

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

|  |  |  |
| --- | --- | --- |
|  | | |
| И.о. зав. кафедрой | | «КБИС» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | Д.А. Короченцев |
| подпись | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. | |

**ВЫПУСКНАЯ** **КВАЛИФИКАЦИОННАЯ РАБОТА**

Тема: «РЕАЛИЗАЦИЯ ПРОГРАММНОГО СРЕДСТВА ОБРАБОТКИ ВИДЕОПОТОКА СТЕГАНОГРАФИЧЕСКИМ МЕТОДОМ»

Специальность 10.05.01 Компьютерная безопасность

Специализация Математические методы защиты информации

Обозначение ВКР 10.05.01.220000.000 Группа ВКБ62

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.С. Коробов

подпись, дата

Руководитель ВКР \_\_\_\_\_\_\_\_\_\_\_\_\_\_ доцент О.В. Куликова

подпись, дата

Нормоконтроль \_\_\_\_\_\_\_\_\_\_\_\_\_\_ доцент Р.В. Егорова

подпись, дата

Ростов-на-Дону

2022



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
 «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

|  |  |  |
| --- | --- | --- |
|  | | |
| И.о. зав. кафедрой | | «КБИС» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | Д.А. Короченцев |
| подпись | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_г. | |

**ЗАДАНИЕ**

на выполнение выпускной квалификационной работы

Тема: «РЕАЛИЗАЦИЯ ПРОГРАММНОГО СРЕДСТВА ОБРАБОТКИ ВИДЕОПОТОКА СТЕГАНОГРАФИЧЕСКИМ МЕТОДОМ»

Обучающийся Коробов Вячеслав Сергеевич

Обозначение ВКР 10.05.01.220000.000 Группа ВКБ62

Тема утверждена приказом по ДГТУ от «10» января 2022 г. №10-ЛС-О

Срок представления ВКР к защите «\_\_» \_\_\_\_\_\_\_ 2022 г.

Исходные данные для выполнения выпускной квалификационной работы:

Научные статьи и литература по стеганографии, стеганографическим алгоритмам и методам.

Материалы исследований, проводимых в ходе преддипломной практики.

|  |
| --- |
| Содержание выпускной квалификационной работы |
| Введение: |
| Во введении необходимо: изложить актуальность выбранной темы, обозначить объект и предмет исследования, цель и задачи выпускной квалификационной работы, теоретическую и практическую значимость работы, структуру работы. |
| Наименование и краткое содержание разделов: |
| 1 Теоретические сведения.  Основные термины и определения. Методы стеганографии. Анализ аналогов разрабатываемого программного средства. |
| 2 Алгоритмическая реализация программного средства  Общая концепция программного средства. Алгоритмы встраивания и извлечения информации. Метод внедрения информации в видеопоток и использование ключа. |
| 3 Программная реализация программного средства |
| Описание средств разработки. Структура программного средства. Демонстрация работы программного средства. |
| Заключение: |
| Основные выводы о проделанной работе, оценка достижения цели.  Перечень графического и иллюстративного материалов: |
| Презентация выпускной квалификационной работы на тему «Реализация программного средства обработки видеопотока стеганографическим методом». |

|  |  |  |
| --- | --- | --- |
| Руководитель ВКР | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | доцент О. В. Куликова |
|  |  |  |
| Задание принял к исполнению | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | В. С. Коробов |

**Аннотация**

Данная работа направлена на разработку программного средства, которое позволяет скрывать информация в видеопотоке. В выпускной квалификационной работе приводятся примеры уже существующих на рынке аналогов программных продуктов, способных обрабатывать видеофайлы стеганографическим методом. В виде блок-схем представлены детали разрабатываемого программного средства. Приведена демонстрация работы программы, в результате которой можно сделать вывод о применимости программного средства.

Объем работы – 50, количество иллюстраций – 15, таблиц – 1, приложений – 2, использовано источников – 7.

**Annotation**

This work is aimed at developing a software tool that allows you to hide information in the video stream. In the final qualifying work, examples of software products already existing on the market that are capable of processing video files by the steganographic method are given. The details of the software being developed are presented in the form of flowcharts. A demonstration of the program's operation is given, as a result of which it is possible to draw a conclusion about the applicability of the software tool.

The volume of textual material - 50, the number of illustrations - 15, tables - 1, appendices - 2, sources used - 7.

**Содержание**

Введение 5

1 Теоретические сведения 7

1.1 Основные термины и определения 7

1.2 Классификация стеганографии 8

1.3 Методы стеганографии 11

1.4 Атаки на стеганографические системы 13

1.5 Анализ существующих программных средств 15

2 Алгоритмическая реализация программного средства 20

2.1 Алгоритм встраивания информации в видеопоток 20

2.2 Алгоритм извлечения информации из видеопотока 21

2.3 Метод внедрения информации в видеопоток 23

2.4 Использование ключа 26

3 Программная реализация программного средства 29

3.1 Средства разработки 29

3.2 Структура программного средства 33

3.3 Демонстрация работы программного средства 34

Заключение 39

Перечень использованных информационных ресурсов 40

Приложение А Техническое задание 41

Приложение Б Листинг 45

# Введение

На сегодняшний день в интернете можно найти множество разного мультимедийного контента, к которому относятся цифровые аудиофайлы, изображения, а также видеофайлы. С помощью таких файлов возможно скрыть различную информацию, использовав такой контент как контейнер. Стеганография – специальная наука, которая изучает разные методы сокрытия информации в объектах мультимедии, которая должна оставаться необнаружимой как статистически, так и для человеческого восприятия. Другими словами, стеганография скрывает сам факт того, что какая-либо информация скрыта [1].

Каждый из методов стеганографии имеет основной перечень качественных характеристик. В основном можно выделить такие характеристики, как объём, устойчивость, а также невидимость. Свойство невидимости определяет, насколько хорошо спрятана информация, что её нельзя обнаружить ни человеческим глазу, ни одним из статистических методов. Характеристика устойчивости определяет, насколько стоек контейнер к ошибкам, а именно влияние различных модификаций контейнера на внедрённую в него информацию. Модификация здесь означает возможное применение обработки с помощью различных фильтров, обрезки, изменения размеров контейнера. Объём определяет, какое максимальное количество информации возможно скрыть в контейнере мультимедийного файла. Наибольший объём информации можно внедрить в видеозаписи, так как имеют наибольший размер, занимаемый в памяти.

Актуальность данной темы напрямую связана с информационной безопасностью и защитой информации.

Целью выпускной квалификационной работы является разработка программного средства, позволяющего скрыть информацию в видеопотоке.

Для достижения поставленной цели необходимо решить следующие задачи:

* изучить теоретические сведения стеганографии.;
* проанализировать аналоги программного средства;
* разработать оптимальный комплекс модулей для обработки видеофайлов методом стеганографии;
* программная реализация.

Объект исследования – процесс обработки видеопотока стеганографическим методом.

Предмет исследования: алгоритмы и методы, позволяющие скрыть информацию в видеопотоке.

Дипломная работа состоит из трех глав: теоретические основы, алгоритмическое конструирование программного средства, демонстрация работы программного средства.

После основной части выпускной квалификационной работы присутствуют разделы такие как «Заключение» и «Перечень использованных информационных ресурсов». Так же в конце представлены два приложения с техническим заданием и листингом программы.

**1 Теоретические сведения**

**1.1 Основные термины и определения**

Стеганография – способ передачи или хранения информации с учётом сохранения в тайне самого факта такой передачи (хранения) [4]. Этот термин был введён аббатом бенедиктинского монастыря Св. Мартина в Шпонгейме Иоганном Тритемием в своём трактате «Стеганография» в 1499 году, зашифрованном под магическую книгу.

Стеганография не скрывает содержимое тайного сообщения, как это делает криптография, а только скрывает сам факт его существования. При этом сообщение может выглядеть как судоку, картинка, письмо, статья или даже список покупок. Стеганография обычно используется вместе с различными методами шифрования. Такое дополнение, как правило, только увеличивает защиту скрытой в контейнере информации.

Сообщения, которые не привлекают к себе внимания – главное преимущество стеганографии над криптографией. Если шифрование сообщения никак не скрыто, то они могут вызвать подозрение и бывают сами по себе уличающими в тех странах, в которых криптография запрещена. То есть стеганография защищает сам факт наличия какой-либо скрытой информации, а криптография защищает само содержание сообщения.

Стеганографическая система (стегосистема) – объединение методов и средств, используемых для создания скрытого канала для передачи информации [5]. При создании такой стегосистемы были оговорены некоторый условия. Во-первых, враг представляет работу стеганографической системы. Противник не знает ключ, использующийся для раскрытия факта существования скрытой информации и её содержание. Во-вторых, если злоумышленник смог обнаружить наличие сообщения в контейнере, он всё равно не должен смочь извлечь скрываемую информация без знания ключа. В-третьих, враг не должен иметь технических и прочих преимуществ.

Сообщение – общее название передаваемой скрытой информации, будь то лист с надписями молоком, голова раба или цифровой файл [5].

Контейнер – любая информация, используемая для сокрытия тайного сообщения [5].

Пустой контейнер – контейнер, не содержащий секретного послания [5].

Заполненный контейнер (стегоконтейнер) – контейнер, содержащий секретное послание [5].

Стеганографический канал (стегоканал) – канал передачи стегоконтейнера [5].

