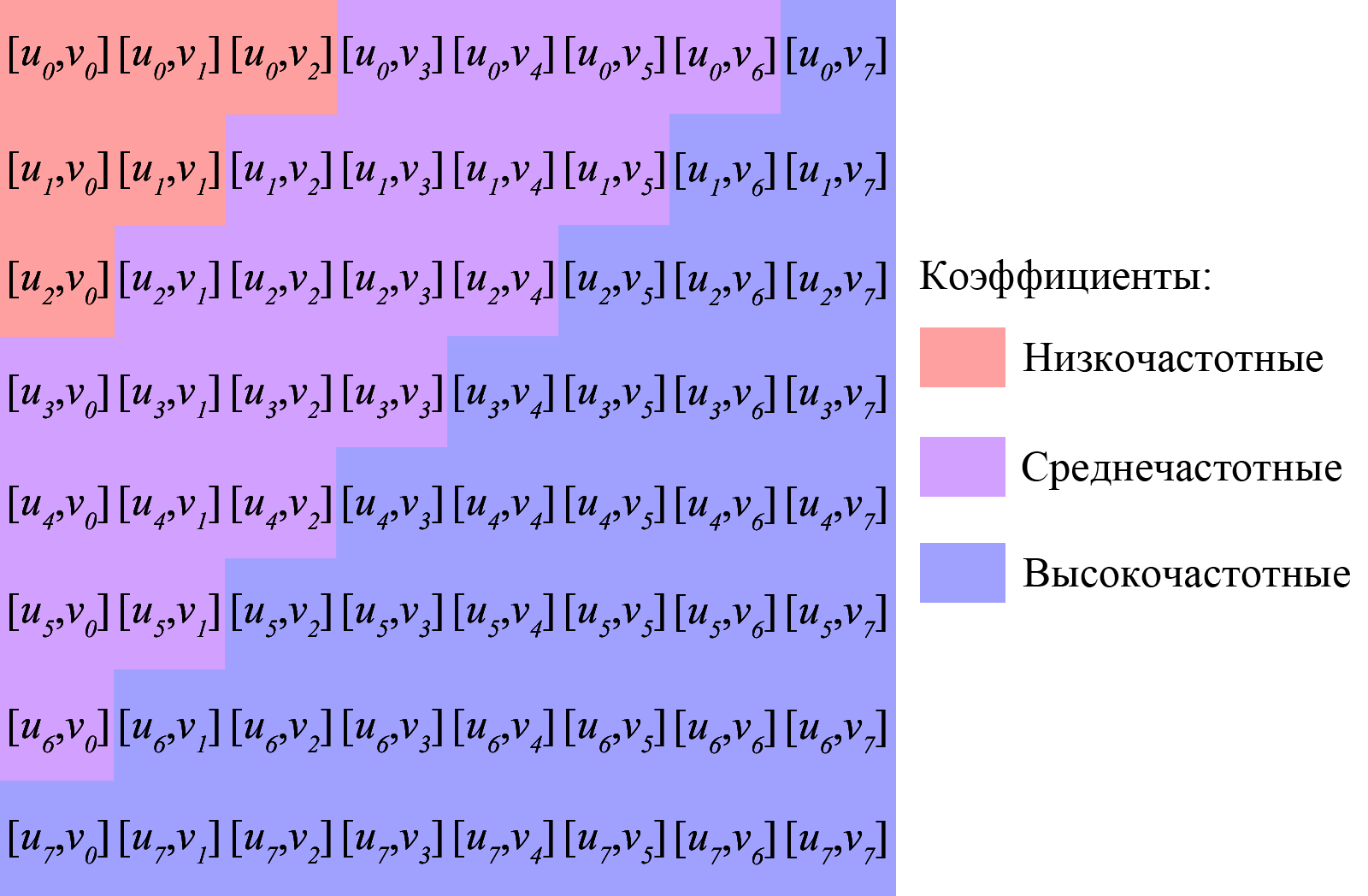


Рисунок 5. Матрица коэффициентов ДКП размером 8х8

В модели сжатия JPEG в качестве данных для матрицы берутся числовые значения компонент цветового пространства . Для этого компоненты , и разбивается на блоки и затем к каждому из блоков применяется ДКП.

# 3.5 Least Significant Bits (LSB)

Пространственный метод, эксплуатирующий специфику такого приемника сигнала, как глаз человека. Суть метода заключается в том, что встраивание секретной информации производится в младшие значащие биты пикселя — как правило последний один или два. Таким образом, интенсивность цвета данного пикселя меняется незначительно — достаточно для того, чтобы записать часть секретной информации и сохранить общий вид изображения очень близким к оригиналу.

Кодирование изображений для веб-ресурсов производится по модели RGB, т.е. каждый пиксель представляет собой 3-компонентную структуру, описывающую значение интенсивности для каждого из трех базовых цветов — красный, зеленый и синий. [8] При этом значение интенсивности для каждого компонента кодируется 1 байтом (8 бит), это значение от 0 до 255 включительно. Примеры кодирования пикселя:

– #000000 — белый пиксель;

– #FFFFFF — черный пиксель;

– #FF0000 — красный пиксель;

– #00FFFF — желтый пиксель (максимальная интенсивность зеленого и

синего каналов).

Чаще всего для метода LSB выбирают синий канал т.к. к изменению интенсивности этого цвета глаз человека наименее восприимчив. Конечно, можно использовать и монохромные изображения, пользуясь единственным

представленным каналом. Для записи сообщения в контейнер сначала определяют стегопуть — массив пикселей, которые будут модифицированы, а затем модифицируют последний бит пикселей в стегопуте в соответствии с очередным кодируемым битом сообщения.

Таким образом, если мы кодируем 3-ий бит из сообщения 0110010 в синий канал пикселя #FAFAFA, то значение синего канала будет модифицировано следующим образом:

– текущее значение: FA;

– бинарное представление: 1111 1010;

– запись 3-го бита, т.е. «1» в младший значащий бит: 1111 1011.

Результирующее значение цвета пикселя: #FAFAFB

Такой метод отлично подходит в тех случаях, когда заранее известно, что стего не будет подвергаться изменениям, т.к. он хрупкий. При этом можно записать очень большое количество информации — буквально еще одно изображение внутрь исходного изображения.

В этом случае ЦВЗ оказался хрупким и не подлежащим восстановлению даже при небольших изменениях, что не подходит для нашей ситуации. Это ни в коем случае не означает, что данный алгоритм не подходит для цифровой стеганографии вообще и не может нигде использоваться. Напротив, иногда к системе предъявляются требования, согласно которым нужно подтверждать оригинальность контента. Это возможно путем встраивания водяного знака (ЦВЗ, не путать с индивидуальным), который рассыпается при малейшем на него воздействии.

Преимущества метода:

 размер файла-контейнера остается неизменным;

 при замене одного бита в канале синего цвета внедрение невозможно заметить визуально;

 возможность варьировать пропускную способность, изменяя количество заменяемых бит.

Недостатки метода:

 Скрытое сообщение легко разрушить, например, при сжатии или отображении.

 Не обеспечена секретность встраивания информации. Точно известно местоположение зашифрованной информации. Для преодоления этого недостатка можно встраивать информацию не во все пикселы изображения, а лишь в некоторые из них, определяемые по псевдослучайному закону в соответствии с ключом, известному только законному пользователю. Пропускная способность при этом уменьшается. [6]

# 3.6 Обоснование выбранного формата.

Метод скрытия данных, основанный на характерных особенностях форматов файлов широко распространен, прост в реализации, но очень легко раскрывается простейшими средствами стегоанализа, а если следовать принципу, согласно которому злоумышленник знает каким методом было спрятано сообщение, то нахождение факта передачи и даже извлечение передаваемой информации не представляет из себя сложной проблемы, что и будет показано на примере формата JPEG.

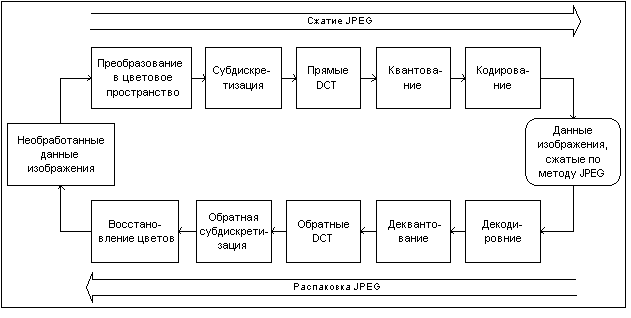


Рисунок 7. Структура JPEG - преобразований

Его характерная особенность заключается в наличии маркера конца кодируемой части изображения, последовательности байтов FF D9, если добавить в файл информацию после этого маркера, то изображение будет отображаться без изменений по сравнению с оригиналом, при этом вносимые данные будут являться частью файла и их можно будет передавать.

Метод замены наименее значащего бита (НЗБ, LSB - Least Significant Bit) – один из самых распространенных методов включения секретных данных в пространственной области файла. Метод основан на том, что последний бит в цветовых компонентах несет в себе меньше всего информации, и человек в большинстве случаев не способен заметить изменений в этом бите.

# 3.7 Математическая модель цифрового изображения JPEG

JPEG является широко используемым методом сжатия с потерями для цифровых изображений, особенно для полученных с помощью цифровой фотографии. Степень сжатия можно регулировать, что позволяет находить компромисс между размером хранения и качеством изображения. JPEG позволяет достигать сжатие 10:1 с небольшой ощущаемой потери качества изображения. Термин "JPEG" является аббревиатурой для Joint Photographic Experts Group, организацией, которая создала стандарт. JPEG поддерживает максимальный размер изображения 65535 × 65535 пикселей, следовательно, до 4 -х гигапикселя (при соотношении сторон 1: 1).

