**УО «Белорусский государственный технологический университет»**

Факультет **информационных технологий**

Кафедра **информационных систем и технологий**

Специальность **1-40 05 01-03 «Информационные системы и технологии (издательско-полиграфический комплекс)»**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОЙ РАБОТЫ**

**по дисциплине «**Защита информации и надежность информационных систем»

**тема «**Реализация и исследование стеганографического метода на основе изменений межстрочных интервалов в документах-контейнерах формата \*.doc и \*.docx»

|  |  |  |
| --- | --- | --- |
| **Исполнитель** |  | |
| Студент 3 курса группы1 | подпись, дата | Д. В. Божко  инициалы и фамилия |
|  |  |  |
| **Руководитель** |  |  |
| Ассистент | подпись, дата | В. О. Берников  инициалы и фамилия |

|  |  |
| --- | --- |
| **Курсовая работа защищена с оценкой** | **\_\_\_\_\_\_\_\_\_\_** |
| **Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  (подпись) | В. О. Берников  инициалы и фамилия |

Минск 2022

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра информационных систем и технологий

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Смелов

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

**ЗАДАНИЕ**

**к курсовому проектированию по дисциплине**"Защита информации и надежность информационных систем"

Специальность 1-40 05 01 03 «Информационные системы и технологии (издательско-полиграфический комплекс)»

Группа: 1

Студент: Божко Денис Владимирович

**Тема: «**Реализация и исследование стеганографического метода на основе изменений межстрочных интервалов в документах-контейнерах формата \*.doc и \*.docx»

**1. Срок сдачи студентом законченной работы**: «10» мая 2022г.

**2. Исходные данные к проекту:**

**2.1**. Функционально должны быть выполнены следующие задачи:

* Реализовать стеганографический метод на основе изменений межстрочных интервалов межстрочных интервалов в документах-контейнерах формата \*.doc и \*.docx
* Разработать программное средство реализации данного метода или алгоритма.
* Составить руководство пользователя.

**2.2. Требования:**

* Необходимо провести аналитический обзор литературы по теме проекта
* Необходимо описать сферу применимости метода
* Программное средство может быть разработано на любом языке
* Архитектура приложения выбирается разработчиком
* Листинги проекта должны содержать комментарии

**3. Содержание расчетно-пояснительной записки:**

* Введение
* Постановка задачи
* Описание метода
* Описание программного средства
* Тестирование программного средства
* Руководство пользователя
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненного курсового проекта:**

* Теоретическая часть курсового проекта должна быть представлена в формате MS Word.
* Оформление записки должно быть согласно правилам.
* Необходимые схемы, диаграммы и рисунки допускается делать в MS Office Visio или копии экрана (интерфейс).
* Полные листинги проекта представляются в приложении.
* К записке необходимо приложить CD (DVD), который должен содержать: пояснительную записку, листинги и файлы базы данных.

#### Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 22.03.2022 |  |
| 2 | Аналитический обзор литературы по теме проекта | 28.03.2022 |  |
| 3 | Разработка метода | 04.04.2022 |  |
| 4 | Разработка прототипа программного средства | 11.04.2022 |  |
| 5 | Разработка программного средства | 18.04.2022 |  |
| 6 | Тестирование программного средства | 23.04.2022 |  |
| 7 | Написание руководства пользователя | 30.04.2022 |  |
| 8 | Оформление пояснительной записки | 08.05.2022 |  |
| 9 | Сдача проекта | 10.05.2022 |  |

**5. Дата выдачи задания** «15» февраля 2022г.

Руководитель *В.О. Берников*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

Оглавление

[Введение 5](#_Toc102315897)

[1. Аналитический обзор литературы 6](#_Toc102315898)

[1.1. Аналоги 6](#_Toc102315899)

[2. Проектирование 9](#_Toc102315900)

[3. Разработка программного средства 10](#_Toc102315901)

[3.1. Структура стегоконтейнера 10](#_Toc102315902)

[3.2. Создание интерфейса приложения 10](#_Toc102315903)

[3.3. Структура проекта, классы и методы 11](#_Toc102315904)

[4. Тестирование программного средства 14](#_Toc102315905)

[5. Руководство пользователя 20](#_Toc102315906)

[Заключение 24](#_Toc102315907)

[Список используемых источников 25](#_Toc102315908)

[ПРИЛОЖЕНИЕ А 26](#_Toc102315909)

[ПРИЛОЖЕНИЕ Б 28](#_Toc102315910)

Введение

В современном мире стеганография приобретает все большую популярность. Развитие этой области во многом связано с развитием электроники, а также методов хранения, обработки и передачи данных.