Ключ (стегоключ) – секретный ключ, нужный для сокрытия стегоконтейнера [5]. Есть два типа ключей, используемых в стеганографии: закрытые (секретные) и открытые. В стегосистеме, использующей секретный ключ, он должен быть создан или перед началом обмена сообщениями, или передан по защищённому каналу. Если стегосистема использует открытый ключ, она должна работать так, чтобы получить из него закрытый ключ было невозможно. В таком случае открытый ключ может быть передан по открытому незащищённому каналу.

**1.2 Классификация стеганографии**

В конце 1990-х годов было выделено три направления стеганографии:

* классическая;
* компьютерная;
* цифровая.  
   Классическая стеганография. Самый распространённый пример для классической стеганографии – писание симпатическими чернилами, которые ещё называют «невидимыми». При использовании таких чернил для написания текста, его можно увидеть только при соблюдении определённых условиях, такие как освещение, нагрев, химический проявитель и др. Филон Александрийский изобрёл эти чернила ещё в первом веке нашей эры. Но их продолжали использовать и в средневековье, так и в новейшее время. В пример можно привести их применение русскими революционерами для написания писем из тюрем. Ещё одним примером является использование молока для написания скрытого письма. В таком случае скрытый текст можно было прочесть только при нагреве от пламени свечи. По рассказу, который называется «Рассказы о Ленине», который изучали ещё советские школьники, таким способом часто пользовался Владимир Ленин.

Ещё одним примером симпатических или «невидимых» чернил можно назвать чернила с нестабильным пигментом. Текст, написанный с их помощью, первоначально выглядит так, будто он был написан обыкновенной шариковой ручкой. Но через некоторое нестабильный пигмент разлагается, в результате чего пропадает и написанный текст. Недостатком можно назвать то, что после использования данного способа скрытый текст возможно восстановить с помощью обычной ручкой. Но он устраняется использованием мягкого пишущего узла, подобного фломастеру.

Ещё одним примером является стеганография, основанная на особенностях компьютерной системы. Это направление классической стеганографии называется компьютерной стеганографией [6]. Примером может являться текстовая стеганография, скрытие данных в неиспользуемых областях форматов файлов, подмена символов в названиях файлов. В Linux существует стеганографическая файловая система StegFS. Рассмотрим несколько примеров.

В качестве первого примера приведём метод, основанный на использовании зарезервированных полей компьютерных форматов файлов. Этот метод основан на том, что часть поля расширений не используется для сохранения информации о расширении и в стандартном режиме остаётся заполненным нулями. Поэтому возможно записать в эту часть скрытую информацию. Но этот метод имеет такие недостатки, как малый уровень скрытности и низкий объём для внедрения данных.

Следующим методом рассмотрим возможность скрытия данных в тех местах гибких дисков, которые никак не используются. Таким местом может являться, например, нулевая дорожка на диске. Небольшая производительность и малый объём для скрытия информации являются основными недостатками данного метода.

Другой метод компьютерной стеганографии – возможность скрытия информации в полях форматов, используя их свойства, невидимые для пользователя, например, «невидимые» поля для получения сносок или указателей, печать белым шрифтом на белом фоне. Также как и в предыдущем методе, недостатки проявляются в малой производительности и небольшом объёме контейнера для передаваемых сообщений.

Ещё один пример методов заключается в использовании особенностей файловых систем. Во многих файловых системах файл занимает целое количество кластеров. Например, в файловой системе FAT32 размер кластера составляет 4 килобайта (КБ). Поэтому для хранения любого файла выделяется минимум 4 КБ, остальное же место может использоваться для скрытия информации. К недостаткам метода относится лёгкость обнаружения.

Последним из трёх направлений рассмотрим цифровую стеганографию. Она заключается в скрытии информации в различные цифровые объекты [5]. К таким объектам могут относиться мультимедийный файлы, такие как видеозаписи, аудиофайлы, изображения, 3D-объекты с помощью различных методов. Как правило, подобные файлы могут вместить в себя достаточно большое количество скрытых данных. Но подобное преобразование всегда сопровождается той или иной степенью искажения, которое снижает маскировку. Но если вносить только такие искажения, которые находятся ниже порога чувствительности органов зрения и слуха среднестатистического человека, то такие изменения будут практически незаметны.

Основными направлениями цифровой стеганографии считаются:

* встраивание идентификационных номеров;
* скрытая передача данных;
* встраивание заголовков;
* ЦВЗ (цифровые водяные знаки).

В настоящий момент цифровая стеганография имеет следующее практическое применение:

* скрытое хранение информации;
* недекларированное хранение информации;
* защита подлинности документа;
* индивидуальный отпечаток в системе электронного документооборота (СЭДО);
* скрытая передача управляющего сигнала;
* защита авторских прав;
* защита исключительного права;
* «камуфлирование» программного обеспечения (ПО);
* защита конфиденциальной информации от несанкционированного доступа (НСД);
* преодоление систем мониторинга и управления сетевыми ресурсами;
* незаметная передача информации;
* создание скрытых от законного пользователя каналов утечки информации.

**1.3 Методы стеганографии**

Все алгоритмы внедрения скрытой информации в цифровые файлы разделяются на подгруппы.

Алгоритмы, которые работают с самим цифровым сигналом. Например, метод наименее значащих бит (НЗБ или LSB).

«Впаивание» скрытых данных. Имеется ввиду наложение скрываемой информации (часто наложение изображения или звука) поверх оригинального файла. Данный метод в основном нужен для внедрения цифровых водяных знаков (ЦВЗ).

Следующим методом является встраивание на основе особенностей форматов цифровых файлов. В пример можно привести скрытие сообщения в метаданные или поля файлов, которые зарезервированы, но не используются.

Существуют линейные (аддитивные), нелинейные и другие алгоритмы стеганографии, отличающихся по способу внедрения данных в цифровые файлы. Линейные стегоалгоритмы встраивания сообщения основаны на линейной модификации оригинального изображения, а извлечение же сообщения происходит с помощью корреляционных методов. В таком случае цифровой водяной знак складывается с изображением, которое является контейнером. Нелинейные алгоритмы, в отличие от аддитивных, основаны на скалярном или векторном квантовании. Существуют также такие алгоритмы, которые используют методы фрактального кодирования изображений.

Внедрение скрываемого сообщения в изображения. Именно в файлы изображения в большинстве случаев внедряются скрытые данные. На это имеются следующие веские причины:

* практическая значимость задачи защиты фотографий, картин, видео и прочей графической информации от незаконного копирования и распространения;
* изображение как контейнер имеет большой объём вместимости данных, что очень удобно для внедрения скрываемой информации;
* заранее известно максимальное количество информации, которое можно встроить в контейнер изображения;
* нет таких ограничений, как внедрение информации в режиме реального времени, как, например, в видеофайлах;
* большинство изображений имеют такие области, которые имеют шумовую структуру и хорошо могут подходить для внедрения данных;
* анализ таких стегосистем начинается, обычно, с визуальной оценки, однако человеческое зрение не может заметить небольшое изменение оттенка цветов, обусловленное внедрением данных в битовое представление цветов, что оставляет такой канал передачи информации без подозрений.

Метод LSB. В случае применения этого метода происходит специальное преобразование контейнера. Это преобразование основано на замене наименее значащих бит

Перейдём к сути данного метода. Допустим, у нас есть 24-битное изображение в цветовой модели красного, зелёного и синего. Тогда один пиксель такой будет представлен в виде последовательности 24-бит или трёх байт, каждый из которых описывает свой цвет, например: 00000000 00000000 00000000. В результате сложения этих байтов получается итоговый цвет. В данном примере получился чёрный цвет. Теперь попробуем внедрить в этот пиксель какую-нибудь информацию, например, символ» «S». Он состоит из одного байта или восьми бит 01000100. Разобьём этот байт на три примерно равные части, получим 010 00 100. Теперь каждой частью заменим последние биты каждого цвета в пикселе. В итоге получим следующие данные пикселя: 00000010 00000000 00000100. Пиксель, очевидно, исказился и изменился в оттенке, но такое малое изменение не будет заметно для человеческого глаза.

**1.4 Атаки на стеганографические системы**

Атака на стеганографическую систему – это попытка злоумышленника обнаружить, извлечь, изменить скрытую информацию в контейнере [4]. Эти атаки называются стегоанализом. Одной из простых атак на стегосистему является субъективная атака. Она заключается во внимательном «внешнем» изучении цифровых файлов. Например, рассматривается изображение на наличие искажений, прослушивается аудиофайл или изучается видеозапись в попытках обнаружить признаки наличия в этих файлах скрытой информации. Данная атака будет успешной только в случае, если стеганографическая система никак не защищена и в большинстве случаев является первым этапом при анализе стегосистемы. Также существуют следующие типы атак:

* атака по известному встроенному сообщению;
* атака на основе выбранного скрытого сообщения;
* атака на основе выбранного заполненного контейнера;
* атака на основе известного пустого контейнера;
* атака по известному заполненному контейнеру;
* атака на основе выбранного скрытого сообщения;
* атака на основе выбранного пустого контейнера;
* атака по известной математической модели контейнера.
* адаптивная атака на основе выбранного скрытого сообщения.

Рассмотрим некоторые из этих атак подробнее. Атака по известному заполненному контейнеру заключается в том, что у злоумышленника есть в наличии одно или несколько стего. Если их несколько, то это в большинстве случаев значит, что встраивание скрытого сообщения в контейнер проводилось одним и тем же способом. В таком случае злоумышленнику необходимо распознать факт наличия скрытой информации, получить к нему доступ или определить ключ, с помощью которого появится возможность извлекать другие скрываемые данные.

