МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ

ГОСУДАРСТВЕННЫЙТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 05 01 Информационные системы и технологии

Направление специальности 1-40 01 02 03 Информационные системы

и технологии (издательско-полиграфический комплекс)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Защита информации и надежность информационных систем»

Тема: «Реализация стеганографического метода на основе изменения межстрочного интервала в pdf»

Исполнитель

Студент 3 курса группы 2 Злобин Р.Ю.

(Ф.И.О.)

Руководитель работы ассистент Берников В.О. (учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

(подпись, дата)

Минск 2021

**Содержание**

[Введение 3](#_Toc72600647)

[1 Аналитический обзор литературы 4](#_Toc72600648)

[2 Обзор аналогов 8](#_Toc72600649)

[3 Описание метода изменения межсимвольных интервалов 11](#_Toc72600650)

[4 Проектирование и разработка приложения в среде Visual Studio 13](#_Toc72600651)

[5 Тестирование программного средства 21](#_Toc72600652)

[6 Руководство пользователя 24](#_Toc72600653)

[Заключение 27](#_Toc72600654)

[Список используемых источников 28](#_Toc72600655)

[Приложение А 29](#_Toc72600656)

[Приложение Б 31](#_Toc72600657)

# Введение

Современные технологии позволяют быстро и эффективно распространять информацию. А интенсивное внедрение сетевых компьютерных технологий во многие сферы жизни общества обусловило появление проблемы защиты информации в компьютерных сетях.

Для того, чтобы защитить свою работу, охраняемую авторским правом, от незаконного копирования и нелегального распространения, люди стали применять методы стеганографии и криптографии. Методы стеганографии направлены на сокрытие факта передачи или хранения информации, а методы криптографии отвечают за шифрование самой информации. Но важно понимать, что в отличии от криптографии, где неприятель точно может определить является ли передаваемое сообщение зашифрованным текстом, методы стеганографии позволяют встраивать секретные сообщения в документ так, чтобы невозможно было заподозрить существование встроенного тайного послания.

Зачастую стеганографические и криптографические методы применяются вместе, что создает мощную систему по защите информации.

Развитие средств вычислительной техники дало новый толчок для развития компьютерной стеганографии. Появилось много новых областей применения. Сообщения встраивают в цифровые данные, имеющие аналоговую природу: речь, аудиозаписи, изображения, видео. Так же стали встраивать информацию в текстовые файлы и исполняемые файлы программ.

В настоящее время можно выделить две причины популярности исследований в области стеганографии: ограничение на использование криптосредств в ряде стран мира и появление проблемы защиты прав собственности на информацию, представленную в цифровом виде. Первая причина повлекла за собой большое количество исследований классической стеганографии (т.е. скрытия факта передачи информации), вторая – многочисленные работы в области так называемых водяных знаков. Цифровой водяной знак– специальная метка, незаметно внедряемая в изображение или другой сигнал, чтобы контролировать его использование.

Стеганография является эффективным программно-техническим методом сокрытия данных и защиты их от несанкционированного доступа. Эффективное использование стеганографии совместно с другими методами защиты информации обеспечит многоуровневую безопасность.

Целью данного курсового проекта является разработка программного средства реализующего стеганографический метод на основе изменения межсимвольного интервала в pdf.

Исходя из этого сформулированы следующие задачи:

* реализация анализа документа на основе работ служебных программ;
* изучение метода изменения межсимвольного интервала для документов в формате pdf.
* реализация стеганографического метода в pdf документе.

# Аналитический обзор литературы

Как мы уже выяснили, стеганография — это метод, позволяющий пользователям скрыть сообщение в другом сообщении.

Стеганографию можно разделить на три раздела [1]:

* классическая стеганография – включает в себя все «некомпьютерные методы»: симпатические чернила, микрофотоснимки и другие методы, придуманные людьми задолго до вхождения человечества в компьютерную эпоху;
* компьютерная стеганография – направление классической стеганографии, основанное на особенностях компьютерной платформы и использования специальных свойств компьютерных форматов данных;
* цифровая стеганография – направление классической стеганографии, основанное на сокрытии или внедрении дополнительной информации в цифровые объекты, вызывая при этом некоторые искажения этих объектов. Используется избыточность аудио- и визуальной информации.

Стеганографические системы активно используются для решения большого круга задач:

* для защиты конфиденциальной информации от возможного саботажа, кражи или несанкционированного просмотра;
* для преодоления систем мониторинга и управления сетевыми ресурсами;
* при камуфлировании программного обеспечения – в случаях, когда использование программного обеспечения незарегистрированными пользователями является нежелательным, оно может быть замаскировано под стандартные программные продукты;
* для защиты авторского права на некоторые виды интеллектуальной собственности – на компьютерные графические изображения наносится специальная метка, которая остается невидимой для глаз, но распознается специальным программным обеспечением.

Кроме скрытой передачи сообщений, стеганография является одной из самых перспективных направлений, применяемых для аутентификации и маркировки авторской продукции. При этом, часто в качестве внедряемой информации используются дата и место создания продукта, данные об авторе, номер лицензии, серийный номер, дата истечения срока работы и др. Эта информация обычно внедряется как в графические и аудио произведения, так и в защищаемые программные продукты. Все внесенные сведения могут рассматриваться как веские доказательства при рассмотрении вопросов и судебных разбирательств об авторстве или для доказательства факта нелегального копирования, и часто имеют решающее значение

Стегосистема – это совокупность средств и методов, которые используются для формирования скрытого канала передачи информации. При построении стегосистемы должны учитываться следующие положения:

* методы скрытия должны обеспечивать аутентичность и целостность информации, в которой скрывается сообщение;
* противник имеет полное представление о стегосистеме и деталях её реализации. Единственной информацией, неизвестной потенциальному противнику, является ключ, с помощью которого только его держатель может установить факт присутствия и содержания скрытого сообщения;
* если противник каким-то образом узнает о факте существования скрытого сообщения, то это не должно позволить ему извлечь скрытую информацию до тех пор, пока ключ хранится в тайне;
* потенциальный противник должен быть лишён каких-либо технических и иных преимуществ в распознавании или раскрытии содержания скрытых сообщений.

Рассмотрим структурную схему стегосистемы (рисунок 1.1):

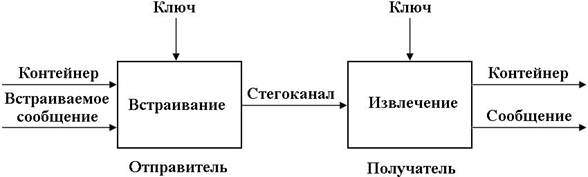


Рисунок 1.1 – Структурная схема стегосистемы

Контейнер – это любая информация, предназначенная для скрытия сообщения. Вид контейнера оказывает существенное влияние на надёжность стегосистемы и возможность обнаружения факта передачи скрытого сообщения. По размеру контейнеры бывают: непрерывные (потоковые) и ограниченной (фиксированной) длины.

Особенность потокового контейнера - невозможно определить его начало и конец. В таком контейнере биты информации, используемые для скрытия сообщения, включаются в общий поток в реальном масштабе времени и выбираются с помощью специального генератора, задающего расстояния между ними. 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. Трудность отправителя сообщения – возможны проблемы с размещения всего сообщения, из-за длины потока контейнера.

При использовании контейнера ограниченной длины отправитель заранее знает размер файла и может выбрать скрывающие биты в подходящей псевдослучайной последовательности. Но из-за ограниченного объёма, встраиваемое сообщение иногда может не поместиться в файл-контейнер. На практике чаще всего используются контейнеры ограниченной длины как наиболее распространённые и доступные.

В зависимости от вида информации, используемой для встраивания сообщений, контейнеры могут быть:

* визуальные (представляет собой картинку или фотографию, в которой для встраивания сообщений используются младшие биты, приводящие к небольшим изменениям яркости определённых точек растра изображения);
* звуковые (представляет собой речевой или музыкальный сигнал, в котором для встраивания сообщений используются младшие биты аудиосигнала, что практически не отражается на качестве звука);
* текстовые (представляет собой текстовый файл, подготовленный к печати на принтере, в котором для встраивания сообщений используются небольшие изменения стандартов печати (расстояния между буквами, словами и строками, размеры букв, строк и др.)).

Встроенное (скрытое) сообщение – это сообщение, встроенное в контейнер.

Стеганографический канал (стегоканал) – это канал передачи скрытого сообщения.

Сообщение – это любая информация, подлежащая скрытой передаче. В качестве сообщения может использоваться любой вид информации: текст, изображение, аудиосигнал.