Под криптографией понимается способ передачи или хранения информации с учётом сохранения в тайне самого факта такой передачи (хранения). В отличие от криптографии, которая скрывает содержимое тайного сообщения, стеганография скрывает сам факт его существования. Как правило, сообщение будет выглядеть как что-либо иное, например, как изображение, статья, список. Стеганографию обычно используют совместно с методами криптографии, таким образом, дополняя её.

Преимущество стеганографии над чистой криптографией состоит в том, что сообщения не привлекают к себе внимания. Сообщения, факт шифрования которых не скрыт, вызывают подозрение и могут быть сами по себе уличающими в тех странах, в которых запрещена криптография. Таким образом, криптография защищает содержание сообщения, а стеганография защищает сам факт наличия каких-либо скрытых посланий.

Целью данного курсового проекта является изучение стеганографического метода осаждения информации в контейнеры формата \*.doc и \*.docx на основе изменения межстрочных интервалов, а также разработка программного средства, демонстрирующего изученный метод.

На основе поставленной цели были выделены следующие задачи:

* реализовать возможность осаждения и извлечения сообщения в контейнеры формата \*.doc и \*.docx;
* реализовать возможность работы с копией документа;
* реализовать возможность осаждения хеша сообщения;
* составить руководство пользователя.

1. Аналитический обзор литературы
   1. Аналоги

Текстовые стеганографические методы разделяют на два вида: синтаксические и лингвистические. Лингвистические трансформируют текстовое содержимое файла так, чтобы сохранялось смысловое содержание текста. Синтаксические же никак не затрагивают текст сообщения, однако изменяют параметры текста, чтобы осадить или извлечь информацию [1].

В рамках данного курсового проекта будут рассматриваться синтаксические методы стеганографии. Первый из них носит название «Word-Shift coding» [2]. Суть метода состоит в том, что между словами в тексте можно поставить несколько пробелов. Допустим, один пробел – 0, два пробела, стоящих подряд, – единица (либо наоборот). Таким образом, количество информации, которое мы можем записать в файл, равняется количеству пробелов на странице. На лист формата А4 помещается примерно 260 слов, таким образом в документ из одной полной страницы мы можем записать 260 бит или примерно 32 байта информации. Будем использовать данную характеристику для сравнения далее. Метод «Word-Shift coding» имеет разновидность, когда комбинация пробелов будет означать определенную информацию. Например, комбинация пробела и идущих за ним двух смежных пробелов соответствует нулю, а комбинация из двух смежных и одного пробела соответствует единице. Интерпретация данного метода изображена на рисунке 1.1.

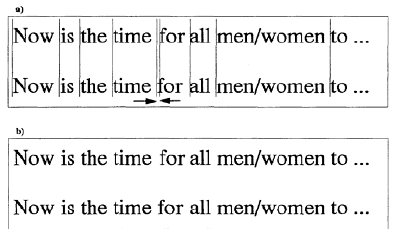


Рисунок 1.1 – Метод «Word-Shift coding»

Следующий метод имеет название «Line-Shift coding» [3]. Можно догадаться, что суть данного метода в изменении значений межстрочных интервалов: регулируя длину строки, некоторой высоте можно задать значение единицы, и другой высоте соответственно нуля. На страницу помещается примерно 32 строки, а значит мы можем записать на страницу примерно 32 бита или 4 байта информации. Оценивая идею метода и количество информации, которое можно осадить с его помощью, можно прийти к выводу, что «Line-Shift coding» проигрывает предыдущему способу по всем направлениям. Изменения в высоте строк можно заметить невооруженным взглядом, так как не все редакторы способны изменить высоту строки на очень маленькое значение, а количество осаждаемой информации слишком мало и не рационально в использовании. Представление метода «Line-Shift coding» изображено на рисунке 1.2.

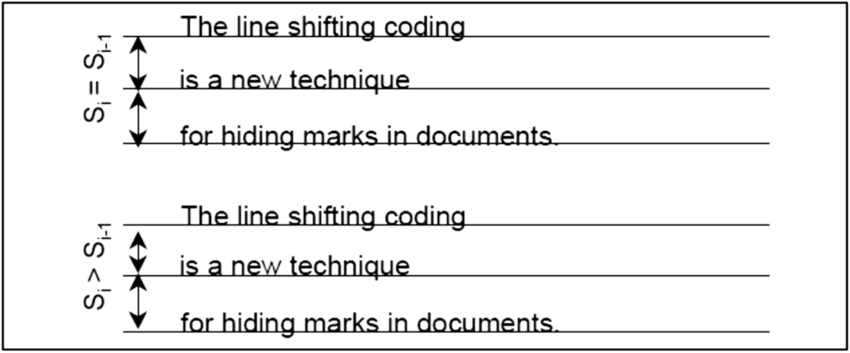


Рисунок 1.2 – Метод «Line-Shift coding»