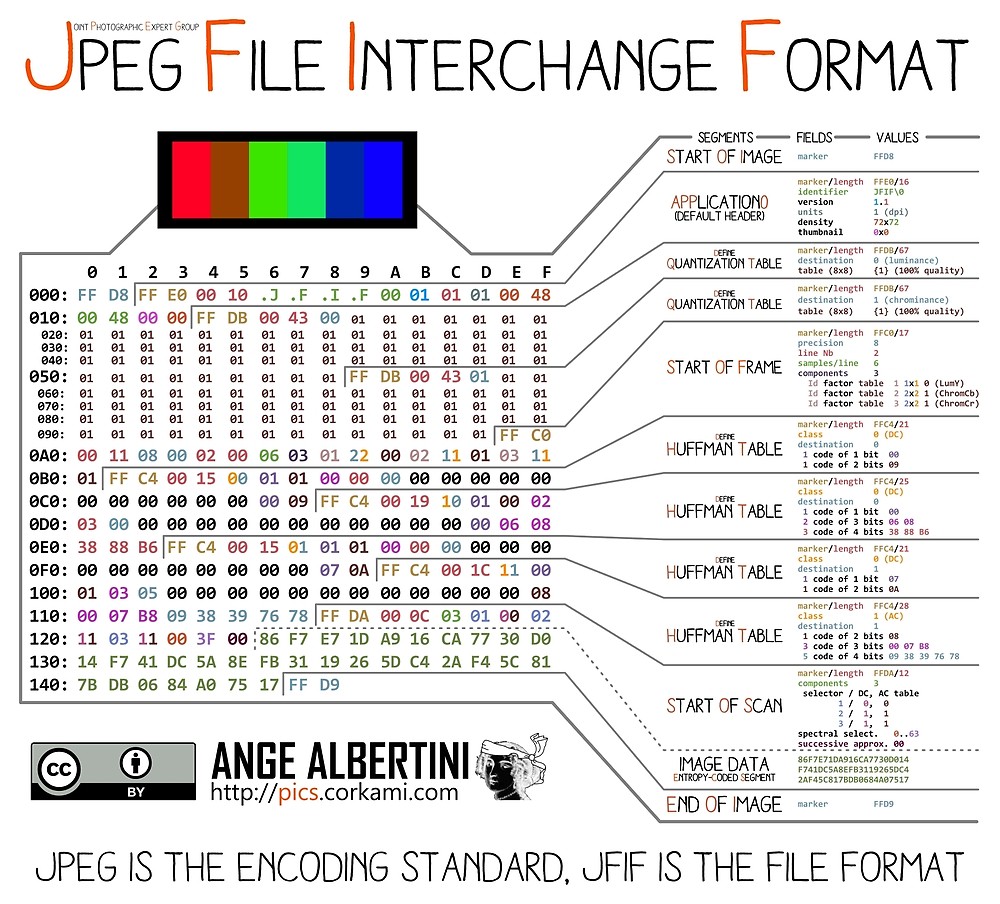


Рисунок 8. Структура формата JPEG.

Процесс сжатия по схеме JPEG включает ряд этапов:

1. Преобразование изображения в оптимальное цветовое пространство;

Первым делом происходит преобразование из цветового пространства RGB в цветовое пространство YCbCr, которое состоит из трёх компонент: компоненты яркости и двух компонент цветности и (рис. 9.1-9.2). Такое разделение вызвано тем, что человеческое зрение имеет большую чувствительность к яркости, чем к цвету предмета [13]. В связи с этим компоненты цветности и можно хранить с меньшим разрешением, что позволяет уменьшить объём сохраняемых или передаваемых данных [13]. Поэтому данное цветовое пространство широко используется в цифровых изображениях и видео. Также цветовое пространство очень часто используется в стеганографических алгоритмах, для того чтобы работать с чёрно-белым изображением, используя канал яркости.

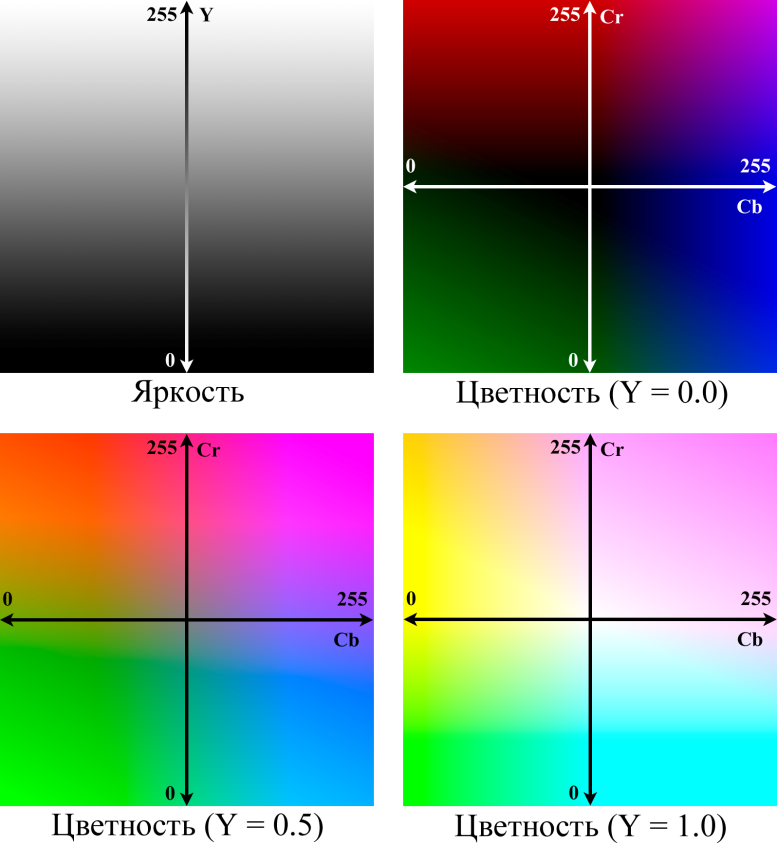


Рисунок 9.1. Цветовое пространство

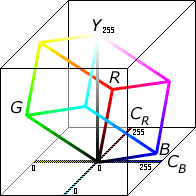


Рисунок 9.2. 3D представление цветового пространства

Перевести изображение из цветового пространства в пространство можно с помощью формулы (4.1), используемой форматом JPEG. В данной формуле компоненты пространства и пространства лежат в интервале .

Обратное преобразование из цветового пространства в пространство можно выполнить с помощью формулы (4.2), в которой компоненты обоих пространств также должны находиться в интервале .

2. Субдискретизация компонентов цветности усреднением групп пикселей;

Из-за характера светочувствительных рецепторов человеческого глаза люди могут видеть значительно более мелкие детали в яркости изображения Y, чем в цветовом тоне и насыщенности цвета Cb и Cr. Используя эти знания осуществляется более эффективное сжатие изображений, таким образом следующий шаг – «прореживание», которое заключается в том, что каждому блоку из 4 пикселов (2х2) яркостного канала Y ставятся в соответствие усреднённые значения Cb и Cr (схема прореживания «4:2:0»). При этом для каждого блока 2х2 вместо 12 значений (4 Y, 4 и ) используется всего 6 (4 Y и по одному усреднённому и ).

Если к качеству восстановленного после сжатия изображения предъявляются повышенные требования, прореживание может выполняться лишь в каком-то одном направлении — по вертикали (схема «4:4:0») или по горизонтали («4:2:2»), или не выполняться вовсе («4:4:4»). Стандарт допускает также прореживание с усреднением и не для блока 2х2, а для четырёх расположенных последовательно (по вертикали или по горизонтали) пикселов, то есть для блоков 1х4, 4х1 (схема «4:1:1»), а также 2х4 и 4х2 (схема «4:1:0»). Допускается также использование различных типов прореживания для и , но на практике такие схемы применяются исключительно редко.

3. Применение дискретных косинусных преобразований для уменьшения избыточности данных изображения;

Дискретное косинусное преобразование представляет собой разновидность преобразования Фурье и имеет обратное преобразование. Эта математическая операция преобразует изображение от координатного (2D) пространства в частотное. Перцептивная модель, основанная на психовизуальной системе человека, отбрасывает информацию высокой частоты, например, резкие переходы в интенсивности и цветовом оттенке.

Зная коэффициенты преобразования, мы всегда может произвести обратное действие - вернуть исходную картинку. DCT непосредственно применяемый к блоку (в нашем случае 8х8 пикселей) изображения будет выглядеть так:

(4.3)

Где *х,y* - пространственные координаты пикселя (0..7) ,

𝑔𝑥,- значения пикселей исходного макроблока (допустим, яркость),

*u,v* - координаты пикселя в частотном представлении (0..7),

𝛼(𝑢)= при *u*=0, в остальных случаях α(*u*) = 1,

𝛼(𝑣)=при v=0, в остальных случаях α(*v*) = 1.

4. Квантование каждого блока коэффициентов ДКП с применением весовых функций, оптимизированных с учетом визуального восприятия человеком;

Человеческий глаз хорошо видит небольшие различия в яркости на относительно большом диапазоне, но не так хорошо различает изменения яркости высокой частоты. Это позволяет значительно сократить количество информации в высокочастотных составляющих. Это делается путем простого деления каждого компонента в частотной области на константу для этого компонента, а затем округляется до ближайшего целого числа. Эта операция округления является единственной операцией с потерей качества во всем процессе (кроме прореживания цветовых компонент), если вычисление ДКП выполняется с достаточно высокой точностью. В результате этого, как правило, бывает так, что многие из более высоких частотных составляющих, округлены до нуля, и многие из остальных становятся небольшими положительными или отрицательными числами, которые занимают меньше битов при записи.

5. Этап Вторичного Сжатия

Заключительной стадией работы кодера JPEG является кодирование полученной матрицы 8x8.