Стеганографическим ключ – место и порядок скрытия сообщения в открытых данных. Математическая модель стеганографической системы может быть представлена в следующем виде [2]:

(1.1)

(1.2)

где S – множество контейнеров с осажденной информацией,

Emb() и Ext() – функции встраивания скрытого сообщения в файл-контейнер и извлечения из файла-контейнера соответственно.

C – множество всех контейнеров,

V – объем контейнера,

K – множество стеганографических ключей,

M – множество скрытых сообщений.

В зависимости от количества уровней защиты в стегосистеме может быть один или несколько стегоключей.

По типу стегоключа стегосистемы бывают: с секретным ключом и с открытым ключом.

В стегосистеме с секретным ключом для встраивания и извлечения сообщения используется один ключ, который должен быть определён либо до начала обмена секретными сообщениями, либо передан по защищённому каналу.

В стегосистеме с открытым ключом для встраивания и извлечения сообщения используются разные ключи, различие которых состоит в том, что с помощью вычислений невозможно определить один ключ из другого. Поэтому один ключ (открытый) может передаваться свободно по незащищённому каналу связи.

Любая стегосистема должна отвечать следующим требованиям:

* свойства контейнера должны быть модифицированы таким образом, чтобы изменение невозможно было выявить визуально;
* стеганосообщение должно быть устойчиво к искажениям, которые могут иметь место при его передаче, включая и различные трансформации (уменьшение, увеличение, преобразование в другой формат, сжатие без потери информации, сжатие с потерей информации и т. д.);
* для сохранения целостности встраиваемого сообщения необходимо использовать коды с исправлением ошибок;
* для повышения надёжности встраиваемое сообщение должно быть продублировано.

В связи с широким распространением, электронные документы Microsoft Office часто используются в качестве файлов-контейнеров. Для них применяются методы, которые используют наравне с классическими методами текстовой стеганографии методы, свойственные контейнеру, такие как формат и смещение текста, наличие истории редактирования и т.д., что позволяет добиться увеличения скрытности и пропускной способности. [4]

Но стоит рассмотреть несколько видов текстовых контейнеров, таких как: \*.txt, \*.html, \*.xml, \*.doc, \*.docx, \*.pdf.

– TXT – В файле имеется исключительно текст, форматирование, картинки и шрифты в txt не сохраняются. Данный формат можно считать одним из наиболее универсальных – его распознают сотни приложений для различных устройств.

– HTML – создавался специально для разработки веб-страниц, однако его удобство, небольшой размер и универсальность привели к тому, что сейчас данное решение активно используется для хранения текстовой информации. Файл HTML легко воспроизвести базовыми инструментами ОС, он не требует установки стороннего программного обеспечения.

– XML – это язык описания документов, во многом похожий на язык разметки гипертекста HTML, но гораздо более универсальный. Он используется для конструирования Web-страниц.

– DOC – относится к бинарным, их поддерживают практически все существующие сегодня текстовые редакторы. Данный формат поддерживает гиперссылки, вставку в текст картинок и прочих объектов, стили и форматирование.

– DOCX – это усовершенствованный вариант doc. Он использует открытый XML форматирование и использует ZIP- файлы сжатия для уменьшения размера документов.

– PDF – содержит изображение страницы, включающее текст и графику. Файлы в формате PDF могут быть прочитаны при помощи различных программ, предназначенных только для чтения файлов, однако создаются только при помощи программы Adobe Acrobat.

Формат pdf для нас предпочтительнее всего, т.к. данный формат мишно и платформонезависим. Это значит, что любой документ данного формата может быть открыт на компьютере, работающем под управлением любой операционной системы (с помощью соответствующего программного обеспечения), и при этом будет выглядеть точно так же, как и на том ПК, на котором он создавался. Это – главное преимущество формата PDF.

# Обзор аналогов

Методы компьютерной стеганографии делятся на:

* методы, основанные на избыточности визуальной и аудиоинформации;
* методы, основанные на использовании специальных свойств компьютерных форматов.

Методы, основанные на избыточности визуальной и аудиоинформации, для скрытия информации используют младшие разряды цифровых отсчётов цифрового изображения и звука, которые содержат очень мало полезной информации. Их заполнение дополнительной информацией практически не влияет на качество восприятия, что и даёт возможность скрытия конфиденциальной информации.

Преимущество – передача большого объёма скрытой информации и возможность защиты авторского права путём создания скрытого изображения товарной марки, регистрационного номера и т. п.

Недостаток – за счёт введения дополнительной информации искажаются статистические характеристики цифровых потоков. Для снижения компрометирующих признаков требуется коррекция статистических характеристик.

Методы, основанные на использовании специальных свойств компьютерных форматов, делятся на:

* методы использования зарезервированных для расширения полей компьютерных форматов данных;
* методы специального форматирования текстовых файлов;
* методы скрытия в неиспользуемых местах компакт дисков;
* методы использования имитирующих функций;
* методы удаления идентифицирующего файл заголовка.

Нас интересуют методы специального форматирования текстовых файлов, поэтому рассмотрим данные методы подробнее.

В текстовых файлах, секретная информация чаще всего кодируется путем изменения количества пробелов, использования невидимых символов, регистра букв, путем изменения межстрочных интервалов, табуляций и т.п.

При использовании методов межстрочных интервалов и табуляций объем информации, встраиваемой в файл, в несколько раз меньше, чем при использовании метода изменения количества пробелов [3].

Очень подробно рассматривал проблему встраивания скрытой информации в текстовые файлы Брассил. К разработанным им методам относят:

* Line-Shift Coding (рисунок 2.1) – изменение расстояния между строками электронного текста. Он также называется методом изменения межстрочных интервалов. Его сущность заключается в том, что используется текст с различными межстрочными расстояниями. Выделяется максимальное и минимальное расстояния между строками, позволяющее кодировать соответственно символы «1» и «0» осаждаемого сообщения;

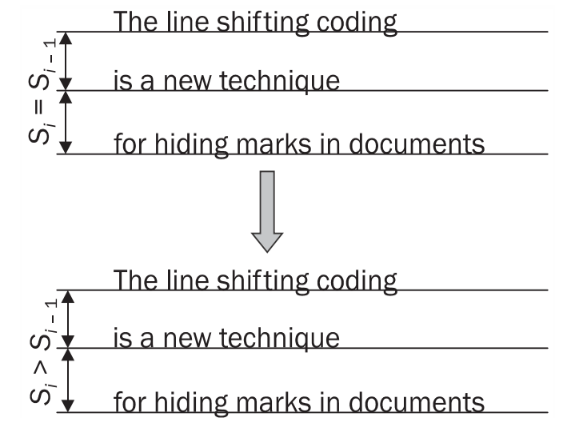


Рисунок 2.1 – Изменение расстояния между строками электронного текста.

* Word-Shift Coding (рисунок 2.2) – изменение расстояния между словами строки электронного текста. Суть метода состоит в том, что борется текст с разными расстояниями между словами. Выделяются максимальное и минимальное расстояния, которые обозначаются соответственно 1 и 0, а остальные расстояния увеличивают или уменьшают до размеров выделенных;

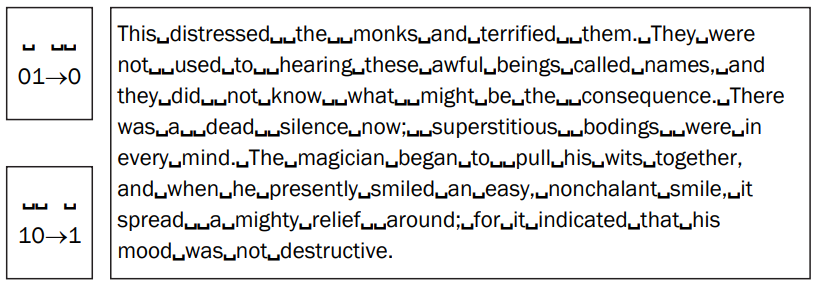


Рисунок 2.2 – Изменение расстояния между словами строки электронного текста на основе длин пробелов.

* Feature Coding (рисунок 2.3) – внесение специфических изменений в шрифты отдельных букв. Этот метод заключается в изменении написания отдельных букв используемого стандартного шрифта. Визуально заметны различные образы, соответствующие буквам с верхними или нижними выносными элементами. При этом можно закодировать стегосообщение так, что модифицированная буква будет означать «1», а не модифицированная – «0». Модифицировать можно несколько букв. Таким образом, объем встраиваемого сообщения будет увеличиваться.