Третий метод носит название «Feature coding». Данный метод имеет две интерпретации. Первая предлагает создание специфического шрифта, с внесением изменений в отдельные символы. С помощью таких символов-индикаторов можно с легкостью осадить и извлечь информацию. Количество данных, которое мы можем записать, зависит от количества изменений в шрифте. Таким образом, очевидно, что с помощью данного метода можно записать гораздо больше информации, чем в ранее предложенных способах. Тем не менее, недостатком «Feature coding» является его сложность, так как создание, внедрение и использование такого шрифта – крайне трудоемкий процесс. Первая интерпретация метода изображена на рисунке 1.3.

Вторая интерпретация предполагает использование одинаковых по начертанию знаков из одного алфавита. Например, ‘C’ в русском и ‘С’ в английском языках. Аналогично ‘p’ – ‘р’, ‘a’ – ‘а’, ‘o’ – ‘о’ и так далее. Из преимуществ данного способа можно отметить его уникальность. Оценить количество осаждаемой информации на страницу сложно, так как способ основан на локальном алфавите и найденных соответствиях из других языков.

Метод, который будет использован в курсовом проекте, не имеет своего названия и является некоторой модификацией и комбинацией «Word-Shift coding» и «Line-Shift coding» [4]. Реализация данного метода возможна благодаря особенностям редактора Microsoft Word. В нем можно задавать высоту не только для всей строки, но и для отдельных символов. В данном случае изменяться будут неотображаемые символы, такие как пробел. Примем, что смещение пробела на один пиксель вверх соответствует единице, а смещение пробела на один пиксель вниз – нулю. Количество информации, которое мы сможем записать, будет, как и в первом предложенном аналоге ~32 байта на страницу. Метод представлен на рисунке 1.3. Большим преимуществом является его скрытность. Очевидно, что определить различия в высоте пробела не просто, особенно, когда она отличается на 1 пиксель и не влияет на остальной текст.