5.1 Зигзагообразная перестановка 64 DCT коэффициентов

На этом этапе происходит просматривание полученной матрицы зигзагом, начиная с первого коэффициента. Суть этого заключается в том, что таким образом мы просматриваем все коэффициенты в порядке повышения пространственных частот, получая вектор из коэффициентов, отсортированных по этому параметру. Первый коэффициент - самая низкая частота, далее они идут по убыванию, таким образом в конце образуется массив из нулей.

5.2 RLE кодирование нулей.

RLE (кодирование длин серий) – алгоритм сжатия данных, заменяющий повторяющиеся символы на один символ и число его повторов. Таким образом применяя этот алгоритм мы укорачиваем запись получившегося на предыдущем этапе вектора.

5.3 Конечный шаг - кодирование Хаффмана.

Суть алгоритма сжатия Хаффмана состоит в том, что часто встречающимся в тексте символам ставятся в соответствие найденные по определенному алгоритму короткие двоичные коды, а редко встречающимся символам – более длинные коды. Для кодирования каждого сообщения может использоваться своя собственная таблица соответствий или же общая, фиксированная таблица. При кодировании пар чисел, осуществляемом в алгоритме сжатия JPEG, учитывается, что длинные последовательности нулей и большие значения коэффициентов встречаются реже, чем короткие последовательности нулей и маленькие значения коэффициентов. Полученный по алгоритму сжатия Хаффмана двоичный код готов к передаче по сети и хранению в памяти компьютера.

**Математическая модель метода LSB**

Встраивание водяного знака в изображение происходит на основе изменения яркости пикселей наименее значащих бит (LSB) контейнера. Яркостью назовём элементы матрицы f (x; y), которые принимают значения 0,1,...,255. В цифровом виде полутоновое изображение не что иное, как матрица. Далее считаем, что контейнер представляет собой RGB изображение, а ЦВЗ - полутоновое.

Пиксель полутонового изображения ЦВЗ принимает только два значения (по использованным данным). Яркость любого пикселя контейнера ‒ целое число> 0, которое можно представить в двоичном виде:

\*2+…+ (4.4)

где все

Наименее и наиболее значащими являются соответственно младший и старший биты и .

При изменении значения младшего бита на одну единицу, яркость изменится на одну единицу, при изменении значения старшего бита ‒ на 128 единиц. Биты одного разряда составляют матрицу BV, которую называют битовой плоскостью. Набор из 8 битовых плоскостей, являющихся бинарными изображениями ‒ полутоновое изображение.

Битовые плоскости в изображении имеют различный вес: 1; 2; 4...128. Младшие битовые плоскости несут минимальную информацию. Основные же детали проявляются приблизительно с четвертой битовой плоскости. Соответственно, младшие битовые плоскости идеально подходят для встраивания в них водяного знака.

Пусть G бинарное изображение, которое необходимо внедрить в один из цветовых каналов RGB изображения Hi контейнера S. Пусть запись происходит путём простого побитового сложения двух бинарных изображений . Тогда процесс слияния выглядит таким образом:

(4.5)

## **РАЗДЕЛ 4. АНАЛИЗ КРИТЕРИЕВ ВЫБОРА КОНТЕЙНЕРА**

# 4.1 Общие критерии выбора контейнеров

Существенное влияние на надежность стегосистемы и возможность обнаружения факта передачи скрытого сообщения оказывает выбор контейнера.

По протяженности контейнеры можно подразделить на два типа: непрерывные (потоковые) и ограниченной (фиксированной) длины. Особенностью потокового контейнера является то, что невозможно определить его начало или конец. Более того, нет возможности узнать заранее, какими будут последующие шумовые биты, что приводит к необходимости включать скрывающие сообщение биты в поток в реальном масштабе времени, а сами скрывающие биты выбираются с помощью специального генератора, задающего расстояние между последовательными битами в потоке.

В непрерывном потоке данных самая большая трудность для получателя - определить, когда начинается скрытое сообщение. При наличии в потоковом контейнере сигналов синхронизации или границ пакета, скрытое сообщение начинается сразу после одного из них. В свою очередь, для отправителя возможны проблемы, если он не уверен в том, что поток контейнера будет достаточно долгим для размещения целого тайного сообщения.

При использовании контейнеров фиксированной длины отправитель заранее знает размер файла и может выбрать скрывающие биты в подходящей псевдослучайной последовательности. С другой стороны, контейнеры фиксированной длины, как это уже отмечалось выше, имеют ограниченный объем и иногда встраиваемое сообщение может не поместиться в файл-контейнер.

Другой недостаток заключается в том, что расстояния между скрывающими битами равномерно распределены между наиболее коротким и наиболее длинным заданными расстояниями, в то время как истинный случайный шум будет иметь экспоненциальное распределение длин интервала. Конечно, можно породить псевдослучайные экспоненциально распределенные числа, но этот путь обычно слишком трудоемок. Однако на практике чаще всего используются именно контейнеры фиксированной длины, как наиболее распространенные и доступные.

Возможны следующие варианты контейнеров:

 Контейнер генерируется самой стегосистемой. Такой подход можно назвать конструирующей стеганографией.

 Контейнер выбирается из некоторого множества контейнеров. В этом случае генерируется большое число альтернативных контейнеров, чтобы затем выбрать наиболее подходящий контейнер, используемый для сокрытия сообщения. Такой подход можно назвать селектирующей стеганографией. В данном случае, при выборе оптимального контейнера из множества сгенерированных, важнейшим требованием является естественность контейнера. Единственной же проблемой остается то, что даже оптимально организованный контейнер позволяет спрятать незначительное количество данных при очень большом объеме самого контейнера.

 Контейнер поступает извне. В данном случае отсутствует возможность выбора контейнера и для сокрытия сообщения берется первый попавшийся контейнер, не всегда подходящий к встраиваемому сообщению. Назовем это безальтернативной стеганографией.

В настоящее время большинство исследований в области стеганографии посвящено использованию в качестве стеганоконтейнеров цифровых изображений. Это обусловлено следующими причинами:

 существованием практически значимой задачи защиты фотографий, картин, видео от незаконного тиражирования и распространения;

 относительно большим объемом цифрового представления изображений, что позволяет внедрять сообщение большого объема либо повышать скрытность внедрения;

 заранее известным размером контейнера, отсутствием ограничений, накладываемых требованиями реального времени;

 наличием в большинстве реальных изображений текстурных областей, имеющих шумовую структуру и хорошо подходящих для встраивания информации;

 слабой чувствительностью человеческого глаза к незначительным изменениям цветов изображения, его яркости, контрастности, содержанию в нем шума, искажениям вблизи контуров;

 хорошо разработанными в последнее время методами цифровой обработки изображений. Надо отметить, что последняя причина вызывает и значительные трудности в обеспечении скрытности секретных сообщений: чем более совершенными становятся методы сжатия, тем меньше остается возможностей для встраивания посторонней информации. [9]

# 4.2 Классификация критериев выбора контейнера для LSB-метода

От выбора контейнера зависит объем секретного сообщения, а также устойчивость стегоконтейнера к различным видам анализа: визуального или статистического. Способов сокрытия данных много, однако, проблема выбора подходящего контейнера до сих пор не решена. При исследовании было найдено всего несколько источников, в которых затрагивалась данная проблема.

Выбор контейнера должен рассматриваться с точки зрения метода внедрения данных, так как именно он определяет биты, которые будут модифицированы на биты сообщения. Также должен учитываться тот факт, что существуют методы анализа, позволяющие обнаружить секретное сообщение.

На данном этапе исследований выбор контейнера сделан для метода замены младших бит (LSB-метода), на основе которого сделано большинство программ внедрения сообщений. Учитывалось влияние визуального стеганоанализа, как начального этапа анализа контейнера на наличие сообщения. [5]

Классификация критериев выбора контейнера:

 отказ от общеизвестных изображений в качестве контейнера, как, например, картины «Мона Лиза» или «Черный квадрат»;

 отказ от использования в качестве контейнера изображений, конвертированных из JPEG-формата в формат BMP;

 получение изображения при помощи фотоаппарата или сканера, а не при помощи графических редакторов;

 большой размер контейнера;

 отсутствие полезной составляющей на младших битовых плоскостях изображения;

 зашумленность;

 отсутствие плавных переходов и монотонных областей;

 «пестрость»;

 большое число перепадов яркости;

 наличие большого числа пикселей, оттенки цветов которых плохо различаются глазом человека (зеленый, желтый).

Эти критерии в достаточной мере учитывают все особенности контейнера, необходимые для получения стеганоустойчивого контейнера к визуальному стеганоанализу для метода замены младших бит. [4]

**4.3 Цветность изображения как критерий выбора контейнера**

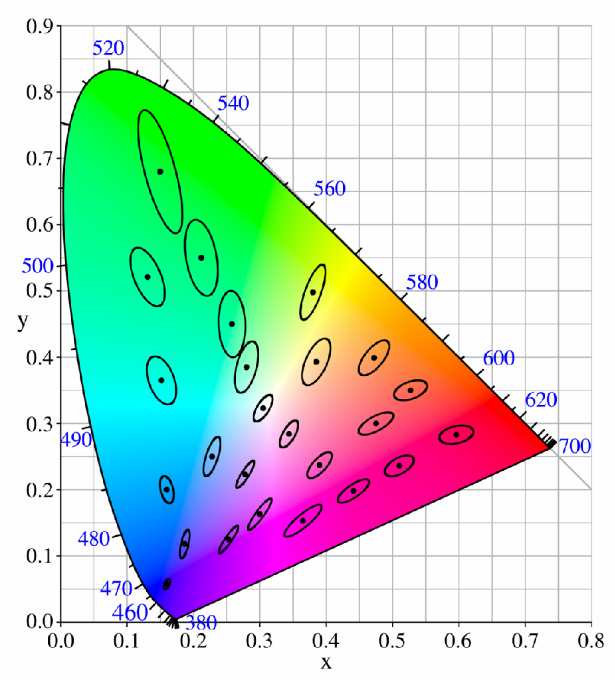
На визуальную скрытность данных влияет цветность изображения, то есть наличие цветовых областей того или иного цвета. Это объясняется неравномерной чувствительностью человеческого глаза к малым изменениям различных длин волн видимого диапазона. Человеческий глаз обладает свойством порога цветоразличения при небольших цветовых отличиях, то есть он воспринимает цвет и его «соседний» цвет как один. Величина этого порога неодинакова для разных цветов. Этот эффект представлен на рисунке 10:

Рисунок 10. – Диаграмма цветностей и пороговые эллипсы

Таким образом, замена одинакового количества младших бит красной, синей области будет более опасной для обнаружения произведенной замены глазом, чем младших бит желтой или зеленой области за счет разного порога различимости этих цветов. Выбор контейнера, который содержит наибольшие области зеленого, желтого и их смесей с белым цветом обеспечит наилучшую скрытность данных с точки зрения визуального стеганоанализа.