Атака по известной математической модели контейнера заключается в сравнении предполагаемого контейнера с известной ему моделью.

Следующей рассматриваемой атакой является атака на основе известного пустого контейнера. Для применения такой атаки взломщик должен знать пустой контейнер. При сравнении этого контейнера с подозрительным, можно обнаружить факт наличия в нём скрытой информации.

**1.5 Анализ существующих программных средств**

Рассмотрим существующие аналоги разрабатываемого программного средства сокрытия информации в видеопотоке. Аналогами являются те программы и утилиты, главный функционал которых полностью или частично совпадает с разрабатываемой. То есть они должны иметь возможность скрывать какую-либо информацию в видеофайлах, а также уметь извлекать её. Поэтому после их исследования будет составлен перечень плюсов и минусов каждой из них.

Первая рассматриваемая программа – OpenPuff. Её интерфейс представлен на рисунке 1.

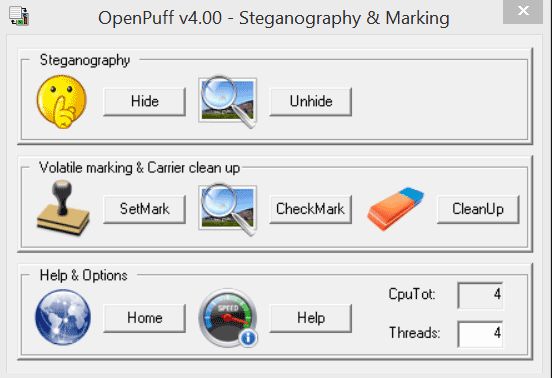


Рисунок 1 – основное окно OpenPuff

Данное программное средство вполне удобно для использования, и выполняет все необходимые нам функции. Также в нем имеются дополнительные функции, например, отдельное скрытие информации в изображениях.

Главным же недостатком программы является сильное увеличение итогового файла по сравнению с исходным. Это существенный недостаток, так как по большой объём данных файла ухудшает главную особенность стеганографии – скрытность передаваемой информации.

Следующий рассматриваемый программный продукт - MSU StegoVideo, интерфейс которого можно увидеть на рисунке 2.



Рисунок 2 – интерфес MSU StegoVideo

Особенностью данной программы является помехоустойчивое кодирование (сверточный код с декодером Витерби), что в результате обеспечивает достаточно высокую вероятность восстановления информации из сжатого видео.

Главный недостаток программного средства MSU StegoVideo – достаточно медленная скорость работы и поддержка только одного формата видеофайла – AVI (Audio Video Interleave).

Рассмотрим ещё одну программу – StegoStick. Программное средство позволяет скрывать не только текст, но и файлы в других файлах. Помимо этого, оно имеет возможность зашифровать исходную информацию одним из трёх алгоритмов шифрования на выбор, таких как DES (Data Encryption Standard), Тройной DES и RSA (аббревиатура от фамилий Rivest, Shamir и Adleman). Также StegoStick имеет понятный и удобный интерфейс, представленный на рисунке 3.

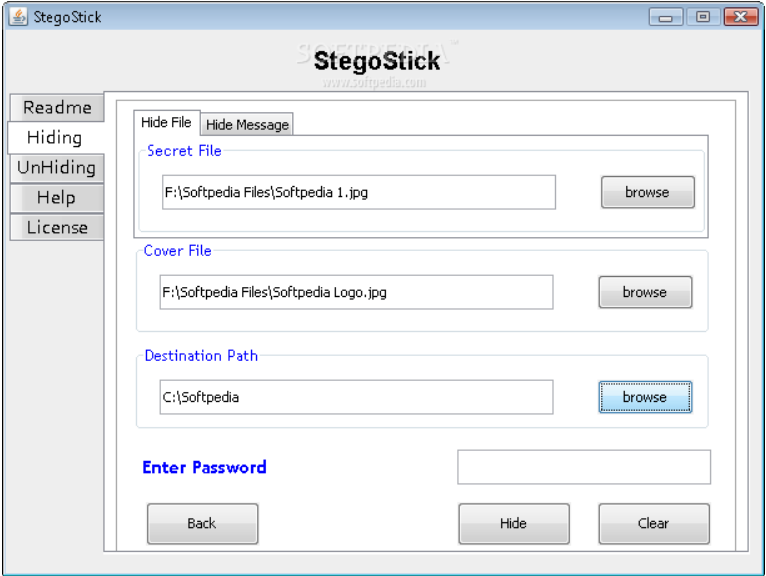


Рисунок 3 – интерфейс StegoStick

Недостатком является необходимость наличия предустановленной библиотеки java runtime library, без которой программа не будет работать. Также из видеофайлов поддерживается только формат AVI.

Таким образом, в результате анализа существующих программных средств сокрытия информации в видеопотоке для сравнения были выделены такие критерии работы программ, как разница размера итогового и исходного файлов, скорость работы, необходимость установки дополнительного ПО и поддерживаемые форматы видеофайлов.

Первый критерий – разница размера итогового и исходного файлов, определяет то, насколько скрытен сам факт наличия встроенной информации в видеопотоке. Представим, что потенциальный злоумышленник каким-либо образом перехватил видеофайл со скрытой информацией. Если его размер несоизмерим с длительностью и качеством видео, то враг может заподозрить присутствие в видеопотоке избыточной информации. Поэтому чем больше эта разница, тем более вероятно то, что скрываемая информация может быть обнаружена посторонним лицом.

Критерий скорости работы определяет, сколько времени может потребоваться для процесса встраивания информации в видеофайл и получения итогового файла. Также влияет на время извлечения исходной информации из обработанного видеопотока.

Необходимость наличия дополнительного программного обеспечения (ПО) может требоваться для запуска или корректной работы программы. Если такого ПО нет, пользователю необходимо будет его установить, что вызывает неудобство эксплуатации программного средства.

Самый последний критерий определяет, какие форматы видеофайлов могут использоваться для сокрытия информации в них.

Итоговую таблицу критериев можно увидеть на таблице 1.

Таблица 1 – Критерии сравнения существующий программ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование  приложения      Критерий | OpenPuff | MSU StegoVideo | StegoStick | VideoSteg (разрабатываемое программное средство) |
| Разница размера итогового и исходного файлов | Большая | Малая | Малая | Малая |
| Скорость работы | Быстро | Медленно | Быстро | Быстро |
| Необходимость установки дополнительного ПО | Нет | Нет | Да | Нет |
| Поддерживаемые форматы видеофайлов | MP4, MPG, VOB | AVI | AVI | AVI, MKV, MP4 |

В результате видно, что разрабатываемое программное средство, по сравнению с аналогами не имеет минусов ни по одному из критериев.

**Вывод по главе**

В данной главе были рассмотрены теоретические основы стеганографии, классификация, её алгоритмы и методы. В результате для работы был выбран метод наименее значащих битов. Выделены и описаны возможные атаки на стеганографические системы. Также был проведен анализ аналогов разрабатываемого программного средства, в результате которого был составлен перечень их плюсов и минусов для сравнения с VideoSteg.

**2** **Алгоритмическая реализация программного средства**

**2.1 Алгоритм встраивания информации в видеопоток**

Для дальнейшей разработки программного средства была разработана оптимальная модель его работы. В основу работы программы входят следующие два алгоритма – встраивания и извлечения информации.

На рисунке 4 представлен алгоритм встраивания информации.

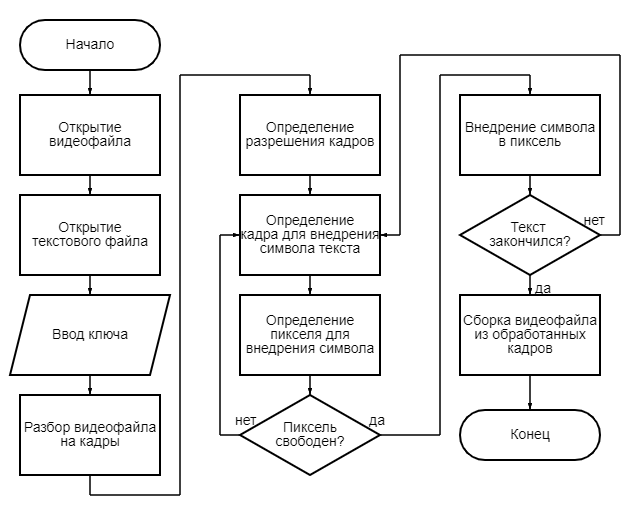


Рисунок 4 – Алгоритм процесса встраивания

Алгоритм состоит в следующем: мы выбираем видеофайл, в который хотим скрыть информацию. Первым делом программа вычисляет количество кадров, содержащееся в исходном файле. После этого вычисляется количество кадров в секунду выбранного видеофайла. Данное значение запоминается для дальнейшей сборки итогового видеопотока. Далее исходное видео разбивается на кадры. Полученные изображения нумеруются. Здесь работа с видеофайлом завершается, так как все необходимые данные с него получены.

Следующий этап начинается с вычисления разрешения кадра на основе одного из полученных с видеофайла. После этого для программа принимает от пользователя текстовой документ с исходным текстом, который необходимо внедрить в видеопоток, затем принимается закрытый ключ.

Затем, после обработки всех полученных данных, программа начинает процесс заполнения контейнеров, которыми являются пиксели в кадрах видеофайла, исходной информацией. Данный процесс начинается с поиска свободного пикселя в видеокадрах.