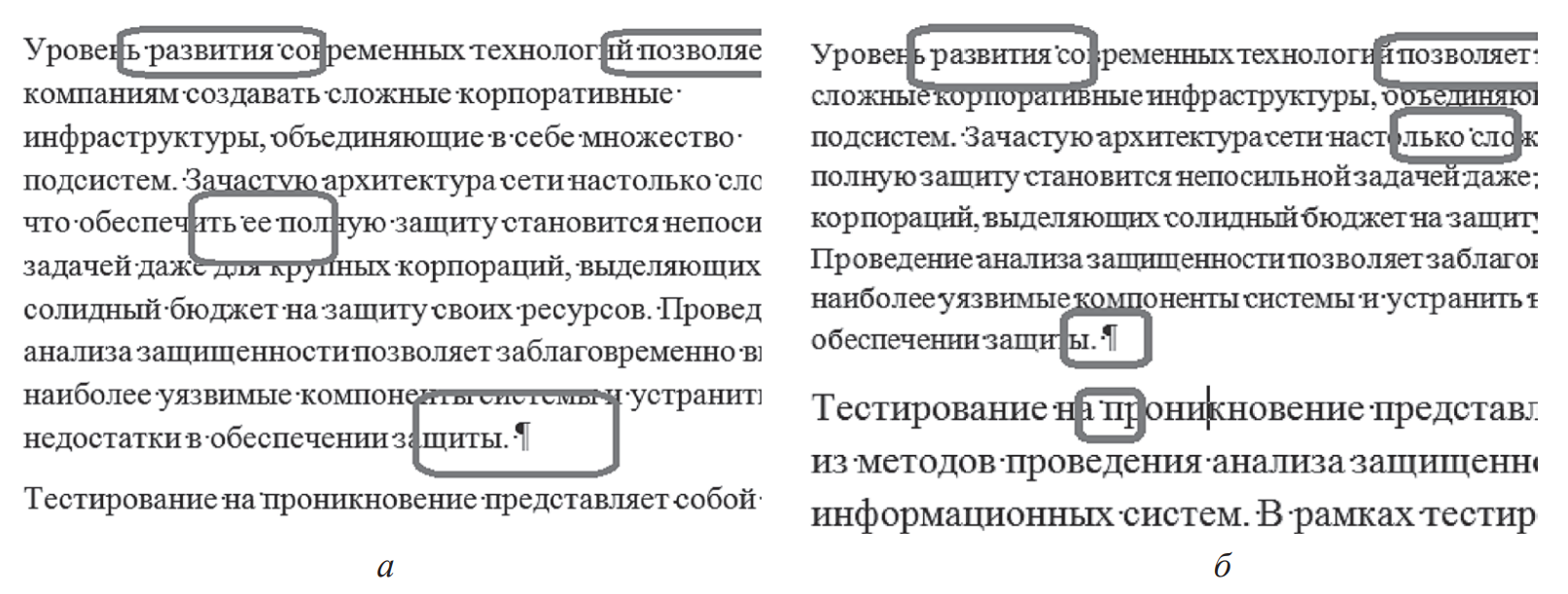


Рисунок 1.3 а – Текст со смещенными неотображаемыми символами;

б – Текст со смещенными неотображаемыми символами с изменением начертания и размера шрифта

Таким образом, результатом данной главы стал анализ методов текстовой стеганографии. На основе выделенных преимуществ и недостатков, для реализации был выбран метод, сочетающий в себе преимущества других аналогов. Проектирование, разработка и тестирование программного средства будет описана в следующих главах.

1. Проектирование

В данной главе описаны технологии, которые были использованы для разработки программного средства, для осаждения и извлечения информации в стеганографический контейнер формата \*.doc и \*.docx. Для начала рассмотрим системы и технологии, которые мы будем использовать в данном курсовом проекте.

Open XML – это открытый стандарт для текстовых документов, презентаций и электронных таблиц, который может быть реализован в разных приложениях на различных платформах [5]. Open XML позволяет точно представлять существующие текстовые документы, презентации и электронные таблицы, которые кодируются в двоичных форматах, используемых приложениями Microsoft Office. Цель стандарта Open XML – снять зависимость документов от приложений Microsoft Office, в которых они были созданы, чтобы с ними можно было работать в других приложениях независимо от формата и без потери данных.

Windows Presentation Foundation — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML [6]. WPF представляет собой обширный API-интерфейс для создания настольных графических программ, имеющих насыщенный дизайн и интерактивность.

C# — это объектно- и компонентно-ориентированный язык программирования. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов.

Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения.

1. Разработка программного средства
   1. Структура стегоконтейнера

На первом этапе разработки программного средства была рассмотрена и изучена структура стегоконтейнера. В данном случае контейнером будет являться файл формата \*.doc либо \*.docx. Оба этих формата являются файлами Open XML. Файл Open XML хранится в ZIP-архиве, чтобы уменьшить размер файла. Структуру файла Open XML можно просмотреть с помощью средства просмотра ZIP-файлов. Документ Open XML состоит из нескольких частей документа. Отношения между частями хранятся в частях документа. Формат ZIP поддерживает произвольный доступ к каждой части. Например, приложение может удалить все примечания из текстового документа без анализа его содержимого.

Структурно документ Open XML — это пакет Open Packaging Conventions (OPC). Как говорилось ранее, пакет состоит из набора частей документа. У каждой части есть имя, которое состоит из последовательности сегментов или пути, например "/word/theme/theme1.xml". Пакет содержит часть [Content\_Types].xml, которая позволяет определить тип содержимого всех частей документа в пакете. Набор явных связей для пакета или части содержится в части связей с расширением .rels (рисунок 3.1).

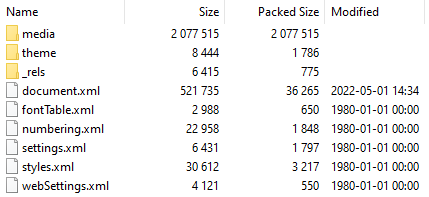


Рисунок 3.1 – Структура стегоконтейнера Open XML

Части документа в пакете Open XML создаются в виде разметки XML. Поскольку XML – это обычный структурированной текст, мы можем просмотреть содержимое части документа с помощью средства чтения текста или выполнить синтаксический анализ содержимого с помощью таких процессов, как XPath.

* 1. Создание интерфейса приложения

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.

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, простая поддержка кода.

Разработка интерфейса осуществлялась с использованием стандартных элементов, предлагаемых средой разработки Visual Studio. Конечный вариант представлен на рисунке 3.2.