# 4.4 Критерий эффективности в стеганографии изображений

Под термином «эффективность» в стеганографии будем понимать возможность решения с помощью цифровых изображений основных задач стеганографии: быстро и скрытно передавать большие объемы информации. Существует очень большое количество факторов, влияющих на эффективность стеганографии цифровых изображений.

Среди этих факторов можно выделить группу технических критериев эффективности, которые поддаются строгому математическому описанию и имеют некоторый набор численных характеристик. В качестве примера такого критерия можно привести отношение максимального размера встраиваемого сообщения, не приводящего к искажению изображения, к размеру самого контейнера.

С другой стороны, существуют критерии эффективности, не поддающиеся техническому описанию, но по-прежнему играющие исключительную роль в формировании понятия «эффективность». Рассматривая несколько графических форматов, можно утверждать, что применять один из них эффективнее, чем другой. Причиной для этого может являться то, что один из форматов имеет гораздо большее распространение (в том числе, в сети Интернет), чем остальные. Более того, использование некоторых форматов для нетипичных для них целей само по себе может быть подозрительным и провоцировать атаки. Например, выложенные на сайт в сети Интернет фотографии друзей в формате BMP (имеющие размер порядка нескольких мегабайт) определенно вызовут подозрение у посетителей (ведь современные алгоритмы сжатия позволяют сжимать фотографии в 20-30 раз с приемлемой потерей качества). К тому же, для некоторых форматов (например, упомянутый выше формат BMP) разработан широчайший спектр методов и инструментов стеганоанализа, и эти форматы являются более уязвимыми, а значит и менее эффективными с точки зрения стеганографии, а значит их нет смысла использовать для работы.

Удовлетворение требованию скрытности является обязательным для абсолютно любой стеганосистемы. В применении к графической стеганосистеме, стойкость связана с изменениями (искажениями), вносимыми в исходное изображение при встраивании сообщения. Требование стойкости считается невыполненным, если изображение поддается атаке посредством простого визуального анализа. Данная стеганосистема обладает крайне низкой эффективностью и не может найти практического применения, так как не соответствует минимальному уровню безопасности.



Рисунок 11. Результат работы алгоритма, не отвечающего требованиям стойкости: 1 – исходное изображение, 2 – изображение со встроенным сообщением.

Как правило, при создании стеганографических алгоритмов, наибольший объем исследований связан именно с обеспечением скрытности. Производятся эксперименты, позволяющие установить, как изменение той или иной части файла-контейнера влияет на результирующее изображение. Стойкость стеганографического алгоритма в значительной степени определяется размерами встраиваемого сообщения.

**4.5 Размер встраиваемого сообщения**

Эффективность использования цифрового изображения для хранения секретной информации в значительной мере определяется максимальным возможным размером секретного сообщения. Как правило, численно этот критерий характеризуется процентным соотношением между объемом встраиваемого сообщения и исходным объемом контейнера. В отношении изображений, данная величина варьируется в зависимости от используемого графического формата.

Главным «ограничителем» максимального размера сообщения для конкретного графического файла выступает описанное выше требование скрытности. В стеганографии имеется фундаментальная зависимость между стойкостью встраивания и размером встраиваемого сообщения. Эта зависимость имеет обратно пропорциональный характер: чем больше объем встраиваемого в заранее заданный контейнер сообщения, тем ниже надежность сокрытия этой информации в контейнере.

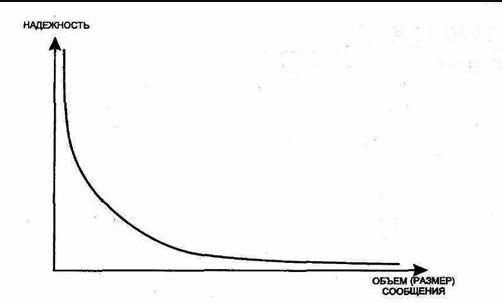


Рисунок 12. – Зависимость надежности сокрытия информации от объема сообщения

Казалось бы, приведенная закономерность не позволяет увеличивать эффективность стеганографического встраивания информации путем наращивания размера сообщения. Но это не так. Существует несколько методов повышения размеров сообщения без ущерба стойкости, о которых речь пойдет дальше.

Устойчивость к модификации заполненного контейнера (сжатию) Устойчивость к модификации характеризует вероятность восстановления сообщения, при условии некоторой модификации заполненного контейнера. Частным случаем модификации является сжатие с потерями. Особое значение этот фактор эффективности имеет для технологий внедрения цифровых водяных знаков.

Модификация заполненного контейнера может осуществляться как непреднамеренно (сжатие, ошибки при передаче файла по каналу связи с помехами), так и преднамеренно (попытка нарушить авторские права путем уничтожения ЦВЗ). Повышение устойчивости к сжатию осуществляется путем тщательного исследования алгоритмов компрессии с целью определения областей контейнера, не подвергающихся модификациям. Действенным методом борьбы с преднамеренным разрушением ЦВЗ может считаться встраивание информации в ту область файла-контейнера, изменение которой приводит к деградации изображения. Традиционным и достаточно мощным способом борьбы с «помехами» может служить увеличение избыточности встраиваемого сообщения.

В значительной степени эффективность применения цифровых изображений в стеганографии зависит от формата их хранения. В компьютерной стеганографии в качестве контейнера может выступать практически любой файловый формат, однако наиболее распространенным типом носителя являются файлы изображения формата JPEG. Это объясняется тем, что для целей стеганографии наиболее предпочтительны файлы форматов, в которых используются методы сжатия без потерь (такие виды сжатия типичны для изображений формата JPEG, BMP, PNG, TGA, и др.). Также положительной стороной в пользу выбора формата JPEG выступает высокое качество изображения и простота формата.

**JPEG**— графический формат, предназначенный для хранения файлов-изображений любого типа. Появился в 1991 году. Также .JPEG — расширение имени файла в формате JPEG и однозначно указывающее на него.

**JPG**— одно из самых популярных расширений имени файла формата JPEG, распространенное наряду с .jpeg. В программной реализации будем работать с файлами формата **.JPG** [10].

С позиции стеганографии файлы данного формата позволяют скрывать сравнительно большие объемы информации. В данной работе в качестве контейнера рассматривается 24-битовое растровое изображение в системе цветности RGB формата JPEG. Каждая цветовая комбинация тона (пикселя) представляет собой комбинацию значений яркости трех составляющих цветов – красного (R), зеленого (G) и синего (B), которые занимают каждый по 1 байту (итого по 3 байта на точку). Таким образом, яркость каждой составляющей записывается 8 - битным числом и может изменяться в диапазоне от 0 до 255 (комбинация (0, 0, 0) соответствует черному цвету, комбинация (255, 255, 255) – белому).

# 4.6 Атаки на стегосистему.

При атаке на стегосистему злоумышленник пытается взломать ее, при этом он не ограничен в вычислительных возможностях и может осуществлять любые атаки. Она взломана в том случае, когда нарушителю удалось установить и доказать факт передачи скрытого сообщения в перехваченном контейнере. Если же ему это не удалось, то считается, что эта стеганосистема устойчива. В большинстве случаев выделяют несколько этапов взлома стеганографической системы:

1. Выявление самого факта присутствия скрытого сообщения.

2. Извлечение скрытого сообщения.

3. Изменение (модификация) скрытой информации.

4. Запрет на выполнение любой пересылки информации, в том числе скрытой.

Первые два этапа являются пассивными атаками на стеганосистему, а последние - активными (или злонамеренными) атаками.

Выделяют следующие виды атак на стеганосистемы [2]:

• Атака на основании известного заполненного контейнера. При этом типе атаки нарушитель имеет в своем распоряжении один или несколько заполненных контейнеров (при этом предполагается, что встраивание скрытых данных выполнялось одним и тем же способом). Задача нарушителя может заключаться в обнаружении факта наличия стеганоканала (основное задание), а также в извлечении данных или нахождении ключа. Зная ключ, нарушитель может анализировать другие стеганосообщения.

• Атака на основании известного встроенного сообщения. Этот тип атаки больше характерен для систем защиты интеллектуальной собственности, когда в качестве ЦВЗ, например, используется известный логотип фирмы. Задачей анализа является определение ключа. Если соответствующий скрытому сообщению заполненный контейнер неизвестен, то задача является практически неразрешимой.

• Атака на основании выбранного скрытого сообщения. В этом случае нарушитель может навязывать для передачи свои сообщения и анализировать полученные при этом заполненные контейнеры.

• Адаптивная атака на основании выбранного сообщения. Этот тип атаки является частным случаем предыдущего. При этом нарушитель имеет возможность выбирать сообщения для навязывания их передачи адаптивно, в зависимости от результатов анализа предшествующих контейнеров-результатов.

• Атака на основании выбранного заполненного контейнера. Этот тип атаки более характерен для систем ЦВЗ. У нарушителя есть детектор заполненных контейнеров и несколько таких контейнеров. [1] Анализируя продетектированные скрытые сообщения, нарушитель пытается узнать ключ.