Поиск основывается на генерации псевдослучайных чисел. Последовательность этих чисел зависит от «зерна». Тут и используется введённый пользователем программы ключ – он является этим самым «зерном» для генерации чисел. Данные числа же служат координатами пикселей, используемых для внедрения информации.

После нахождения очередного пикселя производится проверка, не занят ли уже он символом исходного текста. Если пиксель свободный, то он заполняется символом исходного текста, затем его координаты сохраняются в специальном массиве, который и используется каждый раз при проверке заполненности очередного пикселя.

Когда все символы исходного текста внедрены, в последний пиксель записывается символ с нулевым кодом. Из полученных изображений вновь собирается и сохраняется видео, которое уже будет содержать в себе скрытую информацию. При сборке используется сохранённое на первом этапе значение количества кадров в секунду у исходного видео. На этом моменте заканчивается кодирование – первый основной алгоритм работы программы.

**2.2 Алгоритм извлечения информации из видеопотока**

Теперь опишем второй основной алгоритм программного средства – извлечение, то есть получение скрытого текста из видеофайла. Данный алгоритм представлен на рисунке 5.

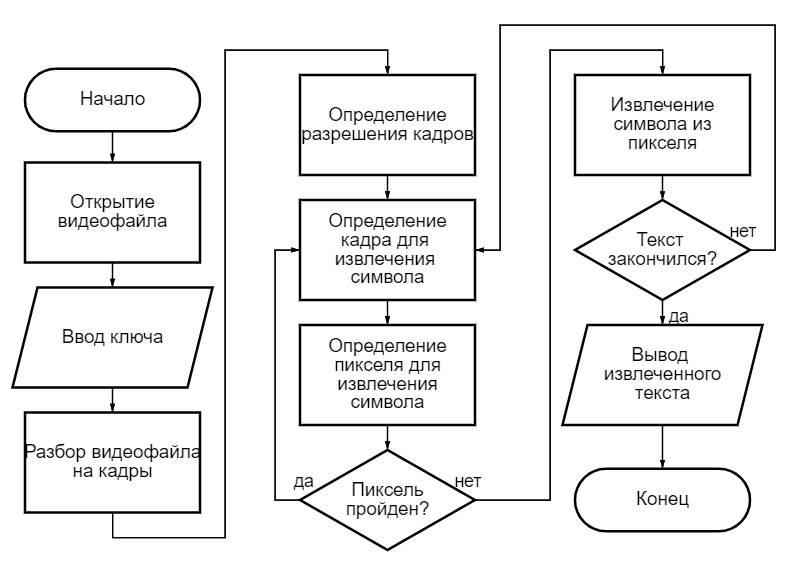


Рисунок 5 – Алгоритм процесса декодирования

Алгоритм начинается с выбора видеофайла, с которого необходимо получить скрытую в нём информацию. Вычисляется количество кадров, после чего видео разбивается на изображения, которые, как и в первом алгоритме, также нумеруются. Работа с видеофайлом завершается.

Далее вычисляется размер полученных изображений и принимается введённый пользователем закрытый ключ.

Затем начинается процесс поиска пикселей, в которых содержится символы скрытого текста. Аналогично алгоритму кодирования поиск происходит с помощью псевдослучайных чисел, в которых зерном является полученный закрытый ключ. Таким образом, если ключ, использованный для извлечения скрытого текста из видео, будет отличаться от ключа, с помощью которого происходило внедрение текста в алгоритме кодирования, получить скрытую информацию в обработанном видео не получится, так как в таком случае в процессе поиска будут найдены не те пиксели, в которых хранятся символы исходного текста.

Найдя очередной пиксель, происходит проверка, проходил ли цикл поиска через него до этого. Если да, то поиск происходит дальше, иначе с данного пикселя извлекается символ. Процесс поиска происходит до тех пор, пока в очередном пикселе не обнаружится нулевой символ, означающий конец скрытого текста.

В конце алгоритма декодирования все извлеченные символы сохраняются в отдельном текстовом файле. Полученный текст выводится на поле вывода программы, после чего алгоритм декодирования завершается.

**2.3 Метод внедрения информации в видеопоток**

Под понятием «пиксель», во-первых, понимают физический элемент матрицы дисплея, то есть экран монитора, телевизора, телефона и т.д. Во-вторых, это понятие также используется в описании наиболее маленькой цветовой точки изображения. Из множества таких цветовых точек и образуется полноценное изображение.

Данное понятие может использоваться корректно в этих обоих случаях, так как принцип работы у них в целом один и тот же.

Цвет пикселя получается из трёх цветовых элементов – красным, зелёным и синим, зависящий от их значений.

Red, green, blue (RGB) – это аддитивная цветовая модель. Она описывает метод преобразования цвета для воспроизведения цвета из трёх основных. Красный, зелёный и синий цвета были выбраны в соответствии с особенностями человеческого зрения.

Теперь на рисунке 6 наглядно покажем результат смешивания цветов с помощью шкалы интенсивности каждого из элементов цвета. Зелёный цвет показывает заполненность шкалы в виде текущей интенсивности. В квадрате справа можно увидеть итоговый цвет – результат смешивания всех цветов. Так как значение каждого цвета на максимуме, получается белый цвет.

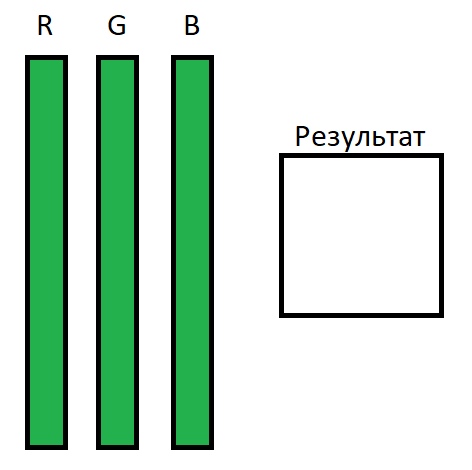


Рисунок 6 – Белый цвет RGB модели

Приведем пример для красного цвета, представленный на рисунке 7.

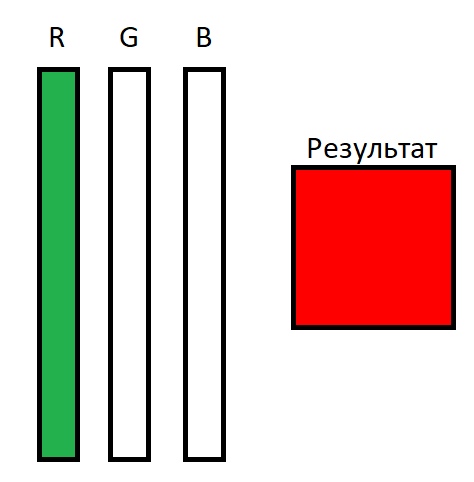


Рисунок 7 – Красный цвет RGB модели

Представим, что имеется максимальное значение интенсивности у каждого из цветов, в результате чего мы получаем белый цвет пикселя на дисплее. В случае, если понизить значения интенсивности зелёного и синего цветов до нуля, то в результате останется только красный цвет максимальной яркости.

Диапазон интенсивности – это диапазон значения цветового элемента от его минимального или нулевого значения до его максимального значения.

В нашем случае каждый пиксель изображения состоит из 24 бит или 3-х байт, каждый из которых определяет интенсивность красного, зелёного и синего цветов соответственно. То есть диапазон интенсивности каждого цвета – от 0 до 255.

Каждый символ исходного текста состоит из 8 бит. Мы будем использовать кодировку по стандарту кодирования символов Unicode Transformation Format, 8-bit (UTF-8). Программа перемешивает эти биты символа псевдослучайным образом, после чего заменяет ими младшие биты каждого цвета в пикселе, как это представлено на рисунке 8.

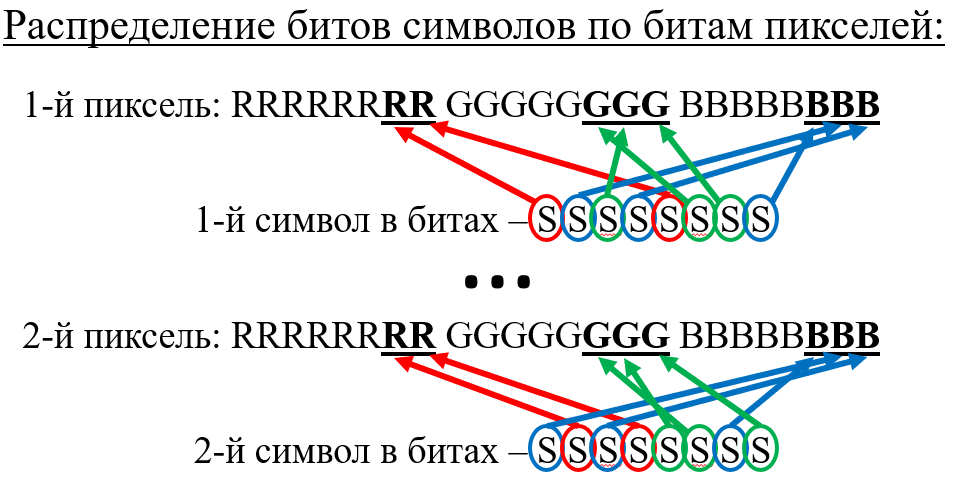


Рисунок 8 – распределение битов по пикселям

При этом с каждым очередным пикселем биты перемешиваются по-разному, что повышает защиту скрываемого текста. Так как изменение яркости красного цвета наиболее заметно для человеческого глаза, его байт изменен меньше остальных. Данная методика позволяет скрыть информацию так, что вызванное искажение изображений практически незаметно для человеческого зрения.