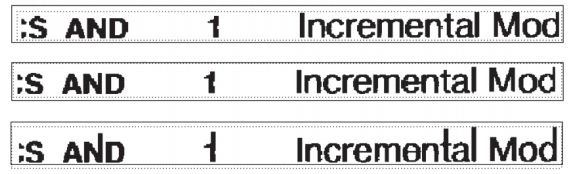


Рисунок 2.3 – Изменение в шрифты отдельных букв

Эти методы легко, встраиваются в любой текст, независимо от его содержания, назначения и языка. Такие системы легко разрабатывать, и выполняются они автоматически.

Недостатком является то, что этими методами нельзя передавать большое количество скрытой информации.

В настоящее время манипулирование пробелами кажется полезным и имеет свой потенциал в скрытии информации, так как в текстовых документах пробелов достаточно много. Это даже преимущество, когда никто не узнает, что пустой кусок документа на самом деле является жизненно важной секретной информацией.

Можем заметить, что методов текстовой стеганографии не мало и каждый имеет свои плюсы и минусы. Следует отметить, что при стеганографическом преобразовании данные не шифруются, однако, часто предполагается, что скрытое сообщение может быть предварительно зашифровано криптографическими методами для дополнительной защиты данных.

Рассмотрим приложения, которые работают с методами специального форматирования текстовых файлов:

1. Приложение AH4S (рисунок 2.4) – использует структуру сети omega для сокрытия и извлечения секретных сообщений. Генерация секретного сообщения выполняется следующим образом: берем каждую букву из исходного сообщения, используем сеть омега для генерации двух связанных букв из выбранной буквы и, наконец, ищем в словаре подходящее английское заглавное слово, чтобы скрыть генерируемые две буквы. Чтобы увеличить шанс найти подходящие слова, генерируемые две буквы, не обязательно должны находиться рядом с главным словом. В результате используется стеганография с белым пробелом, чтобы скрыть положение двух букв в выбранном заглавном слове.

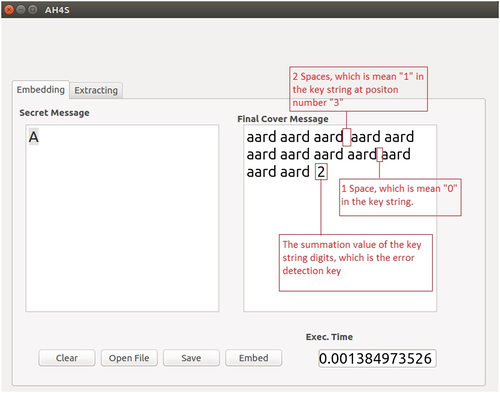


Рисунок 2.4 – Приложение AH4S

На основе рассмотренных выше аналогов, можно сделать вывод, что стеганография с текстовыми контейнерами чаще всего использует методы изменения неотображаемых символов (пробелов, табуляций, знаков переноса строки, абзацев и т.п.) для сокрытия внесенных изменений. Так как такие изменения трудно рассмотреть невооруженным глазом и при изменении в исходном документе размера текста, шрифта и других отображаемых параметров, они с большей вероятностью останутся незаметны.

Исходя из выше сказанного была поставлена задача, а именно разработать программу для сокрытия текстовой информации методом изменения межсимвольных интервалов в текстовом документе формата PDF.

# Описание метода изменения межсимвольных интервалов

Большинство современных стеганографических методов используют мультимедийные файлы, например, фотографии, аудио и видеофайлы для скрытия информации. В основном из-за возможностей стеганографического внедрения, которые они предоставляют. Самое важное свойство стеганографического метода – это вместимость с обеспечением безопасности.

Несмотря на популярность мультимедийных файлов для стеганографических целей, другие файлы, будь то файлы двоичных данных, исполняемые или текстовые файлы, могут также использоваться для скрытия информации. Широкое использование PDF-файлов и его использование для этой цели представляет собой интересное и практическое решение. Хотя это сложнее сделать поскольку, как правило, меньше свободного места для скрытой информации. На основе текста формат документа PDF также может быть ограниченным, поскольку легко анализировать его содержимое, но может быть сложнее скрыть данные в нем. Было сделано несколько попыток в области стеганографии PDF, но представленные решения и реализация не всегда очень доступно описаны и/или опубликованы. Поэтому трудно предположить, как работает предлагаемый метод. Необходимо больше исследований в области PDF– стеганографии для проверки или опровержения предложенного метода.

Обзор существующих методов для предоставления четкого понимания о ландшафте стеганографии PDF приведен ниже. Некоторые из существующих методов фокусируются только на сокрытии данных с помощью невидимых PDF. В результате данные будут совершенно не обнаружимыми, если PDF открывается в обычном средстве просмотра PDF.

Есть два способа скрыть сообщение используя метод межстрочных интервалов. Первый метод использует специальные неотображаемые символы – неразрывный пробел, табуляция, знак переноса строки, абзацев, пробел и так далее. Изменение начертания и высоты шрифта этих символов не влияет на отображение электронного текста. Можно предложить некоторые модификации данного метода.

К примеру, изменения могут проводится по заранее заготовленной маске, например, только по нечётным абзацам. Недостатком этого метода является величина пропускной способности, однако для небольших по объёму скрытых меток это безразлично.

Второй метод межстрочных интервалов изменяет документ путем изменения расстояния между строками в документе. В документ добавляется скрытая метка и особым образом форматируется, а именно изменяется межстрочных интервал таким образом, чтобы тест ушёл за зону отображения. Недостатком данного метода является, увеличение размера результирующего файла, однако он позволяет существенно увеличить количество встраиваемой информации.

Т.к. документ PDF не позволяет извлечь межстрочное расстояние отдельного символа в тексте стандартным способом, то в данном проекте бы будем использовать второй метод, который также изменяет расстояние между строками, но который позволяет отследить скрытую метку считывая документ особым образом.

Одной из особенностей PDF документов является то, что место положение текста на странице определяет порядок считывания текста. Тем выше и левее находится начало строки электронного текса тем раньше будет считана данная строка.

Таким кодировщик должен добавлять символы метки в файл и форматировать таким образом, чтобы они были выше или ниже зоны отображения. Для удобства определения метки кодировщик обрамит метку особым символом, который будет служить маячком начала и конца текста метки. Декодер производит считывание текста со страницы документа и ищет текст, обрамлённый специальным символом.

Следует отметить, что осаждать метку имеет смысл только на первой странице документа, это существенно сократит время извлечения метки из многостаночных документов.

Преимуществом данного метода является его высокая эффективность: можно зашифровать большое количество информации в относительно небольшом контейнере.

Недостатком данного метода является изменение размера файла. Однако данная проблема решаема встраиванием небольших сообщений. Следовательно, данный метод будет полезен для передачи секретной информации, не требующей хранения большого количества символов.

Данный метод можно модифицировать, применяя перестановочные алгоритмы шифрования для более безопасной передачи данных.

# 4 Проектирование и разработка приложения в среде Visual Studio

Форма реализации продукта – десктопное приложение.

Десктоп приложение реализовано с использованием языка С#. Технология для реализации – WPF.

Само приложение должно:

* открывать окно для выбора файла контейнера;
* встраивать введенное пользователем сообщение в выбранный текстовый контейнер;
* извлекать сообщение пользователя из текстового контейнера.

Рассмотрим решение проекта «PDFStegano», которое представлено двумя проектами, «PDFSteganoLib» и «PDFSteganoWPF», имеющие структуру, представленную на рисунке (Рисунок 4.1) ниже:

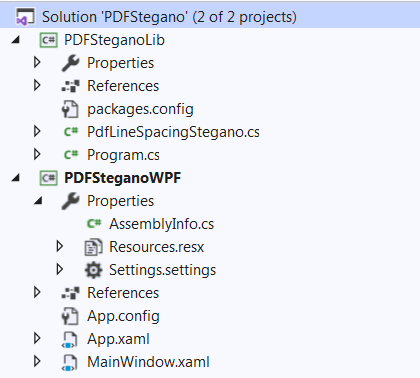


Рисунок 4.1 – Структура проекта

Разделение решения на два проекта, библиотеку и оконное приложение, отделяет проектирование и тестирование стеганографического метода от процесса создания оконного приложения, что позволяет вести параллельную разработку обоих проектов.

Для работы с форматом pdf необходимо было выбрать одну или несколько библиотек.