Кроме того, у нарушителя может существовать возможность применять еще три атаки, не имеющих прямых аналогов в криптоанализе:

• Атака на основании известного пустого контейнера. Если нарушитель знает пустой контейнер, то сравнивая его с подозреваемым на присутствие скрытой информации контейнером он всегда может установить факт наличия стеганоканала. Хоть этот случай и тривиален, в ряде изданий приводится его информационно-теоретическое обоснование. Более интересна вариация этого сценария, когда контейнер известен только приблизительно, с некоторой погрешностью (например, если к нему добавлен шум). В этом случае возможно построить устойчивую стеганосистему.

• Атака на основании выбранного пустого контейнера. В этом случае нарушитель способен навязать для пользования предложенный им пустой контейнер. Последний может иметь нужные особенности, например, иметь значительные однородные области (однотонные изображения), и тогда добиться секретности встраивания будет сложнее.

• Атака на основании известной математической модели контейнера или его части. При этом атакующий пытается определить отличие подозреваемого сообщения от известной ему модели.

Например, можно допустить, что биты в середине определенной части изображения являются коррелированными. Тогда отсутствие такой корреляции может служить сигналом о наличии скрытого сообщения. Задача того, кто встраивает сообщение, заключается в том, чтобы не нарушить статистики контейнера. Отправитель и тот, кто атакует, могут иметь в своем распоряжении разные модели сигналов, тогда в информационно-скрывающем противоборстве победит тот, кто владеет более эффективной (оптимальной) моделью.

Основная цель атаки на стеганосистему аналогична цели атак на криптосистему с тем отличием, что резко возрастает значимость активных (злонамеренных) атак. Любой контейнер может быть заменен с целью удаления или разрушения скрытого сообщения, независимо от того, существует оно в контейнере или нет. Обнаружение существования скрытых данных ограничивает время на этапе, их удаления, необходимое для обработки только тех контейнеров, которые содержат скрытую информацию. [1]

Даже при оптимальных условиях для атаки задача извлечения скрытого сообщения из контейнера может оказаться очень сложной. Однозначно утверждать о факте существования скрытой информации можно только после ее выделения в явном виде. Иногда целью стеганографического анализа является не алгоритм вообще, а поиск, например, конкретного стеганоключа, используемого для выбора битов контейнера в стеганопреобразовании.

Фактически, LSB – это шум, поэтому его можно использовать для встраивания информации путем замены менее значащих битов пикселей изображения битами секретного сообщения. Из-за особенностей формата JPEG данные в реализуемой стегосистеме будут записываться не в цветовые компоненты, а в коэффициенты ДКП. Основной недостаток метода – слабая устойчивость к искажениям контейнера. [2]

**Выводы**

Во второй главе рассмотрены общие типы стеганографических контейнеров, классифицированных по критерию протяженности. Цифровые изображения относятся к типу контейнеров фиксированной длины. Ограниченный объем контейнеров фиксированной длины является существенным недостатком данного типа контейнеров, что, в свою очередь, делает еще более актуальным исследование возможности увеличения объема встраиваемой информации за счет использования старших бит изображений. Сформулированы основные требования к выбору контейнера для стеганографического скрытия данных методом наименее значащих бит цифрового изображения, основанных на свойствах цифровых изображений. Представленные требования к выбору контейнера являются важными условиям, позволяющими лишить нарушителя заведомых преимуществ в обнаружении факта сокрытия информации и необходимы для удовлетворения условий эффективности в стеганографии, использующей цифровые изображения в качестве контейнеров. Критерии эффективности, описанные в текущей главе, можно выделить в две условные группы: технические критерии и критерии, не поддающиеся техническому описанию. В качестве технического критерия оценки эффективности, можно привести пример отношения максимального размера встраиваемого сообщения, не приводящего к искажению изображения, к размеру самого контейнера. В свою очередь, используемых графический формат, не поддающийся строгому математическому описанию, является важным условием эффективности в стеганографии.

Таким образом, в ходе анализа форматов изображений, как контейнера был выбран формат .JPG, как самое распространенное расширение .JPEG.   
Из-за его глобального распространения и всеобщего использования, и к тому же, чтоб подчеркнуть уникальность работы, т.к большинство программных средств работают с форматом BMP, из-за простоты и понятности, а как показывает практика и статистика уже никто не пользуется этим форматом (BMP).



## **РАЗДЕЛ 5. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

# 5.1 Назначение и структура ПО

Для реализация выбранного стеганографического метода наименьшего значащего бита (LSB) для скрытия информации в графических изображениях формата JPEG было разработано программное обеспечение в виде пакета «Стеганография».

Приложение выполняет стеговложение в графическое изображения формата JPEG (JPG) любой введенной в специальное поле текстовой информации при введении пароля для генерации стегоключа. Также в программе реализован процесс извлечения данных из контейнера в обратном порядке.

# 5.2 Выбор инструментов

В качестве языка для разработки был выбран Python — высокоуровневый язык с минималистичным синтаксисом, сфокусированный на производительности разработчика и читаемости кода. Очень распространенный язык в следствии своей простоты.

Существует большое количество сторонних библиотек и пакетов, расширяющих стандартную функциональность языка. Это и многопоточная работа, и работа с удаленными серверами, и работа с большими массивами данных, а также астрономические расчеты, удобная работа с изображениями и построение графиков. Кстати, последнее является очевидным преимуществом языка и экосистемы Python в целом перед другими скриптовыми аналогами.

Разработанный прототип будет являться пригодным для встраивания в работающие системы. Для разработки прототипа были выбраны следующие библиотеки:

– numpy — работа с разными типами чисел и массивами данных;

– **hashlib** — шифрование строк;

– base64 — применение кодировки Base64;

– cv2 — мощная библиотека машинного зрения;

– PyQt5 — создание графического интерфейса.

Библиотеки cv2 или OpenCV как она называется python используется для реализации стеганографии изображений (вложение/извлечение). Это делается с использованием метода младшего значащего бита (LSB) программа позволяет пользователю выбирать количество бит, используемых для встраивания, поэтому при использовании 1 бит встроенное изображение почти не поддается обнаружению человеческому глазу, а с помощью 7 вы можете четко отображать скрытое изображение.

Кроме того, cv2.imread() будет генерировать массив numpy пикселей. Вместо того, чтобы перебирать каждый пиксель, вы можете прорисовывать свои вычисления. Также среди возможностей пакета: чтение изображений разных форматов (PNG, JPEG, TIF, GIF и др.); сохранение изображений в эти форматы; разбор изображений на составляющие (цветные изображения на каналы); работа с изображением, как с двумерным массивом данных. Что удобно, данные изображения легко преобразовываются в типы данных, с которыми работает пакет numpy, и обратно. Нет необходимости прикладывать дополнительные усилия для визуализации двумерного массива данных.

Для шифрования строк предназначен **модуль hashlib**. Наиболее часто применяемой функцией является функция md5(), которая шифрует строку с помощью алгоритма MD5. Она используется для шифрования паролей так как не существует алгоритма для дешифровки. Для сравнения введенного пользователем пароля с сохраненным в базе необходимо зашифровать введенный пароль, а затем произвести сравнение.

Cryptography.fernet гарантирует, что сообщение, зашифрованное с его помощью, невозможно манипулировать или читать без ключа. [Fernet](https://github.com/fernet/spec/) - это реализация симметричной (также известной как «секретный ключ») аутентифицированной крипMultiFernet.

Ключ высчитывается хэш-функцией MD5 на основании пароля, который запрашивается у пользователя при старте программы.

# 5.3 Описание основных компонентов программы.

*Подключение библиотек в программе выполняется следующими инструкциями:*

from cv2 import imread,imwrite

import numpy as np

from base64 import urlsafe\_b64encode

from hashlib import md5

from cryptography.fernet import Fernet

from custom\_exceptions import \*

from PyQt5 import QtCore, QtGui, QtWidgets

*Двоичное представление строки и ее возврат:*

def str2bin(string):

return ''.join((bin(ord(i))[2:]).zfill(7) for i in string)

*Возвращает текстовое представление двоичной строки:*

def bin2str(string):

return ''.join(chr(int(string[i:i+7],2)) for i in range(len(string))[::7])

*Возвращает зашифрованную / расшифрованную форму строки в зависимости от ввода режима:*

def encrypt\_decrypt(string,password,mode='enc'):

\_hash = md5(password.encode()).hexdigest()

cipher\_key = urlsafe\_b64encode(\_hash.encode())

cipher = Fernet(cipher\_key)

if mode == 'enc':

return cipher.encrypt(string.encode()).decode()

else:

return cipher.decrypt(string.encode()).decode()

*Кодирует секретные данные в изображении:*

def encode(input\_filepath,text,output\_filepath,password=None,progressBar=None):

if password != None:

data = encrypt\_decrypt(text,password,'enc') #If password is provided, encrypt the data with given password

else:

data = text

data\_length = bin(len(data))[2:].zfill(32)

bin\_data = iter(data\_length + str2bin(data))

img = imread(input\_filepath,1)

if img is None:

raise FileError("The image file '{}' is inaccessible".format(input\_filepath))

height,width = img.shape[0],img.shape[1]

encoding\_capacity = height\*width\*3

total\_bits = 32+len(data)\*7

if total\_bits > encoding\_capacity:

raise DataError("The data size is too big to fit in this image!")