В случае, если у злоумышленника окажется доступ к исходному видеофайлу, он сможет сравнить его с обработанным. Таким образом, ему не сложно будет выявить те пиксели, в которых произошли изменения. Но ему предстоит ещё как-то выяснить правильную их очерёдность для декодирования. Помимо это, необходимо будет определить, как именно распределён каждый символ в соответствующем пикселе.

При этом задачу декодирования можно усложнить ещё больше. Для этого, перед началом внедрения битов символов исходного текста в пиксели, необходимо изменить все пиксели всех кадров видеопотока. Чтобы при этом не было заметно никаких искажений, изменять нужно также младшие биты каждого цвета. Для большей защиты, изменять их необходимо случайным образом, иначе возможно было бы выявить закономерность.

В результате, в случае обнаружения злоумышленником факта хранения информации в видеопотоке, её извлечение окажется невероятно сложной задачей.

**2.4 Использование ключа**

Для скрытия информации или её извлечения программа использует ключ, вводимый пользователем. С помощью данного ключа по определённому алгоритму программное средство определяет, где и как в видеопотоке будет скрыта информация. То есть происходит определение, в каких кадрах будет скрыта информация, в каких пикселях будут скрыты символы текста, и как именно будут перемешаны биты символов перед внедрением, что представлено на рисунке 9. Таким же образом ключ используется и для извлечения информации из видеопотока, определяя те же данные. Поэтому при применении одного и того же ключа для скрытия и извлечения информации данные совпадут, и будет извлечена скрытая информация. В противном случае вместо скрытого текста извлекутся значения пикселей, не несущих в себе никакой информации.

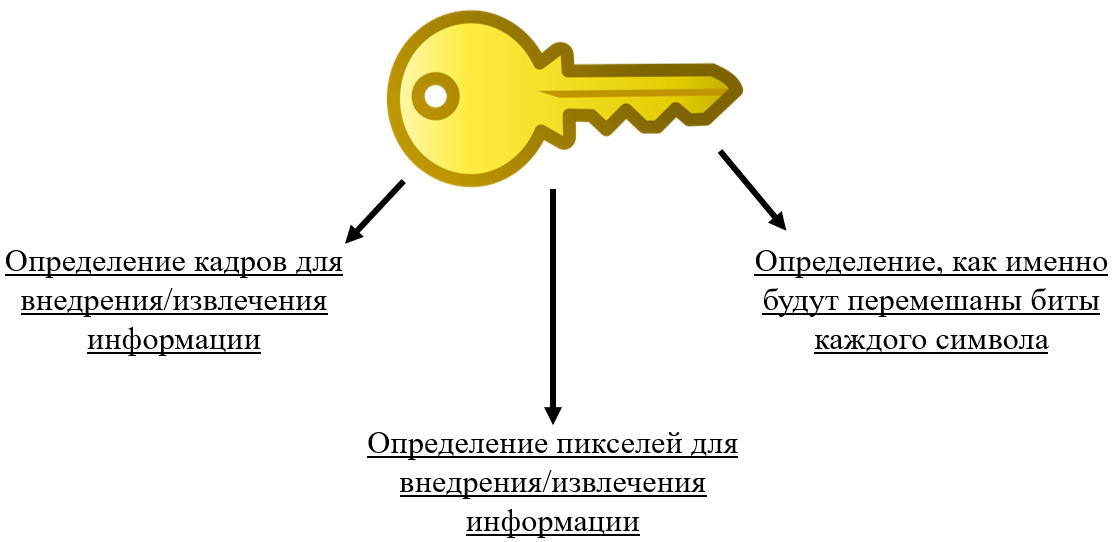


Рисунок 9 – Использование ключа

Алгоритмом, по которому вычисляются данные для внедрения информации, является генератор псевдослучайных чисел, который генерирует последовательность чисел. Получившиеся в результате числа будут практически независимы.

Ключ, который вводит пользователи в разрабатываемой программе, используется в качестве порождающего элемента для генератора псевдослучайных чисел. Таким образом, последовательность сгенерированных чисел будет отличаться от ключа к ключу. Благодаря данному свойству, потенциальный злоумышленник не сможет получить доступ к скрытой информации без знания ключа. В таком случае ему не поможет даже наличие у него данного разрабатываемого программного средства. Если же враг сможет получить исходный код программы одним из методов программных реализаций, например, методом дизассемблирования, он всё равно не сможет прочесть скрытую информацию в видеопотоке.

**Вывод по главе**

В данной главе были подробно описаны алгоритмы скрытия информации в видеопотоке и её извлечения из видеопотока. Приведён метод внедрения информации в видеофайл, а конкретно процесс записи символов исходного текста в пиксели кадров, полученных из видеофайла. Также пояснено использование ключа в программном средстве.

**3** **Программная реализация программного средства**

**3.1 Средства разработки**

Для написания разрабатываемого программного средства был использован язык программирования Python.

Python – это интерпретируемый высокоуровневый язык программирования. В качестве главных своих преимуществ выделяет такие качества, как читабельность кода и скорость его написания. Создатель сконструировал его таким образом, чтобы программисты с лёгкостью могли понимать и писать логичный и идеальный код для разработки как небольших программ и утилит, так и проектов крупных масштабов.

Python динамически типизируется и самостоятельно собирает мусор, то есть лишние данные, более не используемые в работе программного средства для освобождения памяти. Язык имеет поддержку таких методологий программирования, как структурированное, объектно-ориентированное и функционально. Так как он имеет очень большую базу встроенной библиотеки, его иногда называют лишь дополнением к этой базе.

Язык программирования Python начал разрабатываться в 1980-х годах. Создатель Гвидо ван Россум рассматривал его как преемником языка программирования ABC. Первая версия была представлена как Python 0.9.0. Спустя 9 лет, в 2000 году вышла новая версия языка 2.0. Главным нововведением было добавление таких функций, как распознавание списков и сбор мусора. Следующая версия 3.0 вышла в 2008 году, которая актуальна до сих пор. Поддержка Python 2 была прекращена с версии 2.7.18 в 2020 году.

Рассмотрев преимущества и возможности Python, было решено, что он отлично подходит для разработки программного средства.

PyCharm – интегрированная среда разработки для множества языков программирования, включая JavaScript, обе версии Python и большое количество различных фрэймворков [2]. Включает в себя такие средства и инструменты для разработки, как графический отладчик, анализ написанного кода, запуск элементных тестов и многое другое. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA.

PyCharm может использоваться на разных операционных системах, включающих Linux, Windows и macOS.

Благодаря таким обширным функциям, как мгновенная подсветка допущенных при написании кода ошибок, его анализ и автодополнение, удобная навигация по всему разработанному программному кода, рассматриваемая интегрированная программная среда обеспечивает продуктивную и быструю разработку программных продуктов.

PyCharm также включает в себя набор инструментов «из коробки», который имеет поддержку работы с базами данных, встроенный терминал, возможность удалённой разработки и многое другое.

Учитывая весь набор возможностей, предоставляемых средой разработки PyCharm, было решено использовать его.

В ходе разработки программного средства также были использованы такие библиотеки как OpenCV, Ffmpeg-python и Tkinter.

OpenCV – библиотека с открытым исходным кодом для машинного зрения и машинного обучения. Он имеет 2500+ оптимизированных алгоритмов - полный набор как классических, так и самых современных алгоритмов компьютерного зрения и машинного обучения. У него много реализаций, включая Python, Java, C++ и Matlab.

Вот некоторые функции библиотеки OpenCV:

* хранения математических функций и вычислений, алгебры и структур данных;
* хранения моделей для машинного обучения;
* ввода и вывода картинок или видео, чтения и записи в файл;
* обработки изображения;
* распознавания примитивов;
* обнаружение объектов – лиц, предметов и других.

В разрабатываемом средстве данная библиотека используется редактирования изображений и внедрения символов в пиксели.

FFmpeg – мощный набор различных инструментов и библиотек с открытым исходным кодом, позволяющих работать с видео, например, преобразовывать его в разные форматы с использованием большого количества кодеков на выбор, записывать видеопотоки и т.д.

С помощью данной библиотеки разрабатываемое программное средство разбирает видеофайл на кадры, а также собирает видеофайл из множества изображений.

Tkinter – графическая библиотека для написания и разработки графического пользовательского интерфейса на языке программирования Python. Является кроссплатформенной, благодаря чему может использоваться на различных операционных системах. Для его разработки использовался язык программирования Tcl.

После создания графического интерфейса с помощью библиотеки Tkinter получившиеся окно и его элементы будут выглядеть так же, как выглядят стандартные окна и графическое окружение операционной системы, в которой этот интерфейс запущен. Это отличный выбор для разработки графического интерфейса программных средств в Python, если нужный интерфейс не должен быть слишком сложным в визуальном плане. Так же его преимуществом является большая функциональность и скорость работы.

В рассматриваемую библиотеку входят следующие компоненты.

Toplevel/Tk – Основное окно программы.

Frame – Рамка. Помещает в себя окно и другие графические компоненты. Также организует их расположение в интерфейсе.

Label – Этикетка. Позволяет отображать текст или рисунки разного формата.

Entry – Поле, предназначенное для ввода текста. Содержит одну строку.

Text – Ещё одно поле для ввода текста пользователя. С помощью этого компонента можно также осуществлять вывод текста на экран, изменять его, использовать изображения.