В ходе поиска были найдены библиотеки для работы над разными задачами с форматом pdf:

* iTextSharp – позволяет создавать и манипулировать PDF документами. Она позволяет разработчикам совершенствовать веб и прикладные приложения с помощью динамической генерации и/или манипуляции над PDF документами;
* [Report.NET](http://report.sourceforge.net/) – это мощная библиотека, которая поможет Вам сгенерировать PDF документы простым и гибким способом. Документ может быть создан с помощью данных, которые были получены из любого ADO.NET data set;
* PDFsharp – это библиотека на C#, которая очень просто создаёт PDF документы на лету. Похоже на GDI+ рисование процедур, аналогично можно создавать PDF документы, рисовать на экране или отправлять на печать любого принтера. PDFsharp может изменять, объединять и разделять существующие PDF файлы или перемещать страницы из существующих PDF файлов в новые PDF документы;
* SharpPDF – это C# библиотека, которая может создавать различные объекты в PDF документах за несколько шагов. Она создана для .NET framework 1.1 и может создавать 100% совместимые PDF файлы (протестировано с Acrobat Reader, Ghostscript, JAWS PDF Editor и другими PDF-readers). Самая важная цель библиотеки – это простое использование;
* PDFjet – библиотека с открытым исходным кодом для динамического создания PDF документов из Java и .NET;
* FO PDF -- похожа на ASP.NET Server Controls, написана на C#. Она получает DataTable и несколько других параметров для генерации XSL FO и рендерит DataGrid как PDF отчёт используя NFOP (Apache FOP Port in J#) PDF Formatter.

В ходе анализа был сделан выбор в пользу библиотеки ITextSharp, т.к. при ее использовании у разработчика появляются такие возможности как:

* передачи PDF в браузер;
* генерировать динамические документы из XML файлов или базы данных;
* использовать много интерактивных возможностей PDF;
* добавлять закладки, число страниц, водяные знаки и т.д.
* разделять, объединять и манипулировать PDF страницами;
* автоматизация заполнения PDF форм;
* добавлять цифровую подпись в PDF файл.

Основная логика стеганографического метода реализована в классе «PdfLineSpacingStegano» проекта «PdfSteganoLib». Структура класса и его методы приведены на рисунке 4.1.

Класс «PdfLineSpacingStegano» отвечает за операции проекта и имеет реализации следующих методов, которые приведены на рисунке 4.1.

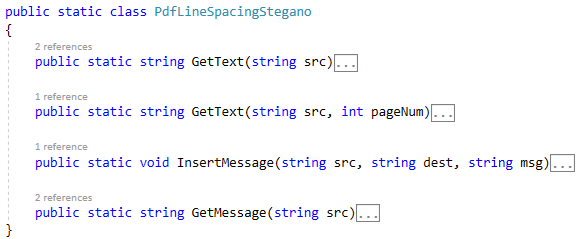


Рисунок 4.1 – Структура класса «PdfLineSpacingStegano»

Методы «GetText» отвечают за извлечение информации из всего файла и из его первой страницы, структура показана в листинге 4.1.

|  |
| --- |
| public static string GetText(string src, int pageNum)  {  if (File.Exists(src))  {  using (var pdfDoc = new PdfDocument(new PdfReader(src)))  {  if (pageNum <= 0 || pageNum > pdfDoc.GetNumberOfPages())  return "";  var strategy = new LocationTextExtractionStrategy();  var page = pdfDoc.GetPage(pageNum);  return PdfTextExtractor.GetTextFromPage(page, strategy); }  }  return "";  } |

Листинг. 4.1 – Структура метода «readPDF»

Метод GetText имеет внутри себя несколько важных проверок, а именно, проверка на наличие файла и на наличие в документе необходимой страницы. При не прохождении проверок метод возвращает пустую строку. Извлечение текста со страницы происходит с помощью метода GetTextFromPage класса PdfTextExtractor.

Метод «InsertMessage» отвечает за осаждение секретной метки в контейнер, структура показана в листинге 4.2.

|  |
| --- |
| public static void InsertMessage(string src, string dest, string msg)  {  if (File.Exists(src))  {  int i = GetText(src, 1).Count(x=> x == '\u00A0');  using (PdfReader pdfReader = new PdfReader(src))  using (PdfWriter pdfWriter = new PdfWriter(dest))  {  using (PdfDocument pdfDoc = new PdfDocument(pdfReader, pdfWriter))  {  using (Document doc = new Document(pdfDoc))  {  var par = new Paragraph();  par.SetFixedLeading(-1000f - 111f \* (i + 2));  par.SetCharacterSpacing(-4f);  par.SetWordSpacing(-4f);  Text text = new Text(string.Format("{0}{1}{0}", '\u00A0', msg));  text.SetFontColor(ColorConstants.WHITE, 0);  par.Add(text);  doc.Add(par);  }  }  }  }  } |

Листинг. 4.2 – Структура метода «InsertMessage»

Метод InsertMessage принимает три параметра: путь к контейнеру, путь выходного файла и секретная метка. Данный метод открывает контейнер для чтения и выходной файл для записи. Затем в открытый документ осаждается секретная метка. Происходит это следующим образом, создаётся новый параграф, в который помещается, специальным образом отформатированная секретная метка. Данный параграф страивается в документ, из происходит закрытие документа.

Метод «GetMessage» отвечает за извлечение секретной метки из контейнера, структура показана в листинге 4.3.

|  |
| --- |
| public static string GetMessage(string src)  {  if (File.Exists(src))  {  string text = GetText(src, 1);  if (text.Count(x => x == '\u00A0') < 2)  {  return "";  }  return text.Split(new char[] { '\u00A0' }, StringSplitOptions.RemoveEmptyEntries)[0];  }  return "";  } |

Листинг. 4.3 – Структура метода «GetMessage»

Метод GetMessage извлекает из текст первой страницы контейнера вышеописанным методом GetText. Затем производит поиск наличия специальных символов-маркеров, которые обозначаю секретные метку в тексте. Разделив текст первой страницы по символу-маркеру метод возвращает метку. При отсутствии специальных символов метод вернёт пустую строку.

Как было выше отмечено, проект «PdfSteganoWPF» отвечает создание оконного приложения проекта c использованием технологии WPF.

Технология WPF (Windows Presentation Foundation) является часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов.

Если при создании традиционных приложений на основе WinForms за отрисовку элементов управления и графики отвечали такие части ОС Windows, как User32 и GDI+, то приложения WPF основаны на DirectX. В этом состоит ключевая особенность рендеринга графики в WPF: используя WPF, значительная часть работы по отрисовке графики, как простейших кнопочек, так и сложных 3D-моделей, ложиться на графический процессор на видеокарте, что также позволяет воспользоваться аппаратным ускорением графики.

Одной из важных особенностей является использование языка декларативной разметки интерфейса XAML, основанного на XML: вы можете создавать насыщенный графический интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C# и VB.NET, либо совмещать и то, и другое.

Что вам, как разработчику, предлагает WPF?

* использование традиционных языков .NET-платформы - C# и VB.NET для создания логики приложения;
* возможность декларативного определения графического интерфейса с помощью специального языка разметки XAML, основанном на xml и представляющем альтернативу программному созданию графики и элементов управления, а также возможность комбинировать XAML и C#/VB.NET;
* независимость от разрешения экрана: поскольку в WPF все элементы измеряются в независимых от устройства единицах, приложения на WPF легко масштабируются под разные экраны с разным разрешением;
* новые возможности, которых сложно было достичь в WinForms, например, создание трехмерных моделей, привязка данных, использование таких элементов, как стили, шаблоны, темы и др;
* хорошее взаимодействие с WinForms, благодаря чему, например, в приложениях WPF можно использовать традиционные элементы управления из WinForms;
* богатые возможности по созданию различных приложений: это и мультимедиа, и двухмерная и трехмерная графика, и богатый набор встроенных элементов управления, а также возможность самим создавать новые элементы, создание анимаций, привязка данных, стили, шаблоны, темы и многое другое;
* аппаратное ускорение графики - вне зависимости от того, работаете ли вы с 2D или 3D, графикой или текстом, все компоненты приложения транслируются в объекты, понятные Direct3D, и затем визуализируются с помощью процессора на видеокарте, что повышает производительность, делает графику более плавной;
* создание приложений под множество ОС семейства Windows - от Windows XP до Windows 10.

В тоже время WPF имеет определенные ограничения. Несмотря на поддержку трехмерной визуализации, для создания приложений с большим количеством трехмерных изображений, прежде всего игр, лучше использовать другие средства - DirectX или специальные фреймворки, такие как Monogame или Unity.

Также стоит учитывать, что по сравнению с приложениями на Windows Forms объем программ на WPF и потребление ими памяти в процессе работы в среднем несколько выше. Но это с лихвой компенсируется более широкими графическими возможностями и провышенной производительностью при отрисовке графики.