completed = False

modified\_bits = 0

progress = 0

progress\_fraction = 1/total\_bits

for i in range(height):

for j in range(width):

pixel = img[i,j]

for k in range(3):

try:

x = next(bin\_data)

except StopIteration:

completed = True

break

if x == '0' and pixel[k]%2==1:

pixel[k] -= 1

modified\_bits += 1

elif x=='1' and pixel[k]%2==0:

pixel[k] += 1

modified\_bits += 1

if progressBar != None: #If progress bar object is passed

progress += progress\_fraction

progressBar.setValue(progress\*100)

if completed:

break

if completed:

break

written = imwrite(output\_filepath,img)

if not written:

raise FileError("Failed to write image '{}'".format(output\_filepath))

loss\_percentage = (modified\_bits/encoding\_capacity)\*100

return loss\_percentage

*Извлекает секретные данные из входного изображения при правильном вводе пароля:*

def decode(input\_filepath,password=None,progressBar=None):

result,extracted\_bits,completed,number\_of\_bits = '',0,False,None

img = imread(input\_filepath)

if img is None:

raise FileError("The image file '{}' is inaccessible".format(input\_filepath))

height,width = img.shape[0],img.shape[1]

for i in range(height):

for j in range(width):

for k in img[i,j]:

result += str(k%2)

extracted\_bits += 1

if progressBar != None and number\_of\_bits != None: #If progress bar object is passed

progressBar.setValue(100\*(extracted\_bits/number\_of\_bits))

if extracted\_bits == 32 and number\_of\_bits == None: #If the first 32 bits are extracted, it is our data size. Now extract the original data

number\_of\_bits = int(result,2)\*7

result = ''

extracted\_bits = 0

elif extracted\_bits == number\_of\_bits:

completed = True

break

if completed:

break

if completed:

break

if password == None:

return bin2str(result)

else:

try:

return encrypt\_decrypt(bin2str(result),password,'dec')

except:

raise PasswordError("Invalid password!")

*Вызов основной функции main, ввод переменных, вывод подтверждений и ошибок на экран:*

if \_\_name\_\_ == "\_\_main\_\_":

ch = int(input('What do you want to do?\n\n1.Encrypt\n2.Decrypt\n\nInput(1/2): '))

if ch == 1:

ip\_file = input('\nEnter cover image name(path)(with extension): ')

text = input('Enter secret data: ')

pwd = input('Enter password: ')

op\_file = input('Enter output image name(path)(with extension): ')

try:

loss = encode(ip\_file,text,op\_file,pwd)

except FileError as fe:

print("Error: {}".format(fe))

except DataError as de:

print("Error: {}".format(de))

else:

print('Encoded Successfully!\nImage Data Loss = {:.5f}%'.format(loss))

elif ch == 2:

ip\_file = input('Enter image path: ')

pwd = input('Enter password: ')

try:

data = decode(ip\_file,pwd)

except FileError as fe:

print("Error: {}".format(fe))

except PasswordError as pe:

print('Error: {}'.format(pe))

else:

print('Decrypted data:',data)

else:

print('Wrong Choice!')

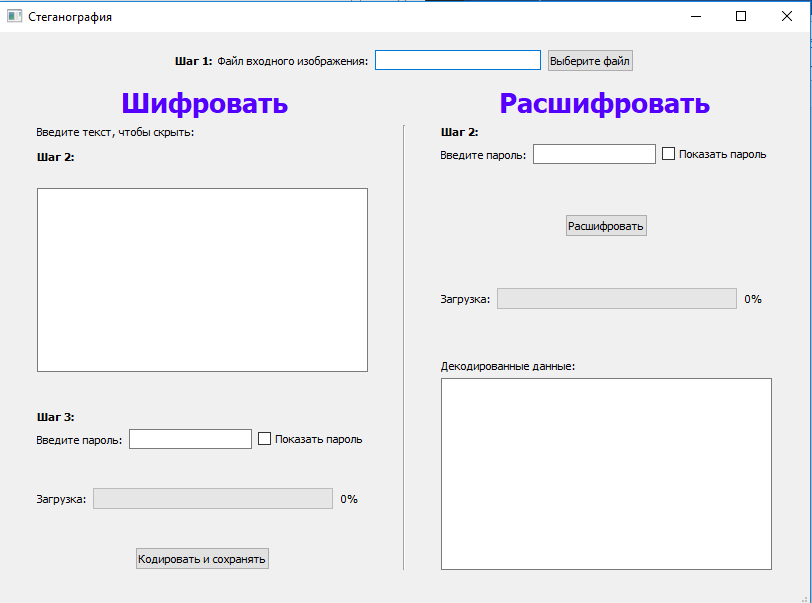


Рисунок 13 - Главное окно программы.

# Перспективы

Нет сомнений в том, что алгоритмы встраивания данных в файлы мультимедиа будут развиваться. Перспективные направления – работа с файлами JPEG, подвижными изображениями (с эффектом анимации) и всевозможными форматами видео. Так или иначе, ставший стандартным подход с LSB будет использоваться ещё несколько лет, и в будущем новые алгоритмы заменят его разве что из-за иных (новых) форм визуализации.

На сегодняшний день видение прогресса можно выразить рисунком.

Рисунок 14. Эволюция стеганографии. Вчера, сегодня, завтра.

## **ЗАКЛЮЧЕНИЕ**

Анализ тенденций развития цифровой стеганографии показывает, что в ближайшие годы интерес к развитию ее методов будет усиливаться все больше и больше. Сильнейшим катализатором этого процесса является лавинообразное развитие Internet, в том числе, такие нерешенные противоречивые её проблемы, как защита авторского права, защита прав на личную тайну, организация электронной торговли, компьютерная преступность и кибертерроризм. Стеганография может обеспечивать помехоустойчивую аутентификацию мультимедийной информации, контроль целостности данных, защиту прав собственника мультимедийной информации, отслеживание распространения информации.

Что касается стегоконтейнеров, на мой взгляд, файлы мультимедиа будут еще очень долгое время применяться в этом качестве поскольку они распространены повсеместно и обмен файлами такого типа весьма распространенное действие, чтоб вызывать подозрения факта скрытой передачи информации. С другой стороны, эти же файлы нуждаются в защите с точки зрения прав авторства на них. Стеганография предлагает и этого вопроса.

Коллекция способов встраивания сообщений, заголовков, маркеров в мультимедийные стегоконтейнеры многообразна и пополняется с каждым днём новыми модификациями алгоритмов. Наиболее простой пример – встраивание сообщений в наименьшие значащие биты изображений. Перспективными являются алгоритмы, приспособленные для работы с файлами JPEG и JPEG 2000. Эти форматы допускают сжатие за счёт избыточности визуальной информации, основанной на корреляции соседних областей изображения. В этом случае изображение рассматривается как сигнал, который может подвергаться преобразованиям Фурье, дискретному конусному преобразованию. Для видеофайлов также применимы подходы ДКП и вейвлет-преобразований.

Основная проблема современной цифровой стеганографии – отсутствие стандартов. При наличии достаточного числа известных алгоритмов, отсутствует организованная практика их применения. Надеюсь, что в ближайшие годы в результате инициатив молодых специалистов определится в каком-то виде стандарт стеганографических средств защиты информации.

## **ЛИТЕРАТУРА**

1. Коханович Г.Ф., Пузыренко А.Ю. Компьютерная стеганография. Теория и

практика. – К.: «МК-Пресс», 2006 г.

2.Pfitzmann B. Information Hiding Terminology, in Information Hiding, Springer Lecture Notes in Computer Science, v.1174, 1996

3. Генне, О. В. Основные положения стеганографии. "Защита информации. Конфидент" №3 за 2000 год

4. Бородин Г.А., Чиркова С.В., «Классификация критериев выбора контейнера для LSB-метода», «Радиоэлектроника, электротехника и энергетика 13-ая межд. науч.-техн. конф. студ. и асп. Тезисы докладов в 3-ех томах». Т.1. –М.: МЭИ, 2007

5.J. Fridrich, R. Du, and L. Meng, “Steganalysis of LSB Encoding in Color Images”, *ICME 2000*, New York City.

6.Замена наименее значащего бита [Электронный ресурс]. – Режим доступа: http://www.nestego.ru/2012/07/lsb.html.

7. Ф.Н. Сафиуллин «Разработка методов встраивания информации в

пространственную область изображения» [Электронный ресурс] – http://

textarchive.ru/c-2793733-pall.html.

8. Гонсалес Р., Вудс Р. Цифровая обработка изображений //М.: Техносфера. –

2012. – Т. 1104.

9. В.Г. Грибунин «Цифровая стеганография» [Электронный ресурс] – https://

tech.wikireading.ru/13298.

10. J. Fridrich, G. Miroslav, R. Du Steganalysis Based on JPEG Compatibility. - SUNY Binghamton, New York, 2001. - 6 c

11. Грибунин В. Г., Оков И. Н., Туринцев И. В. Цифровая Стеганография. - М.:Солон-Пресс, 2002 - 272 с.

12. Singh, H. V. Robust copyright marking using Weibull distribution / H. V. Singh, S. Rai, A. Mohan, S. P. Singh // Computers & Electrical Engineering. – 2011. – № 37(5). – С. 714-728.

13. Ричардсон, Я. Видеокодирование H.264 и MPEG-4 – стандарты нового поколения / Я. Ричардсон; пер. с англ. В. В. Чепыжова. – Москва: Техносфера, 2005. – 368 с. – (Мир цифровой обработки).

## **ПРИЛОЖЕНИЕ**

Листинг исходного кода программы.