Canvas – Холст. Позволяет отображать различные графические фигуры, например, линии, круги, овалы, квадраты. Также может показывать текст, рисунки и окна.

Button – Кнопка. По нажатию на неё начинается выполнение различных действий и вызов функций.

Radiobutton – Переключатель. Позволяет переключать значение переменных на другие значения, находящихся в одной группе. То есть имеется некоторый выбор среди нескольких доступных значений. Можно выбрать только одно из значений.

Checkbutton – Флаг. Похож на Radiobutton, но имеет возможность множественного выбора, предоставляя отдельную переменную на каждый экземпляр виджета.

Scale – Шкала с ползунком. С помощью ползунка можно листать доступные значения для выбора.

Listbox – Список. Похож на флаг checkbuttom, только в отличие от него выбор осуществляется между списками.

Menu – Меню. Организует работу различных элементов меню, такие как всплывающие и ниспадающие.

Menubutton – Кнопка-меню. Позволяет использовать ниспадающее меню.

Message – Сообщение. Данный компонент похож на Label, но в отличие от него позволяет менять размер всплывающего окна.

Также в состав входят OptionMenu, Spinbox, LabelFrame, PanedWindow.

Библиотека Tkinter, как видно из описания, используется для создания интерфейса разрабатываемой программы. Для тех задач, которые определены к нашему программному средству, библиотека подходит предостаточно.

**3.2 Структура программного средства**

Программное средство состоит из девяти модулей, реализующих все основные функции разрабатываемой программы. Модули следующие: gui, opentext, openvideo, openvideo2, enc, dec, split и mix. Представим описание функций модулей.

Модуль gui описывает графический интерфейс программного средства, с которым будет взаимодействовать пользователь.

Модуль opentext позволяет выбрать текстовой файл с информацией, которую необходимо скрыть в видеопотоке. Данный файл может находиться в любом каталоге операционной системы.

Модуль openvideo позволяет пользователю выбрать видеофайл, в который необходимо будет внедрить исходный текст.

Модуль openvideo2 похож на предыдущий модуль openvideo. Отличается тем, что с помощью него выбирается уже обработанное видео, из которого необходимо извлечь скрытую информацию.

Модуль enc является основным модулем для осуществления функции внедрения информации в видеофайл. Получив на вход текстовой файл и видеофайл, начинается их обработка. Модуль переводит значения пикселей в двоичные числа, обрабатывает символы исходного текста, осуществляет их слияние и сохраняет полученный результат.

Модуль dec – основной модуль функции извлечения скрытого текста из видеопотока. Он обрабатывает значения пикселей и поочередно извлекает символы скрытого текста. По окончанию выводит результат на экран и в отдельный текстовой файл.

Модуль split является инструментом для разбивки видеофайла на отдельный кадры. Используется как модулем enc, так и модулем dec.

Модуль mix в противоположность модуля split собирает видеопоток из уже обработанных изображений. Используется в модуле enc.

**3.3 Демонстрация работы программного средства**

На рисунке 10 можно увидеть интерфейс разработанного программного средства.

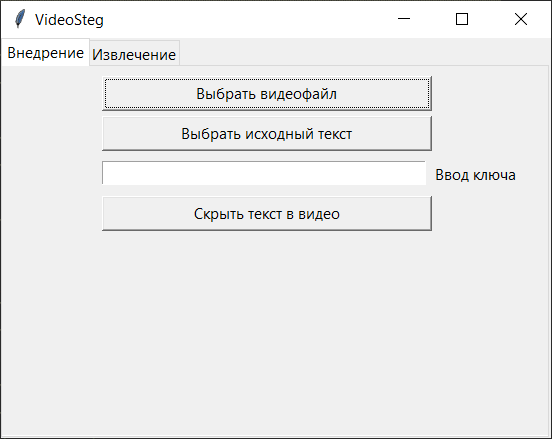


Рисунок 10 – Интерфейс программы

Интерфейс состоит из 2 вкладок - внедрение и извлечение. Первая вкладка используется для процесса скрытия информации в видеопотоке. Здесь можно выбрать видеофайл, в который необходимо внедрить скрытую информацию. Пример представлен на рисунке 11.

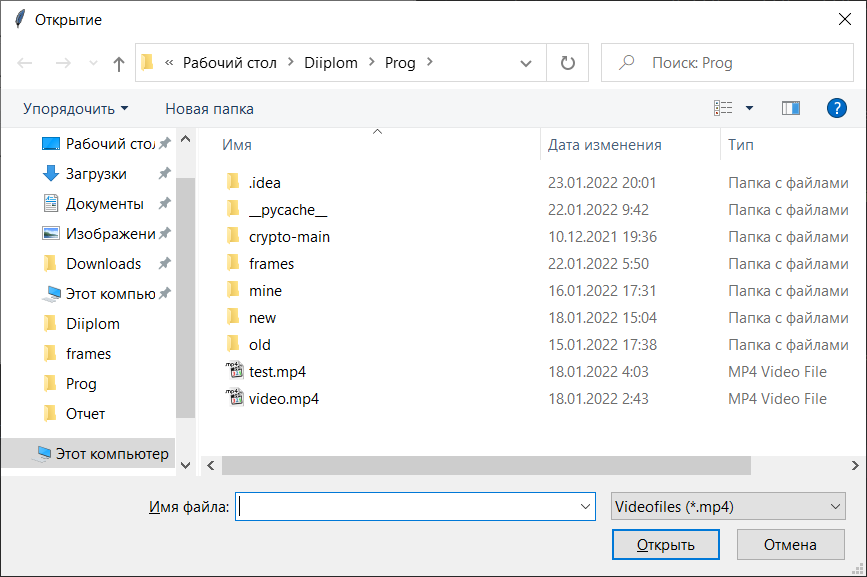


Рисунок 11 – выбор видеофайла

На этой же вкладке есть возможность выбора исходного текста, который нужно скрыть в видеофайле. В качестве исходного текста можно выбрать, например, документ в формате txt, что можно увидеть на рисунке 12.

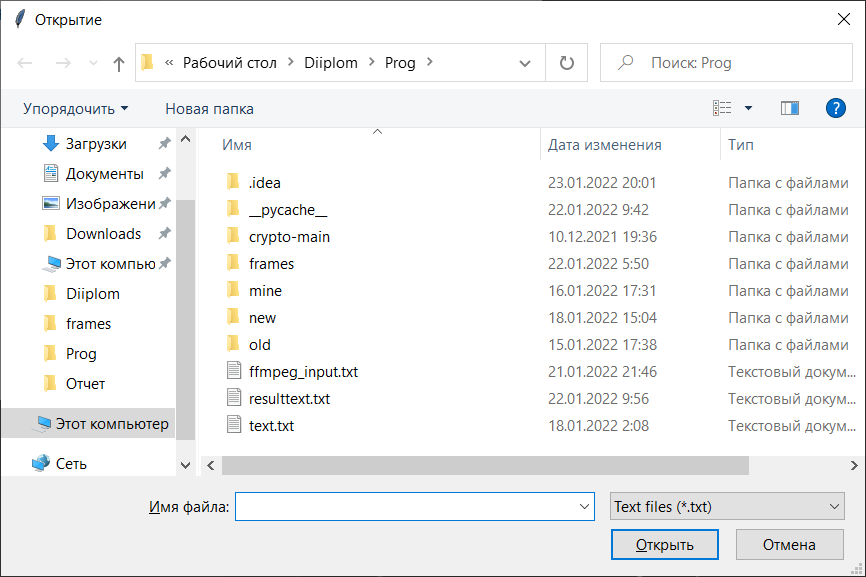


Рисунок 12 – выбор текстового файла

Ещё одной функцией вкладки кодирования в интерфейсе разрабатываемого программного средства является поле, предназначенное для ввода ключа, определяющего пиксели, в которые будет внедрена информация. Данное поле можно увидеть на представленном ранее на рисунке 7.

Последний элемент рассматриваемой вкладки - кнопка «Скрыть текст в видео». Нажатие по этой кнопке запускает процесс внедрения выбранного исходного текста в необходимый видеофайл.

Следующая рассматриваемая вкладка – вкладка «Извлечение». Здесь можно выбрать видеофайл, из которого необходимо извлечь скрытую информацию. Окно выбора аналогично окну из первой вкладки и выглядит также, как на рисунке 8. Тоже самое относится и к полю ввода ключа, только на этот раз ключ используется для определения пикселей, в которых скрыта информация. Последним элементом данной вкладки является кнопка, запускающая процесс изъятия скрытой информации из видеофайла. По окончанию процесса полученный текст выводится в самую нижнюю область интерфейса. Весь функционал вкладки «Извлечение» и результат изъятия скрытого текста представлен на рисунке 13.

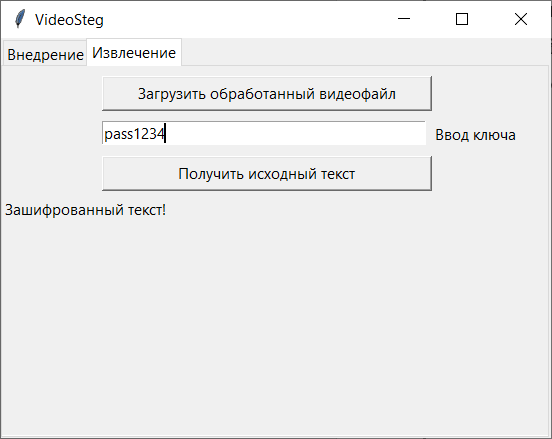


Рисунок 13 – Вкладка извлечения

На рисунке 14 можно увидеть результат ввода неверного ключа. Данный результат объясняется тем, что программное средство извлекло данные не из тех пикселей, в которых скрыт исходный текст.