Основная логика и дизайн приложения описаны в файлах MainWindow.xaml и MainWindow.xaml.cs.

Остальные файлы программы отвечают за правильную сборку программы, обеспечивающую запуск приложения на любом компьютере и различных операционных системах.

В WPF приложениях интерфейс описывается на языке разметки XAML.

XAML (eXtensible Application Markup Language) - язык разметки, используемый для инициализации объектов в технологиях на платформе .NET. Применительно к WPF (а также к Silverlight) данный язык используется прежде всего для создания пользовательского интерфейса декларативным путем. Хотя функциональность XAML только графическими интерфейсами не ограничивается: данный язык также используется в технологиях WCF и WF, где он никак не связан с графическим интерфейсом.

XAML - не является обязательной частью приложения, мы можем обходиться без него, создавая все элементы в файле связанного с ним кода на языке C#. Однако использование XAML все-таки несет некоторые преимущества:

* возможность отделить графический интерфейс от логики приложения, благодаря чему над разными частями приложения могут относительно автономно работать разные специалисты: над интерфейсом - дизайнеры, над кодом логики программисты;
* компактность, понятность, код на XAML относительно легко поддерживать.

При компиляции приложения в Visual Studio код в xaml-файлах также компилируется в бинарное представление кода xaml, которое называется BAML (Binary Application Markup Language). И затем код baml встраивается в финальную сборку приложения - exe или dll-файл.

Интерфейс главного и единственного окна приложения описан в файле MainWindow.xaml. Результат обработки разметки представлен на рисунке 4.2.

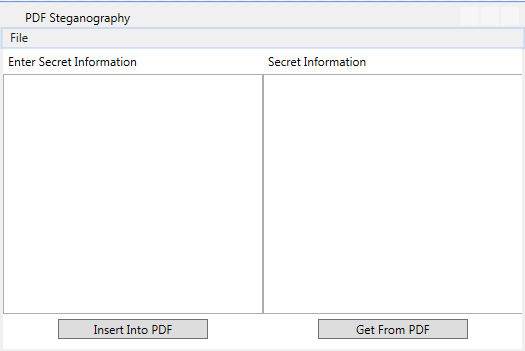


Рисунок 4.2 – Интерфейс программного средства

Окно содержит меню, два текстовых окна, для ввода и вывода информации, и две кнопки. Описание данного окна на языке XAML показано на рисунке 4.3.



Рисунок. 4.3 – Описание окна «MainWindow»

XAML предлагает очень простую и ясную схему определения различных элементов и их свойств. Каждый элемент, как и любой элемент XML, должен иметь открытый и закрытый тег либо элемент может иметь сокращённую форму с закрывающим слешем в конце.

Сначала идет элемент самого высшего уровня - Window, затем идет вложенный элемент Grid - контейнер для других элементов, и в нем уже определены элементы самой страницы.

Основные элементы окна MainWindow:

* StackPanel – элемент компоновки, располагающий все элементы в ряд либо по горизонтали, либо по вертикали в зависимости от ориентации;
* TextBlock – элемент, предназначенный для вывода текстовой информации;
* RichTextBox – элемент, для вывода текстового содержимого, насыщенного форматированием, графикой, предназначен. Можно даже сказать, что он выводит не просто текст, а документы с более сложным форматированием;
* Button – элемент, представляющий собой обычную кнопку;
* Menu – элемент, предназначенный для создания меню окна.

Элемент Menu включает набор элементов MenuItem, которые опять же являются элементами управления содержимым и могут включать другие элементы MenuItem и не только. Также мы можем вложить в меню и другие элементы, которые неявно будут преобразованы в MenuItem.

События нажатия на элементы окна, в наше случае кнопки и элементы меню, отслеживаются и обрабатываются методами, заданными свойством Click. Данные методы подписываются на событие RoutedEvent и принимают в качестве параметров объект, вызвавший это событие, и экземпляр класса RoutedEventArgs, хранящий информацию с вызванном событии.

Нажатие кнопок отслеживается обработчиками, описанными в файле MainWindow.xaml.cs. Структура файла представлена на рисунке 4.4.

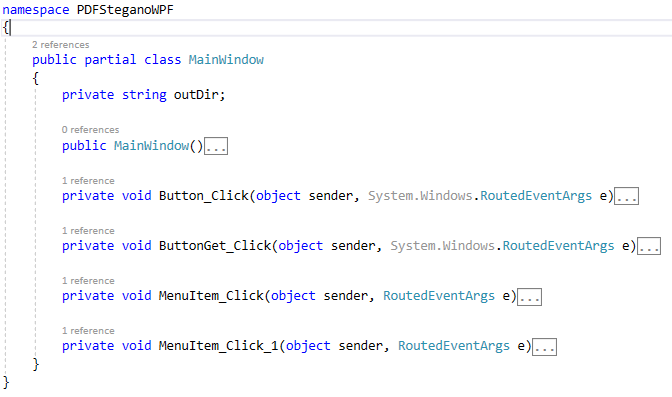


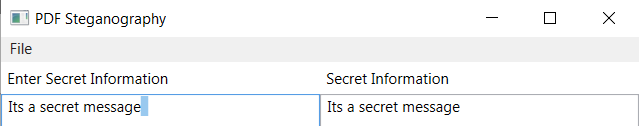
Рисунок 4.4 – Структура файла MainWindow.xaml.cs

Обработчики описаны в классе MainWindow. Конструктор данного класса производит инициализацию элементов окна и базовую настройку приложения. Обработкой событий нажатия на главные кнопки окна занимаются методы Button\_Click и ButtonGet\_Click. За обработку событий меню приложения отвечаю методы MunuItem\_Click и MunuItem\_Click\_1. Полный листинг данного класса представлен в приложении Б.

# 5 Тестирование программного средства

Для оценки работоспособности разработанного приложения были проведены тесты использования памяти и ЦП. Для оценки корректности работы метода были проведены тесты с различными секретными метками.

Для оценки корректности работы метода были использованы два файла out.pdf весом 2 килобайта и UrbanovichShutko.pdf весом около 11 мегабайт.



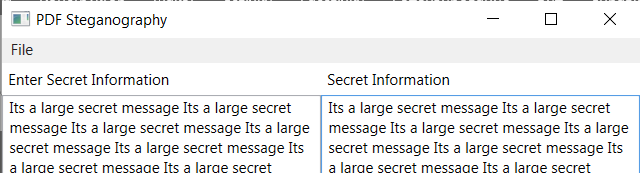
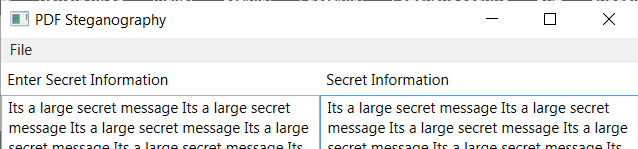
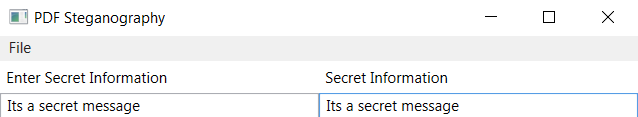


Рисунок 6.1 – Тестирование метода

Как показано на рисунке 6.1 тестирование метода прошло успешно, но следует отметить, что при осаждении большой метки вес маленько файла значительно увеличится. Для большого файла увеличение было не существенно.

Далее была рассмотрена разница между в отображении первых страниц пустого контейнера и контейнера с меткой.