# Исходный код lsb\_steganography.py:

#!/usr/bin/python3

from cv2 import imread,imwrite

import numpy as np

from base64 import urlsafe\_b64encode

from hashlib import md5

from cryptography.fernet import Fernet

from custom\_exceptions import \*

#Returns binary representation of a string

def str2bin(string):

return ''.join((bin(ord(i))[2:]).zfill(7) for i in string)

#Returns text representation of a binary string

def bin2str(string):

return ''.join(chr(int(string[i:i+7],2)) for i in range(len(string))[::7])

#Returns the encrypted/decrypted form of string depending upon mode input

def encrypt\_decrypt(string,password,mode='enc'):

\_hash = md5(password.encode()).hexdigest()

cipher\_key = urlsafe\_b64encode(\_hash.encode())

cipher = Fernet(cipher\_key)

if mode == 'enc':

return cipher.encrypt(string.encode()).decode()

else:

return cipher.decrypt(string.encode()).decode()

#Encodes secret data in image

def encode(input\_filepath,text,output\_filepath,password=None,progressBar=None):

if password != None:

data = encrypt\_decrypt(text,password,'enc') #If password is provided, encrypt the data with given password

else:

data = text

data\_length = bin(len(data))[2:].zfill(32)

bin\_data = iter(data\_length + str2bin(data))

img = imread(input\_filepath,1)

if img is None:

raise FileError("The image file '{}' is inaccessible".format(input\_filepath))

height,width = img.shape[0],img.shape[1]

encoding\_capacity = height\*width\*3

total\_bits = 32+len(data)\*7

if total\_bits > encoding\_capacity:

raise DataError("The data size is too big to fit in this image!")

completed = False

modified\_bits = 0

progress = 0

progress\_fraction = 1/total\_bits

for i in range(height):

for j in range(width):

pixel = img[i,j]

for k in range(3):

try:

x = next(bin\_data)

except StopIteration:

completed = True

break

if x == '0' and pixel[k]%2==1:

pixel[k] -= 1

modified\_bits += 1

elif x=='1' and pixel[k]%2==0:

pixel[k] += 1

modified\_bits += 1

if progressBar != None: #If progress bar object is passed

progress += progress\_fraction

progressBar.setValue(progress\*100)

if completed:

break

if completed:

break

written = imwrite(output\_filepath,img)

if not written:

raise FileError("Failed to write image '{}'".format(output\_filepath))

loss\_percentage = (modified\_bits/encoding\_capacity)\*100

return loss\_percentage

#Extracts secret data from input image

def decode(input\_filepath,password=None,progressBar=None):

result,extracted\_bits,completed,number\_of\_bits = '',0,False,None

img = imread(input\_filepath)

if img is None:

raise FileError("The image file '{}' is inaccessible".format(input\_filepath))

height,width = img.shape[0],img.shape[1]

for i in range(height):

for j in range(width):

for k in img[i,j]:

result += str(k%2)

extracted\_bits += 1

if progressBar != None and number\_of\_bits != None: #If progress bar object is passed

progressBar.setValue(100\*(extracted\_bits/number\_of\_bits))

if extracted\_bits == 32 and number\_of\_bits == None: #If the first 32 bits are extracted, it is our data size. Now extract the original data

number\_of\_bits = int(result,2)\*7

result = ''

extracted\_bits = 0

elif extracted\_bits == number\_of\_bits:

completed = True

break

if completed:

break

if completed:

break

if password == None:

return bin2str(result)

else:

try:

return encrypt\_decrypt(bin2str(result),password,'dec')

except:

raise PasswordError("Invalid password!")

if \_\_name\_\_ == "\_\_main\_\_":

ch = int(input('What do you want to do?\n\n1.Encrypt\n2.Decrypt\n\nInput(1/2): '))

if ch == 1:

ip\_file = input('\nEnter cover image name(path)(with extension): ')

text = input('Enter secret data: ')

pwd = input('Enter password: ')

op\_file = input('Enter output image name(path)(with extension): ')

try:

loss = encode(ip\_file,text,op\_file,pwd)

except FileError as fe:

print("Error: {}".format(fe))

except DataError as de:

print("Error: {}".format(de))

else:

print('Encoded Successfully!\nImage Data Loss = {:.5f}%'.format(loss))

elif ch == 2:

ip\_file = input('Enter image path: ')

pwd = input('Enter password: ')

try:

data = decode(ip\_file,pwd)

except FileError as fe:

print("Error: {}".format(fe))

except PasswordError as pe:

print('Error: {}'.format(pe))

else:

print('Decrypted data:',data)

else:

print('Wrong Choice!')

# Исходный код lsb\_steganography\_GUI.pyw:

#!/usr/bin/python3

from PyQt5 import QtCore, QtGui, QtWidgets

import lsb\_steganography as stego

class Ui\_MainWindow(object):

#Function to display message/error/information

def displayMsg(self,title,msg,ico\_type=None):

MsgBox = QtWidgets.QMessageBox()

MsgBox.setText(msg)

MsgBox.setWindowTitle(title)

if ico\_type == 'err':

ico = QtWidgets.QMessageBox.Critical

else:

ico = QtWidgets.QMessageBox.Information

MsgBox.setIcon(ico)

MsgBox.exec()

#Function to choose input file

def getFile(self):

file\_path = QtWidgets.QFileDialog.getOpenFileName(None, 'Open file','',"Image files (\*.jpg \*.png \*.bmp)")[0]

if file\_path != '':

self.lineEdit.setText(file\_path)

#Function to display save file dialog

def saveFile(self):

output\_path = QtWidgets.QFileDialog.getSaveFileName(None, 'Save encoded file','',"PNG File(\*.png)")[0]

return output\_path

#Function to encode data and save file

def encode(self):

input\_path = self.lineEdit.text()

text = self.plainTextEdit.toPlainText()

password = self.lineEdit\_2.text()

if input\_path == '':

self.displayMsg('Error: No file chosen','You must select input image file!','err')

elif text == '':

self.displayMsg('Text is empty','Please enter some text to hide!')

elif password == '':

self.displayMsg('Error: No password given','Please enter a password!','err')

else:

output\_path = self.saveFile()

if output\_path == '':

self.displayMsg('Operation cancelled','Operation cancelled by user!')

else:

try:

loss = stego.encode(input\_path,text,output\_path,password,self.progressBar)

except stego.FileError as fe:

self.displayMsg('File Error',str(fe),'err')

except stego.DataError as de:

self.displayMsg('Data Error',str(de),'err')

else:

self.displayMsg('Success','Encoded Successfully!\n\nImage Data Loss = {:.5f} %'.format(loss))

self.progressBar.setValue(0)

#Function to decode data

def decode(self):

input\_path = self.lineEdit.text()

password = self.lineEdit\_3.text()

if input\_path == '':

self.displayMsg('Error: No file chosen','You must select input image file!','err')

elif password == '':

self.displayMsg('Error: No password given','Please enter a password!','err')

else:

try:

data = stego.decode(input\_path,password,self.progressBar\_2)

except stego.FileError as fe:

self.displayMsg('File Error',str(fe),'err')

except stego.PasswordError as pe:

self.displayMsg('Password Error',str(pe),'err')

self.progressBar\_2.setValue(0)

else:

self.displayMsg('Success','Decoded successfully!')

self.plainTextEdit\_2.document().setPlainText(data)

self.progressBar\_2.setValue(0)

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.resize(811, 575)

MainWindow.setAutoFillBackground(False)

MainWindow.setStyleSheet("")

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.verticalLayout\_5 = QtWidgets.QVBoxLayout(self.centralwidget)

self.verticalLayout\_5.setObjectName("verticalLayout\_5")

spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)

self.verticalLayout\_5.addItem(spacerItem)

self.horizontalLayout\_2 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_2.setObjectName("horizontalLayout\_2")

spacerItem1 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_2.addItem(spacerItem1)

self.label\_4 = QtWidgets.QLabel(self.centralwidget)

self.label\_4.setObjectName("label\_4")

self.horizontalLayout\_2.addWidget(self.label\_4)

self.label\_3 = QtWidgets.QLabel(self.centralwidget)

self.label\_3.setObjectName("label\_3")

self.horizontalLayout\_2.addWidget(self.label\_3)

self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)

self.lineEdit.setObjectName("lineEdit")

self.horizontalLayout\_2.addWidget(self.lineEdit)

self.pushButton = QtWidgets.QPushButton(self.centralwidget)

self.pushButton.setAutoFillBackground(False)

self.pushButton.setAutoDefault(True)

self.pushButton.setObjectName("pushButton")

self.horizontalLayout\_2.addWidget(self.pushButton)

spacerItem2 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_2.addItem(spacerItem2)

self.verticalLayout\_5.addLayout(self.horizontalLayout\_2)

spacerItem3 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)

self.verticalLayout\_5.addItem(spacerItem3)

self.horizontalLayout = QtWidgets.QHBoxLayout()

self.horizontalLayout.setObjectName("horizontalLayout")

self.label = QtWidgets.QLabel(self.centralwidget)

self.label.setObjectName("label")

self.horizontalLayout.addWidget(self.label)

self.label\_2 = QtWidgets.QLabel(self.centralwidget)

self.label\_2.setObjectName("label\_2")

self.horizontalLayout.addWidget(self.label\_2)

self.verticalLayout\_5.addLayout(self.horizontalLayout)

self.horizontalLayout\_5 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_5.setObjectName("horizontalLayout\_5")

spacerItem4 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_5.addItem(spacerItem4)

self.verticalLayout\_2 = QtWidgets.QVBoxLayout()

self.verticalLayout\_2.setObjectName("verticalLayout\_2")

self.verticalLayout = QtWidgets.QVBoxLayout()

self.verticalLayout.setObjectName("verticalLayout")

self.label\_5 = QtWidgets.QLabel(self.centralwidget)

self.label\_5.setObjectName("label\_5")

self.verticalLayout.addWidget(self.label\_5)

self.label\_7 = QtWidgets.QLabel(self.centralwidget)

self.label\_7.setObjectName("label\_7")

self.verticalLayout.addWidget(self.label\_7)

self.plainTextEdit = QtWidgets.QPlainTextEdit(self.centralwidget)

self.plainTextEdit.setObjectName("plainTextEdit")

self.verticalLayout.addWidget(self.plainTextEdit)

self.verticalLayout\_2.addLayout(self.verticalLayout)

spacerItem5 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)

self.verticalLayout\_2.addItem(spacerItem5)

self.label\_8 = QtWidgets.QLabel(self.centralwidget)