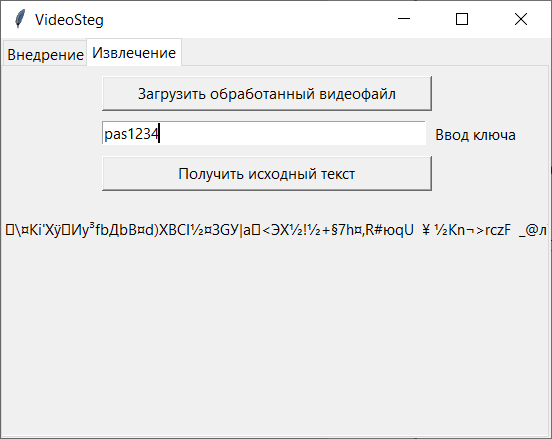


Рисунок 14 – Результат неправильного ввода ключа

Рассмотрим результат работы программного средства и оценим эффективность скрытия информации на примере одного кадра. Результат можно увидеть на рисунке 15.

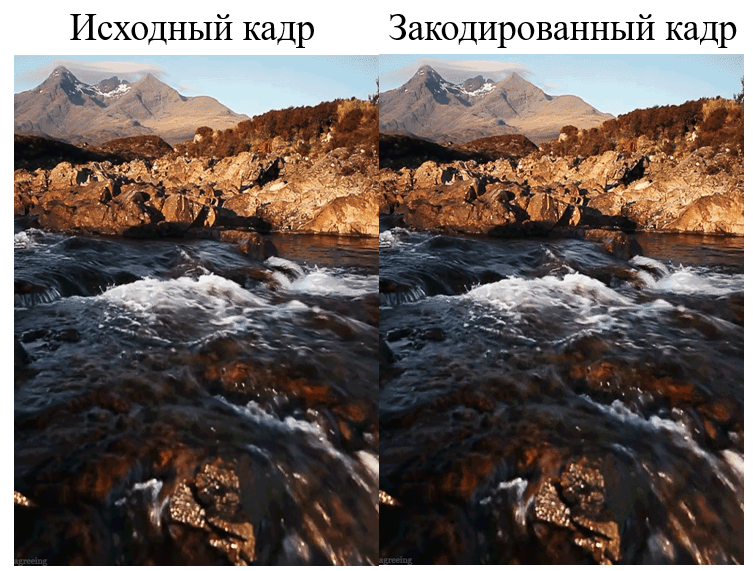


Рисунок 15 – Результат работы программы

Слева изображен исходный кадр, полученный из видеопотока, в котором будет скрыта информация. Справа – кадр видеофайла с внедрённым в него текстом, являющимся темой данного диплома. То есть в нём изменены значения многих пикселей и вызваны искажения. Но как можно наблюдать, различия в кадрах незаметны, а значит, что данный стеганографический метод работает.

**Вывод по главе**

В данной главе был дан перечень использованных средств разработки для написания программного средства, и их конкретное применение в программе. К ним относится среда разработки PyCharm, язык программирования Python, встроенная в язык библиотека Tkinter, с помощью которой был написан интерфейс программного средства, библиотека FFmpeg, использованная для преобразования видеофайла в изображения и наоборот, а также библиотека OpenCV, которая позволила редактировать полученные изображения и изменять значения пикселей. Описана структура разрабатываемой программы, её основные модули, а также продемонстрирована работа программного средства.

**Заключение**

В процессе исследования и разработки данной дипломной работы была поставлена и реализована цель разработать программное средство сокрытия информации в видеопотоке. Обоснована актуальность темы.

Были изучены теоретические основы стеганографии, её алгоритмы и методы. Также представлен анализ существующих программных средств – аналогов разрабатываемой программы, выявлены их достоинства и недостатки.

Рассмотрена алгоритмическая реализация программного средства, которая включает в себя алгоритмы сокрытия информации и её извлечение из видеофайлов. Подробно описан метод внедрения исходного текста.

Также был объяснён выбор программных средств разработки программы. Продемонстрирована работа функционала разрабатываемого программного средства.

Для достижения цели работы были выполнены задачи:

* изучены теоретические сведения стеганографии;
* проанализированы аналоги программного средства;
* разработать оптимальный комплекс модулей для обработки видеофайлов методом стеганографии;
* выполнена программная реализация.

**Перечень использованных информационных ресурсов**

1. Завьялов, С. В. Стеганографические методы защиты информации / С. В. Завьялов, Ю.В. Ветров – Санкт-Петербург : Издательство Политехнического университета, 2012. − 190 с. – Текст : непосредственный.
2. A Brief Timeline of Python. – Text : electronic // Python-history.blogspot: [website]. – 2009. – URL: http://python-history.blogspot.com/2009/01/brief-timeline-of-python.html (date of the application: 10.11.2021).
3. Краткий обзор языка Python. – Текст : электронный // Документация и книги по программированию : [сайт]. – 2016. – URL: https://www.helloworld.ru/texts/comp/lang/python/python2/index.htm (дата обращения 11.11.2021).
4. Стеганография в XXI веке. Цели. Практическое применение. Актуальность. – Текст: электронный // Habr : [сайт]. – 2015. – URL: https://habr.com/ru/post/253045/ (дата обращения 13.11.2021).
5. Грибунин, В. Г. Цифровая стеганография / В. Г. Грибунин, И. Н. Оков, И. В. Туринцев – Москва : Солон-Пресс, 2002. – 272 с. – Текст : непосредственный.
6. Библиотека Pillow. обработка изображений в Python. – Текст : электронный // Python 3 – изучение языка: [сайт]. – URL: https://python-scripts.com/pillow (дата обращения 15.11.2021).
7. Форсье Джефф, Django. Разработка веб–приложений на Python / А. Галунов, А. Киселёв – Санкт-Петербург : Символ-Плюс, 2009. – 456 c. – Текст : непосредственный.

**Приложение А**

**Техническое задание**

|  |  |  |  |
| --- | --- | --- | --- |
| СОГЛАСОВАНО | | | УТВЕРЖДЕНО |
|  | канд. физ-мат.  наук, доцент | | И.о. зав. каф. «КБИС» |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | О.В. Куликова | \_\_\_\_\_\_\_ к.т.н. Д.А.Короченцев |
| (подпись) | |  | (подпись) |
| «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г. | | | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г. |

## А.1 Введение

### А.1.1 Наименование программы

Наименование программы – «VideoSteg».

### А.1.2 Область применения

Данное программное средство применяется в областях, связанных со скрытой передачей информацией.

## А.2 Основания для разработки

Разработка ведётся на основании приказа об утверждении тем дипломных работ, утверждённого Донским государственным техническим университетом. Тема разработки: «Реализация программного средства обработки видеопотока стеганографическим методом».

## А.3 Назначение разработки

### А.3.1 Функциональное назначение

Функциональным назначением программного средства является сокрытие информации в видеофайлах, а также извлечение информации из них.

### А.3.2 Эксплуатационное назначение

Эксплуатационное назначение: программное средство предназначено для эксплуатации в организациях, в которых требуется возможность скрытой передачи информации.

## А.4 Требования к программе

### А.4.1 Требования к функциональным характеристикам

Программное средство должно обеспечить выполнение таких функций, как:

– выбор исходного текста и видеофайла для сокрытия информации;

– ввод ключа для кодирования или декодирования информации;

– встраивание исходного текста в видеопоток;

– получение обработанного видеофайла;

– извлечение скрытой информации из выбранного видеофайла;

– получение исходного текста.

### А.4.2 Требования к надёжности

Надёжность функционирования обеспечивается корректным функционированием программного и аппаратного обеспечения.

### А.4.3 Условия эксплуатации

Условия эксплуатации совпадают с условиями эксплуатации персональных ЭВМ IBM PC и совместимых с ними персональных компьютеров. Программа рассчитана на пользователя, владеющего базовыми навыками работы с десктопными приложениями. Дополнительных требований и ограничений не вводится.

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

Для работы программы требуется наличие IBM PC – совместимого персонального компьютера с тактовой частотой процессора не менее 1,5 Гц, объёмом свободной оперативной памяти не менее 2 ГБ, монитором и клавиатурой. Необходимое свободное место на жёстком диске – 150 Мб.

### А.4.5 Требования к информационной и программной совместимости

Программа должна работать под управлением операционной системы Windows 7 и выше. Базовый язык программирования – Python. Среда разработки – PyCharm.

### А.4.6 Требования к маркировке и упаковке

Требований к маркировке и упаковке не предъявлялось.

### А.4.7 Требования к транспортированию и хранению

Требований к транспортированию и хранению не предъявлялось.

## А.5 Требования к программной документации

Программная документация должна включать следующие документы:

* документ «Техническое задание» (ГОСТ 19.201-78).

## А.6 Стадии и этапы разработки

1. Системный анализ.
2. Общесистемное проектирование.
3. Подготовка технологических средств.
4. Программная реализация, рабочий проект.
5. Отладка программного средства в статике.
6. Тестовые испытания программного комплекса.