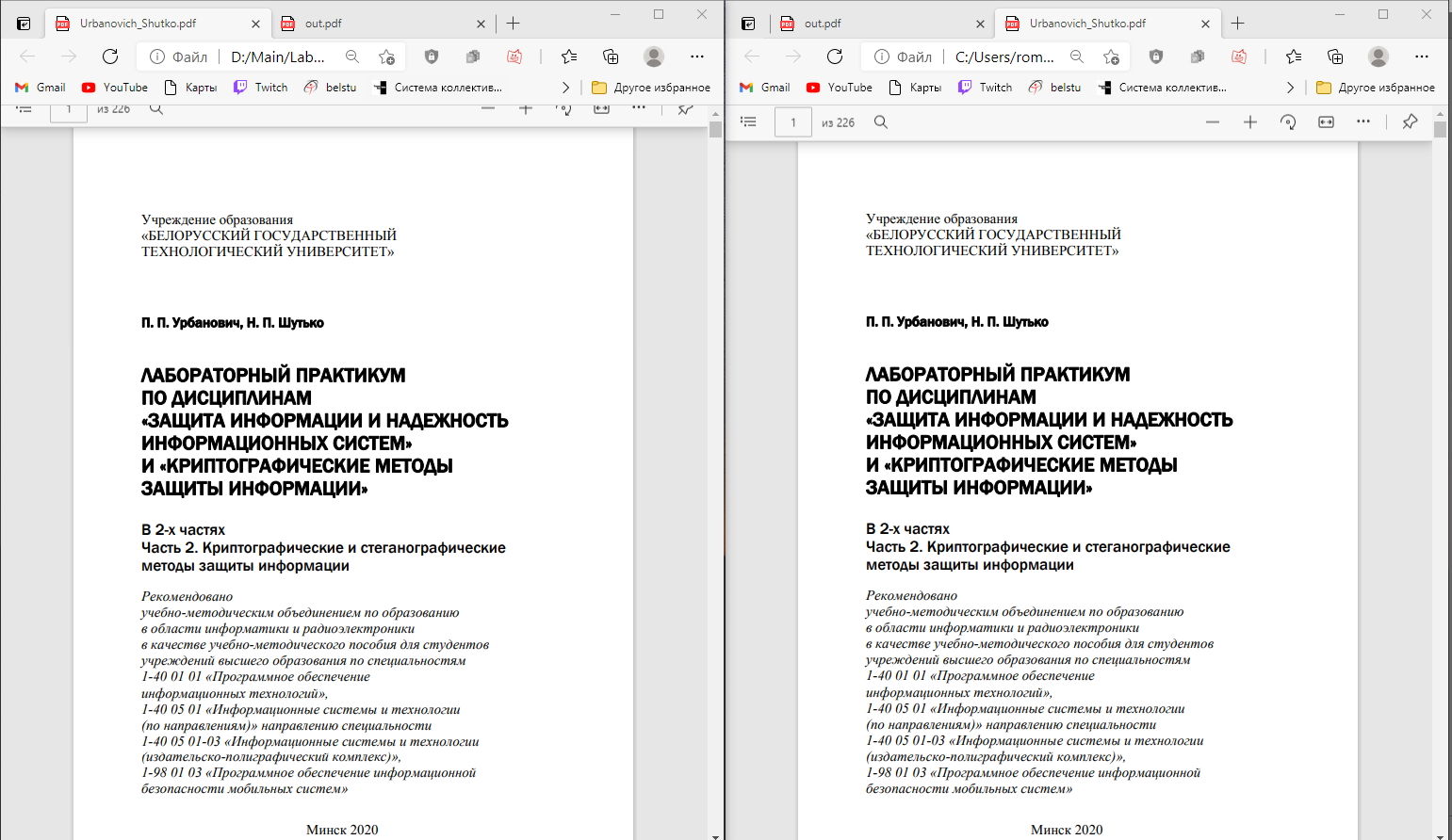
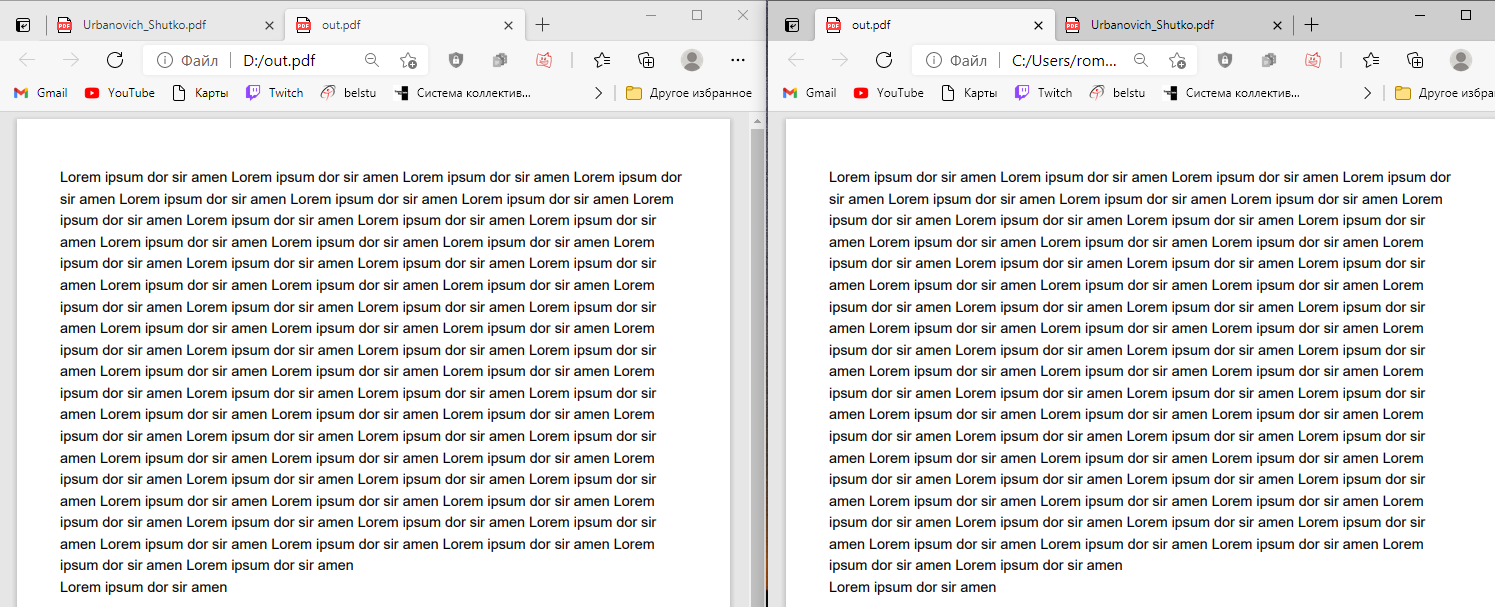
 

Рисунок 6.2 – Файлы без встроенного скрытого сообщения и с ним

Можно с уверенностью сказать, что визуально ничего не изменилось (рисунок 6.2).

Тестирование использование памяти проводится с целью обнаружить проблемы, как утечка памяти.

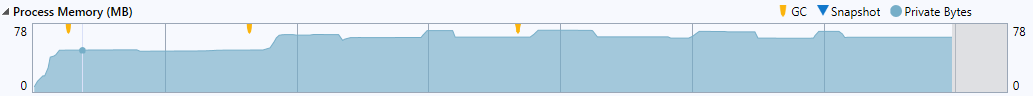


Рисунок 6.3 – Тестирование памяти

Как показано на рисунке 6.3 тестирование приложения на проблему утечку памяти пройдено успешно. Приложение в процессе своей работы использует 40 - 75 мб памяти.

Тестирование использование ЦП проводится с целью нахождения степени загруженности центрального процессора при работе программы.

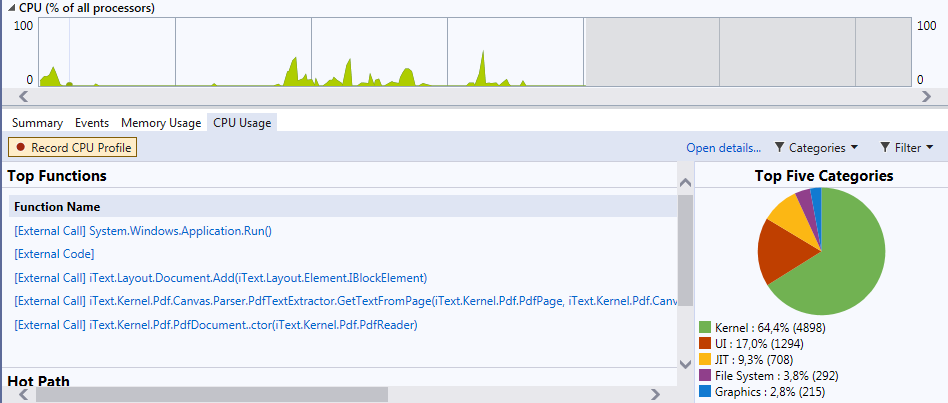


Рисунок 6.4 – Тестирование ЦП

Как показано на рисунке 6.4 тестирование приложения на загруженность процессора основная нагрузка на приложение (42 %) при осаждении метки в контейнер, причём в основном за счёт вызова функций ядра операционной системы.

Протестировав приложение можно сделать вывод, что программное средство «PdfStegano» является нетребовательным к системным характеристикам приложением и может запуститься практически на любом устройстве с оперативной памятью не ниже 256 мб.

# 6 Руководство пользователя

Чтобы запустить приложение нужно запустить exe-файл – PdfSteganoWPF.exe.