self.label\_8.setObjectName("label\_8")

self.verticalLayout\_2.addWidget(self.label\_8)

self.horizontalLayout\_3 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_3.setObjectName("horizontalLayout\_3")

self.label\_9 = QtWidgets.QLabel(self.centralwidget)

self.label\_9.setObjectName("label\_9")

self.horizontalLayout\_3.addWidget(self.label\_9)

self.lineEdit\_2 = QtWidgets.QLineEdit(self.centralwidget)

self.lineEdit\_2.setInputMethodHints(QtCore.Qt.ImhNone)

self.lineEdit\_2.setEchoMode(QtWidgets.QLineEdit.Password)

self.lineEdit\_2.setObjectName("lineEdit\_2")

self.horizontalLayout\_3.addWidget(self.lineEdit\_2)

self.checkBox = QtWidgets.QCheckBox(self.centralwidget)

self.checkBox.setObjectName("checkBox")

self.horizontalLayout\_3.addWidget(self.checkBox)

self.verticalLayout\_2.addLayout(self.horizontalLayout\_3)

spacerItem6 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)

self.verticalLayout\_2.addItem(spacerItem6)

self.horizontalLayout\_6 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_6.setObjectName("horizontalLayout\_6")

self.label\_11 = QtWidgets.QLabel(self.centralwidget)

self.label\_11.setObjectName("label\_11")

self.horizontalLayout\_6.addWidget(self.label\_11)

self.progressBar = QtWidgets.QProgressBar(self.centralwidget)

self.progressBar.setProperty("value", 0)

self.progressBar.setTextVisible(True)

self.progressBar.setObjectName("progressBar")

self.horizontalLayout\_6.addWidget(self.progressBar)

self.verticalLayout\_2.addLayout(self.horizontalLayout\_6)

spacerItem7 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)

self.verticalLayout\_2.addItem(spacerItem7)

self.pushButton\_2 = QtWidgets.QPushButton(self.centralwidget)

self.pushButton\_2.setObjectName("pushButton\_2")

self.verticalLayout\_2.addWidget(self.pushButton\_2, 0, QtCore.Qt.AlignHCenter)

self.horizontalLayout\_5.addLayout(self.verticalLayout\_2)

spacerItem8 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_5.addItem(spacerItem8)

self.line = QtWidgets.QFrame(self.centralwidget)

self.line.setFrameShape(QtWidgets.QFrame.VLine)

self.line.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line.setObjectName("line")

self.horizontalLayout\_5.addWidget(self.line)

spacerItem9 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_5.addItem(spacerItem9)

self.verticalLayout\_4 = QtWidgets.QVBoxLayout()

self.verticalLayout\_4.setObjectName("verticalLayout\_4")

self.label\_6 = QtWidgets.QLabel(self.centralwidget)

self.label\_6.setObjectName("label\_6")

self.verticalLayout\_4.addWidget(self.label\_6)

self.horizontalLayout\_4 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_4.setObjectName("horizontalLayout\_4")

self.label\_10 = QtWidgets.QLabel(self.centralwidget)

self.label\_10.setObjectName("label\_10")

self.horizontalLayout\_4.addWidget(self.label\_10)

self.lineEdit\_3 = QtWidgets.QLineEdit(self.centralwidget)

self.lineEdit\_3.setInputMethodHints(QtCore.Qt.ImhNone)

self.lineEdit\_3.setEchoMode(QtWidgets.QLineEdit.Password)

self.lineEdit\_3.setObjectName("lineEdit\_3")

self.horizontalLayout\_4.addWidget(self.lineEdit\_3)

self.checkBox\_2 = QtWidgets.QCheckBox(self.centralwidget)

self.checkBox\_2.setObjectName("checkBox\_2")

self.horizontalLayout\_4.addWidget(self.checkBox\_2)

self.verticalLayout\_4.addLayout(self.horizontalLayout\_4)

spacerItem10 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)

self.verticalLayout\_4.addItem(spacerItem10)

self.pushButton\_3 = QtWidgets.QPushButton(self.centralwidget)

self.pushButton\_3.setObjectName("pushButton\_3")

self.verticalLayout\_4.addWidget(self.pushButton\_3, 0, QtCore.Qt.AlignHCenter)

spacerItem11 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)

self.verticalLayout\_4.addItem(spacerItem11)

self.horizontalLayout\_7 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_7.setObjectName("horizontalLayout\_7")

self.label\_13 = QtWidgets.QLabel(self.centralwidget)

self.label\_13.setObjectName("label\_13")

self.horizontalLayout\_7.addWidget(self.label\_13)

self.progressBar\_2 = QtWidgets.QProgressBar(self.centralwidget)

self.progressBar\_2.setEnabled(True)

self.progressBar\_2.setMaximum(100)

self.progressBar\_2.setProperty("value", 0)

self.progressBar\_2.setTextVisible(True)

self.progressBar\_2.setObjectName("progressBar\_2")

self.horizontalLayout\_7.addWidget(self.progressBar\_2)

self.verticalLayout\_4.addLayout(self.horizontalLayout\_7)

spacerItem12 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)

self.verticalLayout\_4.addItem(spacerItem12)

self.verticalLayout\_3 = QtWidgets.QVBoxLayout()

self.verticalLayout\_3.setObjectName("verticalLayout\_3")

self.label\_12 = QtWidgets.QLabel(self.centralwidget)

self.label\_12.setObjectName("label\_12")

self.verticalLayout\_3.addWidget(self.label\_12)

self.plainTextEdit\_2 = QtWidgets.QPlainTextEdit(self.centralwidget)

self.plainTextEdit\_2.setReadOnly(True)

self.plainTextEdit\_2.setObjectName("plainTextEdit\_2")

self.verticalLayout\_3.addWidget(self.plainTextEdit\_2)

self.verticalLayout\_4.addLayout(self.verticalLayout\_3)

self.horizontalLayout\_5.addLayout(self.verticalLayout\_4)

spacerItem13 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_5.addItem(spacerItem13)

self.verticalLayout\_5.addLayout(self.horizontalLayout\_5)

spacerItem14 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)

self.verticalLayout\_5.addItem(spacerItem14)

MainWindow.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(MainWindow)

self.menubar.setGeometry(QtCore.QRect(0, 0, 811, 21))

self.menubar.setObjectName("menubar")

self.menuHelp = QtWidgets.QMenu(self.menubar)

self.menuHelp.setObjectName("menuHelp")

MainWindow.setMenuBar(self.menubar)

self.statusbar = QtWidgets.QStatusBar(MainWindow)

self.statusbar.setObjectName("statusbar")

MainWindow.setStatusBar(self.statusbar)

self.actionAbout = QtWidgets.QAction(MainWindow)

self.actionAbout.setObjectName("actionAbout")

self.menuHelp.addAction(self.actionAbout)

self.menubar.addAction(self.menuHelp.menuAction())

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

#Slots

self.pushButton.clicked.connect(self.getFile)

self.pushButton\_2.clicked.connect(self.encode)

self.pushButton\_3.clicked.connect(self.decode)

self.checkBox.stateChanged.connect(lambda: self.lineEdit\_2.setEchoMode(QtWidgets.QLineEdit.Normal) if self.checkBox.isChecked() else self.lineEdit\_2.setEchoMode(QtWidgets.QLineEdit.Password))

self.checkBox\_2.stateChanged.connect(lambda: self.lineEdit\_3.setEchoMode(QtWidgets.QLineEdit.Normal) if self.checkBox\_2.isChecked() else self.lineEdit\_3.setEchoMode(QtWidgets.QLineEdit.Password))

#Menu action

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate("MainWindow", "Стеганография"))

self.label\_4.setText(\_translate("MainWindow", "<html><head/><body><p><span style=\" font-weight:600;\">Шаг 1:</span></p></body></html>"))

self.label\_3.setText(\_translate("MainWindow", "Файл входного изображения:"))

self.pushButton.setText(\_translate("MainWindow", "Выберите файл"))

self.label.setText(\_translate("MainWindow", "<html><head/><body><p align=\"center\"><span style=\" font-size:20pt; font-weight:600; color:#5500ff;\">Шифровать</span></p></body></html>"))

self.label\_2.setText(\_translate("MainWindow", "<html><head/><body><p align=\"center\"><span style=\" font-size:20pt; font-weight:600; color:#5500ff;\">Расшифровать</span></p></body></html>"))

self.label\_5.setText(\_translate("MainWindow", "Введите текст, чтобы скрыть:<html><head/><body><p><span style=\" font-weight:600;\">Шаг 2:</span></p></body></html>"))

self.label\_7.setText(\_translate("MainWindow", ""))

self.label\_8.setText(\_translate("MainWindow", "<html><head/><body><p><span style=\" font-weight:600;\">Шаг 3:</span></p></body></html>"))

self.label\_9.setText(\_translate("MainWindow", "Введите пароль:"))

self.checkBox.setText(\_translate("MainWindow", "Показать пароль"))

self.label\_11.setText(\_translate("MainWindow", "Загрузка:"))

self.pushButton\_2.setText(\_translate("MainWindow", "Кодировать и сохранять"))

self.label\_6.setText(\_translate("MainWindow", "<html><head/><body><p><span style=\" font-weight:600;\">Шаг 2:</span></p></body></html>"))

self.label\_10.setText(\_translate("MainWindow", "Введите пароль:"))

self.checkBox\_2.setText(\_translate("MainWindow", "Показать пароль"))

self.pushButton\_3.setText(\_translate("MainWindow", "Расшифровать"))

self.label\_13.setText(\_translate("MainWindow", "Загрузка:"))

self.label\_12.setText(\_translate("MainWindow","Декодированные данные:"))

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv)

MainWindow = QtWidgets.QMainWindow()

ui = Ui\_MainWindow()

ui.setupUi(MainWindow)

MainWindow.show()

sys.exit(app.exec\_())