## А.7 Порядок контроля и приёмки

Контроль и приёмка разработки осуществляются на основе испытаний контрольно-отладочных примеров. Примеры должны демонстрировать правильность работы используемых в программе структур данных и алгоритмов в различных ситуациях, которые могут возникнуть при выполнении программы. При этом проверяются выполнение всех функций программы и полнота документации.

|  |  |
| --- | --- |
| **Разработчик:** | Коробов Вячеслав Сергеевич |
| **Дата начала разработки:** | « » 20\_\_ г. |

**Приложение Б Листинг**

Листинг Б.1 – main.py

from tkinter import \*

from tkinter import ttk

import glob

from tkinter import messagebox

from tkinter import filedialog

import cv2

import random

from random import randint

import ffmpeg

import os

def enc():

files = glob.glob('frames/\*')

for f in files:

os.remove(f)

try:

(ffmpeg.input(origin\_video\_file)

.output('frames/%04d.png',

start\_number=0)

.run(capture\_stdout=True, capture\_stderr=True))

except ffmpeg.Error as e:

print('stdout:', e.stdout.decode('utf8'))

print('stderr:', e.stderr.decode('utf8'))

DIR = "frames/"

print(len([name for name in os.listdir(DIR) if os.path.isfile(os.path.join(DIR, name))]))

frames = len([name for name in os.listdir(DIR) if os.path.isfile(os.path.join(DIR, name))])

img0 = cv2.imread('frames/0000.png', cv2.IMREAD\_COLOR)

height, width = img0.shape[:2]

#key = input("Введите ключ\n")

key1 = key.get()

random.seed(key1)

#opentext = open('text.txt', encoding='UTF-8')

opentext = open(origin\_text\_file, encoding='UTF-8')

text = opentext.read()

dlina = len(text)

a = dlina \* 3 + 2

mass = [-1] \* a

p = 0

'Заполнение пикселей символами текста'

while (p < dlina + 1):

pixel = False

k = randint(0, frames - 1) # Номер кадра

img = cv2.imread('frames/%04d.png' % k, cv2.IMREAD\_COLOR) # Открытие кадра

h = randint(0, height - 1) # Поиск свободного пикселя

w = randint(0, width - 1)

for j in range(0, p \* 3, 3):

if (mass[j] == k) & (mass[j + 1] == h) & (

mass[j + 2] == w): # Проверка, заполнен ли пиксель

print("da")

pixel = True

break

if pixel:

# print(h, w)

continue

(b, g, r) = img[h, w]

'Перевод пикселей в двоичные числа'

bb = format(b, 'b') # Синий цвет

for i in range(8 - len(bb)):

bb = ('0' + bb)

gg = format(g, 'b') # Зеленый цвет

for i in range(8 - len(gg)):

gg = ('0' + gg)

rr = format(r, 'b') # Красный цвет

for i in range(8 - len(rr)):

rr = ('0' + rr)

'Обработка символа'

if (p == dlina):

s = format(0, 'b') # Последний "нулевой" символ

for i in range(8 - len(s)):

s = ('0' + s) # Расширение до 8 бит

else:

s = format(ord(text[p].encode('cp1251')), 'b') # Получение числа символа

for i in range(8 - len(s)):

s = ('0' + s) # Расширение до 8 бит

'Заполнение пикселя символом'

rr = rr[:5] + s[:3] # Красный цвет

gg = gg[:6] + s[3:5] # Зеленый цвет

bb = bb[:5] + s[5:] # Синий цвет

img[h, w] = (int(bb, 2), int(gg, 2), int(rr, 2)) # Перевод цветов в целые числа и сохранение на изображении

mass[p \* 3] = k

mass[p \* 2 + 3] = h # Сохранение заполненного пикселя

mass[p \* 2 + 3] = w #

print("Пиксель №", p, " - номер кадра: ", k, " - высота: ", h, "; широта: ", w) ###

cv2.imwrite('frames/%04d.png' % k, img)

p = p + 1

# Сборка видео

(

ffmpeg

.input('frames/%04d.png', framerate=24)

.output('STEG\_VIDEO.mp4', codec='copy')

.overwrite\_output()

.run()

)

messagebox.showinfo('Успешно!', "Закодированный видеофайл сохранён")

def dec():

try:

(ffmpeg.input(enc\_video\_file)

.output('frames2/%04d.png',

start\_number=0)

.run(capture\_stdout=True, capture\_stderr=True))

except ffmpeg.Error as e:

print('stdout:', e.stdout.decode('utf8'))

print('stderr:', e.stderr.decode('utf8'))

DIR = "frames2/"

print(len([name for name in os.listdir(DIR) if os.path.isfile(os.path.join(DIR, name))]))

frames = len([name for name in os.listdir(DIR) if os.path.isfile(os.path.join(DIR, name))])

img = cv2.imread('frames2/0000.png', cv2.IMREAD\_COLOR)

height, width = img.shape[:2]

#key = input("Введите ключ\n")

key2 = txt.get()

random.seed(key2)

# dlina = 20

# mass = [-1]\*dlina\*2

mass = []

p = 0

text = []

s2 = 1

while (chr(s2) != chr(0)):

pixel = False

k = randint(0, frames - 1)

img = cv2.imread('frames2/%04d.png' % k, cv2.IMREAD\_COLOR)

h = randint(0, height - 1)

w = randint(0, width - 1)

for j in range(0, p \* 3, 3):

if (mass[j] == h) & (mass[j + 1] == w): # Проверка, пройден ли пиксель

print("da")

pixel = True

break

if pixel:

# print(h, w)

continue

(b, g, r) = img[h, w]

'Перевод пикселей в двоичные числа'

bb = format(b, 'b') # Синий цвет

for i in range(8 - len(bb)):

bb = ('0' + bb)

gg = format(g, 'b') # Зеленый цвет

for i in range(8 - len(gg)):

gg = ('0' + gg)

rr = format(r, 'b') # Красный цвет

for i in range(8 - len(rr)):

rr = ('0' + rr)

s = rr[5:] + gg[6:] + bb[5:]

s2 = (int(s, 2)) # Извлеченный из пикселя символ

if (s2 > 192) & (s2 < 255):

# print(chr(s2 + 848))

text.append(chr(s2 + 848))

else:

# print(chr(s2))

text.append(chr(s2))

# mass[p\*2] = h # Сохранение заполненного пикселя

# mass[p\*2+1] = w

mass.append(k)

mass.append(h)

mass.append(w)

# print("Пиксель №", p, " - высота: ", h, "; широта: ", w)

p = p + 1

# for i in range(len(text)):

# print(text[i])

str\_a = ''.join(text)

print(str\_a)

gottext = open('resulttext.txt', 'w', encoding="utf-8")

gottext.write(str\_a)

vivod.configure(text=str\_a)

def opentext():

global origin\_text\_file

origin\_text\_file = filedialog.askopenfilename(filetypes = (("Text files","\*.txt"),("all files","\*.\*")))

# opentext = open(origin\_text\_file, encoding='UTF-8')

# global text

def openvideo():

global origin\_video\_file

origin\_video\_file = filedialog.askopenfilename(filetypes=(("Videofiles", "\*.mp4"), ("all files", "\*.\*")))

# global img

# img = origin\_video\_file

def openvideo2():

files = glob.glob('frames2/\*')

for f in files:

os.remove(f)

global enc\_video\_file

enc\_video\_file = filedialog.askopenfilename(filetypes=(("Videofiles", "\*.mp4"), ("all files", "\*.\*")))

# global encimg

# encimg = enc\_video\_file

# Интерфейс программы

window = Tk()

window.title("VideoSteg")

window.geometry('550x400')

# Создание вкладок

tab\_control = ttk.Notebook(window)

tab1 = ttk.Frame(tab\_control)

tab2 = ttk.Frame(tab\_control)

tab\_control.add(tab1, text='Кодирование')

tab\_control.add(tab2, text='Декодирование')

# Первая вкладка-----------------------------------------------------------

btn1 = Button(tab1, width=40, height=1, text="Выбрать видеофайл", command = openvideo)

# btn1.grid(column=0, row=0)

btn1.place(x=100, y=10)

btn2 = Button(tab1, width=40, height=1, text="Выбрать исходный текст", command = opentext)

# btn2.grid(column=0, row=1)

btn2.place(x=100, y = 50)

key = Entry(tab1, width=40)

# key1 = key.get()

# txt.grid(column=0, row=3)

key.place(x = 100, y = 95)

lbl1 = Label(tab1, text="Ввод ключа",)

# lbl1.grid(column=1, row=3)

lbl1.place(x = 430, y = 95)

btn3 = Button(tab1, width=40, height=1, text="Закодировать текст", command = enc)

# btn.grid(column=0, row=4)

btn3.place(x = 100, y = 130)

# adigabze = Label(tab1, text="")

# adigabze.grid(column=0, row=5)

# Вторая вкладка-----------------------------------------------------------------

btn = Button(tab2, width=40, height=1, text="Выбрать закодированный видеофайл", command = openvideo2)

btn.place(x = 100, y = 10)

# btn.grid(column=0, row=0)

txt = Entry(tab2, width=40)

txt.place(x = 100, y = 55)

# key2 = txt.get()

# txt.grid(column=0, row=1)

lbl2 = Label(tab2, text="Ввод ключа")

lbl2.place(x = 430, y = 55)

# lbl2.grid(column=1, row=1)

btn = Button(tab2, width=40, height=1, text="Получить исходный текст", command = dec)

btn.place(x = 100, y = 90)

# btn.grid(column=0, row=2)

vivod = Label(tab2, text="")

vivod.place(x = 0, y = 130)

# vivod.grid(column=0, row=3)

tab\_control.pack(expand=1, fill='both')

window.mainloop()