При запуске приложения открывается окно приложения «Pdf Steganography» (рисунок 6.1):

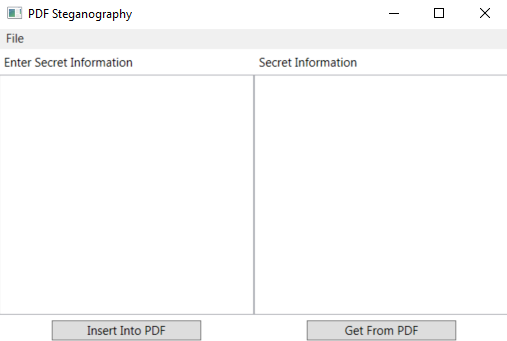


Рисунок 6.1 – Окно приложения «Pdf Steganography».

В данном рабочем окне можно:

* выбрать папку, куда будут сохраняться контейнеры с меткой;
* осадить метку в текстовый контейнер;
* извлечь метку из текстового контейнера;
* создать новый текстовый контейнер.

Для того, что выбрать папку сохранения переходим в меню File> Set Out Folder. В открывшемся окне выбираем нужную папку и жмём ОК (рисунок 6.2). Следует отметить, что по умолчанию выбрана папка PdfSteganography в документах пользователя, открывшего программу.

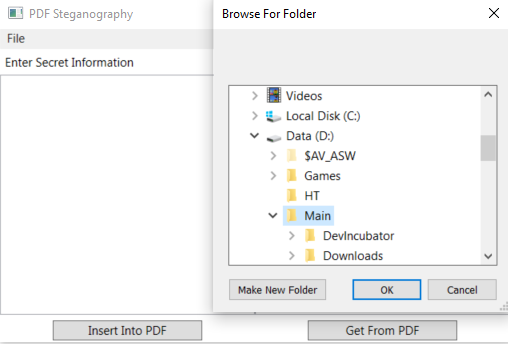


Рисунок 6.2 – Выбор файла контейнера.

Что бы встроить метку в контейнер, пользователю необходимо ввести текс в поле Enter Secret Information, затем нажать кнопку Insert Into PDF (рисунок 6.3).

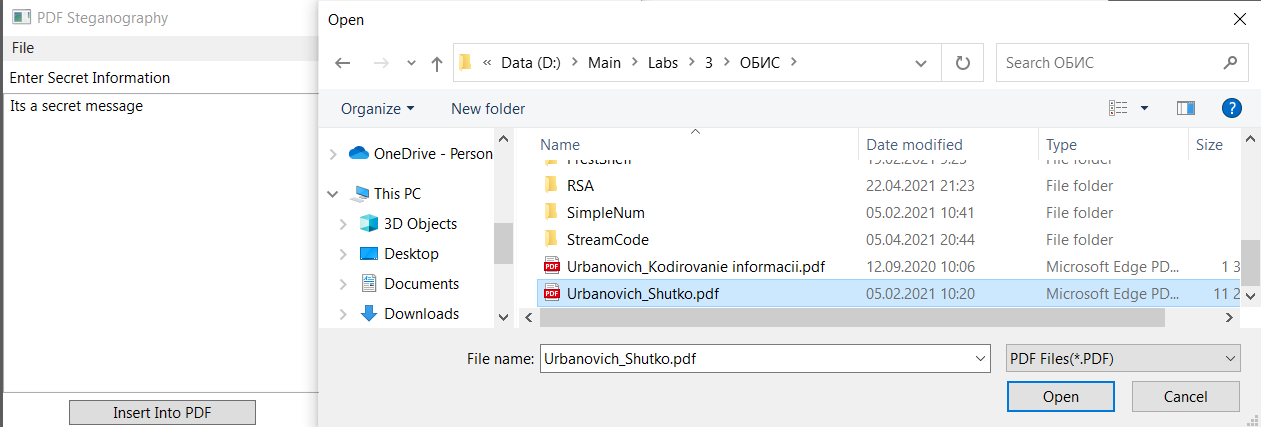


Рисунок 6.3 – Встраивание метки.

Далее необходимо выбрать файл типа pdf, в который будет происходить встраивание метки и нажать Open. После пойдёт процесс осаждения метки, который завершить открытием папки с готовым файлом в проводнике (рисунок 6.4).

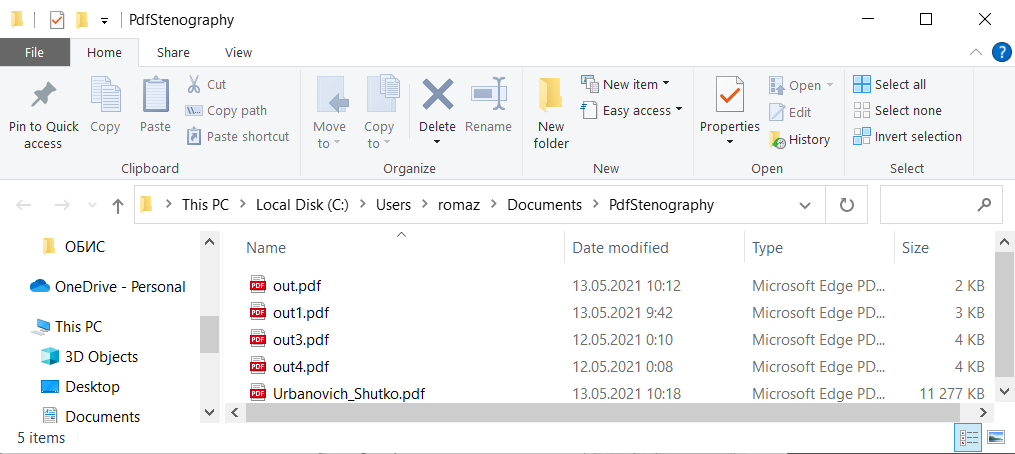


Рисунок 6.4 – Папка с готовыми файлами.

Теперь рассмотрим, как извлечь информацию, которую мы раннее скрыли в выбранном текстовом контейнере.

Что бы извлечь секретную метку из контейнера необходимо нажать кнопку Get From Pdf. Данное действие откроет окно выбора файла, где необходимо выбрать нужный pdf документ (рисунок 6.5).

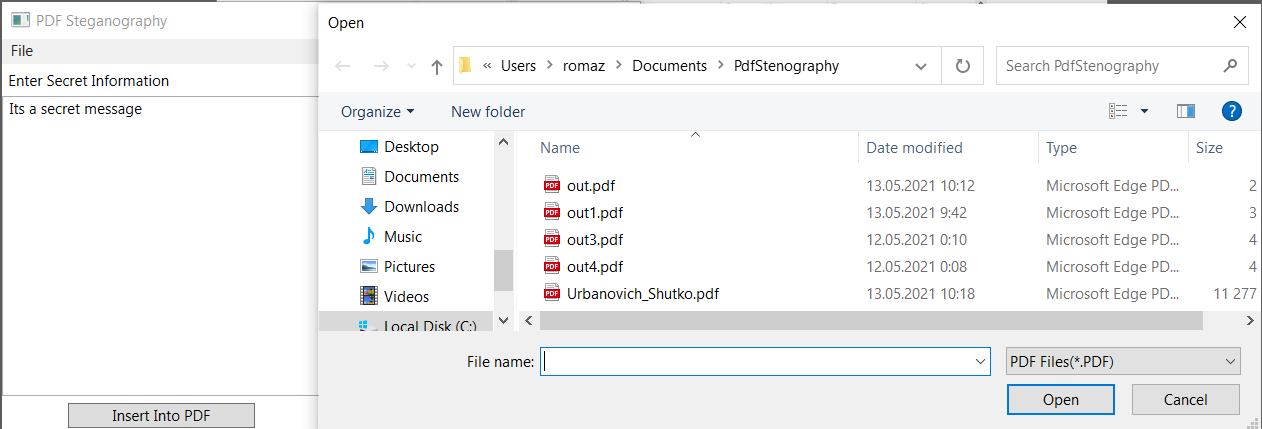


Рисунок 6.5 – Окно выбора файла на открытие.

По умолчанию пользователю покажет папку сохранения готовых документов. Выбрав нужный файл, нажимаем Open. Произойдёт считывание документа и, при наличии секретной метки, она будет выведена в окно Secret Information (рисунок 6.6).

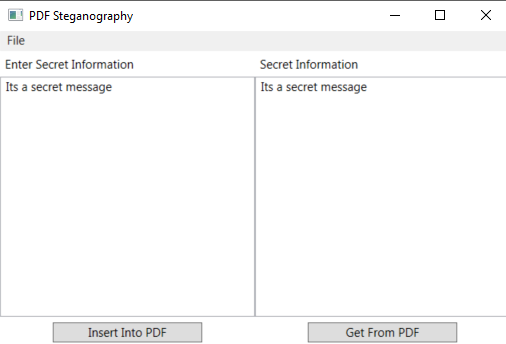


Рисунок 6.6 – Скрытая метка извлечена.

Можно отметить, что программа является очень простой для понимания и даже самый неопытный пользователь сможет использовать ее.

# Заключение

В ходе проделанной работы был приведен и рассмотрен стеганографический процесс сокрытия и обнаружения скрытого сообщения, основанного на методе изменения межстроковых интервалов. Мы рассмотрели его с разных сторон и увидели его проявления себя на разных ситуациях, просмотрели в контексте тестирования. Как было сказано ранее: стеганография, включающая в себя, три раздела (классическую, компьютерную и цифровую) – это метод, позволяющий пользователям скрыть сообщение в другом сообщении, т.е. метод направлен на сокрытие факта передачи информации. Кроме скрытой передачи, на сегодняшний день стеганография является одним из самых перспективных направлений для подтверждения авторского права.

Подтверждение эффективности работы алгоритма при необходимости исследования кем-то сторонним может быть проведена с помощью приложения разработанному мной с использованием технологии WPF. Технология WPF хороша тем предоставляет широкий спектр возможностей по созданию интерактивных настольных приложений.

По итогу разработки моё приложение реализует следующие функции:

* принимать сообщение, введенное пользователем;
* встраивать введенное пользователем сообщение в выбранный текстовый контейнер;
* извлекать сообщение пользователя из текстового контейнера;

Таким образом рассмотренная модификация стеганографического метода изменения межстрокового интервала может быть применена для внесения цифрового водяного знака в электронные документы с целью защиты авторского права на интеллектуальную собственность и подтверждения целостности документа. Также метод может быть использован для размещения различных скрытых стеганографических меток в копиях электронных документов, для выяснения, в случае незаконного копирования и нелегального распространения, принадлежности автору электронных работ.

# Список используемых источников

1. Википедия [Электронный ресурс] / Wikimedia Foundation Inc. – 2020. – Режим доступа: https://ru.wikipedia.org/wiki/Стеганография. – Дата доступа: 27.02.2021.
2. Сущеня, А.А., Блинова Е.А. Применение нескольких стеганографических методов для осаждения скрытых данных в электронных текстовых документах // БГТУ Системный анализ и прикладная информатика 2,2019
3. Урбанович П.П. Защита информации методами криптографии, стеганографии и обфускации. – Минск: БГТУ, 2016, – 220 с.
4. Сущеня, А.А. Способ стеганографического осаждения информации в документ с расширением .DOCX / А. А. Сущеня // XXI Республиканская научная конференция студентов и аспирантов, 19–21 марта, Гомель: сборник научных работ / Гомельский государственный университет имени Ф. Скорины. – C. 303-304.
5. Рябко Б. Я., Фионов А. Н. Основы современной криптографии и стеганографии. — 2-е изд. — М.: Горячая линия — Телеком, 2013. — 232 с

# Приложение А

|  |
| --- |
| using System;  using System.IO;  using System.Text;  using System.Linq;  using iText.Kernel.Colors;  using iText.Kernel.Pdf;  using iText.Kernel.Pdf.Canvas.Parser;  using iText.Kernel.Pdf.Canvas.Parser.Listener;  using iText.Layout;  using iText.Layout.Element;  namespace PDFSteganoLib  {  public static class PdfLineSpacingStegano  {  public static string GetText(string src)  {  if (File.Exists(src))  {  using (var pdfDoc = new PdfDocument(new PdfReader(src)))  {  StringBuilder sb = new StringBuilder();  var strategy = new LocationTextExtractionStrategy();  for (int i = 1; i <= pdfDoc.GetNumberOfPages(); ++i)  {  var page = pdfDoc.GetPage(i);  sb.Append(PdfTextExtractor.GetTextFromPage(page, strategy));  }  return sb.ToString();  }  }  return "";  }  public static string GetText(string src, int pageNum)  {  if (File.Exists(src))  {  using (var pdfDoc = new PdfDocument(new PdfReader(src)))  {  if (pageNum <= 0 || pageNum > pdfDoc.GetNumberOfPages())  return "";  var strategy = new LocationTextExtractionStrategy();  var page = pdfDoc.GetPage(pageNum);  return PdfTextExtractor.GetTextFromPage(page, strategy);  }  }  return "";  }  public static void InsertMessage(string src, string dest, string msg)  {  if (File.Exists(src))  {  int i = GetText(src, 1).Count(x=> x == '\u00A0');  using (PdfReader pdfReader = new PdfReader(src))  using (PdfWriter pdfWriter = new PdfWriter(dest))  {  using (PdfDocument pdfDoc = new PdfDocument(pdfReader, pdfWriter))  {  using (Document doc = new Document(pdfDoc))  {  var par = new Paragraph();  par.SetFixedLeading(-1000f - 111f \* (i + 2));  par.SetCharacterSpacing(-4f);  par.SetWordSpacing(-4f);  Text text = new Text(string.Format("{0}{1}{0}", '\u00A0', msg));  text.SetFontColor(ColorConstants.WHITE, 0);  par.Add(text);  doc.Add(par);  }  }  }  }  }  public static string GetMessage(string src)  {  if (File.Exists(src))  {  string text = GetText(src, 1);  if (text.Count(x => x == '\u00A0') < 2)  {  return "";  }  return text.Split(new char[] { '\u00A0' }, StringSplitOptions.RemoveEmptyEntries)[0];  }  return "";  }  }  } |

# Приложение Б

|  |
| --- |
| using PDFSteganoLib;  using System;  using System.Diagnostics;  using System.IO;  using System.Windows;  using System.Windows.Documents;  using System.Windows.Forms;  using MessageBox = System.Windows.MessageBox;  using OpenFileDialog = System.Windows.Forms.OpenFileDialog;  namespace PDFSteganoWPF  {  public partial class MainWindow  {  private string outDir;  public MainWindow()  {  InitializeComponent();  try {  if (string.IsNullOrEmpty(Properties.Settings.Default.OutDir)) {  string newDir =  Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) + @"\PdfSteganography";  Directory.CreateDirectory(newDir);  Properties.Settings.Default.OutDir = newDir;  Properties.Settings.Default.Save();  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  outDir = Properties.Settings.Default.OutDir;  }  private void Button\_Click(object sender, System.Windows.RoutedEventArgs e)  {  OpenFileDialog dlg = new OpenFileDialog();  string filePath;  string outPath;  string msg;  dlg.Filter = "PDF Files(\*.PDF)|\*.PDF|All Files(\*.\*)|\*.\*";  if (dlg.ShowDialog() == System.Windows.Forms.DialogResult.OK)  {  try  {  filePath = dlg.FileName.ToString();  outPath = outDir + '\\' + Path.GetFileName(filePath);  msg = new TextRange(InputText.Document.ContentStart, InputText.Document.ContentEnd).Text.Replace("\r\n", "");  PdfLineSpacingStegano.InsertMessage(filePath, outPath, msg);  Process.Start("explorer.exe", outDir);  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  }  private void ButtonGet\_Click(object sender, System.Windows.RoutedEventArgs e)  {  OpenFileDialog dlg = new OpenFileDialog();  string filePath;  string msg;  dlg.Filter = "PDF Files(\*.PDF)|\*.PDF|All Files(\*.\*)|\*.\*";  dlg.InitialDirectory = outDir;  if (dlg.ShowDialog() == System.Windows.Forms.DialogResult.OK)  {  try  {  filePath = dlg.FileName.ToString();  msg = PdfLineSpacingStegano.GetMessage(filePath);  OutText.Document.Blocks.Clear();  OutText.Document.Blocks.Add(new Paragraph(new Run(msg)));  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  }  private void MenuItem\_Click(object sender, RoutedEventArgs e)  {  System.Windows.Application.Current.Shutdown();  }  private void MenuItem\_Click\_1(object sender, RoutedEventArgs e)  {  using (var fbd = new FolderBrowserDialog())  {  DialogResult result = fbd.ShowDialog();  if (result == System.Windows.Forms.DialogResult.OK && !string.IsNullOrWhiteSpace(fbd.SelectedPath))  {  Properties.Settings.Default.OutDir = fbd.SelectedPath;  Properties.Settings.Default.Save();  outDir = Properties.Settings.Default.OutDir;  }  }  }  }  } |