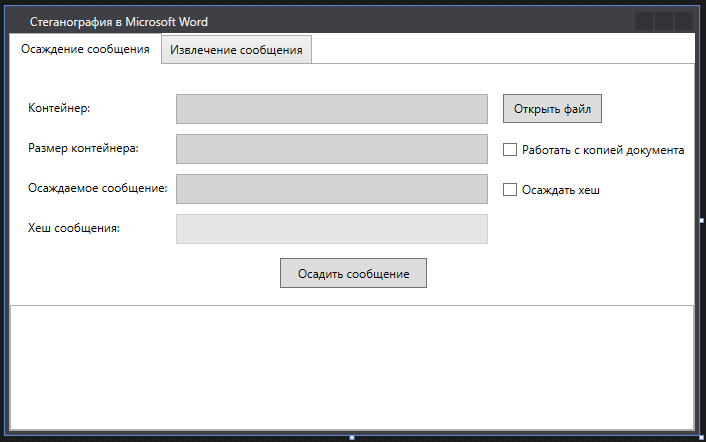


Рисунок 3.2 – Интерфейс приложения

Листинг исходного кода XAML представлен в приложении А.

* 1. Структура проекта, классы и методы

Приложение было разработано в IDE Visual Studio и представляет собой единый проект, структура которого представлена на рисунке 3.3.

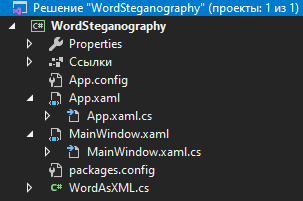


Рисунок 3.3 – Структура проекта

Самыми важными объектами в данной структуре являются три объекта: MainWindow.xaml, MainWindow.xaml.cs, WordAsXML.cs.

MainWindow.xaml отвечает за графический интерфейс приложения. Подробно он был описан в предыдущем разделе.

MainWindow.xaml.cs является классом C#, который инициализирует программный компонент, а также содержит обработчики событий для всех UI элементов интерфейса. Его структурная схема представлена на рисунке 3.4.

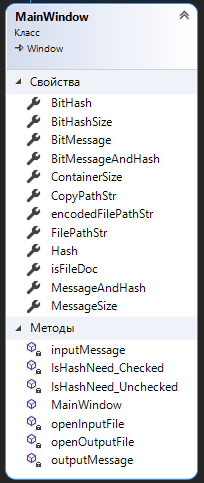


Рисунок 3.4 – Структурная схема класса MainWindow.xaml.cs

Как можно заметить, класс содержит свойства для управления приложением и установки определенных состояний программы.

WordAsXML.cs является классом C#, аналогичным второму, однако, содержит основные методы работы с контейнером, как с Open XML объектом. Структурная схема этого класса представлена на рисунке 3.5.

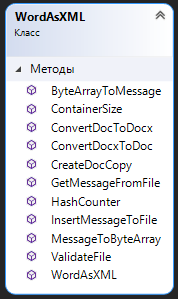


Рисунок 3.5 – Структурная схема класса WordAsXML.cs

Как можно заметить, именно этот класс с помощью описанных методов может рассчитывать размер открытого контейнера, генерировать хеш, проверять исходный формат документа и осуществлять валидацию, преобразовывать формат .doc в .docx, создавать рабочую копию документа, а также осаждать и извлекать сообщение.

Листинг кода классов представлен в приложении Б.

1. Тестирование программного средства

На данном этапе курсового проекта проведем тестирование разработанного приложения, чтобы удостовериться в правильности выполнения функция, а также проверить программу на отказоустойчивость.

Рассмотрим осаждение сообщения. Кроме стандартной функции встраивания текстового сообщения в контейнер формата \*.doc и \*.docx, присутствуют дополнительные функции, такие как работа с копией и осаждение хеша сообщения.

Для начала протестируем поддержку разных форматов документа. Для этого откроем файлы Word начиная с ранних версий. Для теста были выбраны файлы двух различных форматов: документ Word 97 – 2003 (.doc) и документ Word 2019 (.docx). Все документы были успешно открыты как для записи, так и для чтения (рисунок 4.1).

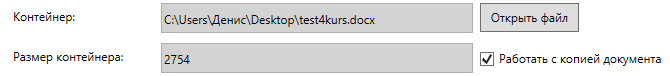


Рисунок 4.1 – Отчет об успешном открытии контейнера

Также для документов приложением были созданы рабочие копии с постфиксом «v2», чтобы сохранить оригинал. На рисунке 4.2 можно заметить, что файл успешно создался. Размер контейнера был также успешно посчитан, и был равен для всех версий, что и ожидалось.



Рисунок 4.2 – Созданная вторая версия документа

Далее протестируем функцию подсчета размера контейнера. Для этого создадим три файла. Первый файл будет полностью пустой. Второй файл будет содержать один пробел. Последний файл будет содержать 1000 пробелов. Результат открытия пустого файла изображен на рисунке 4.3.

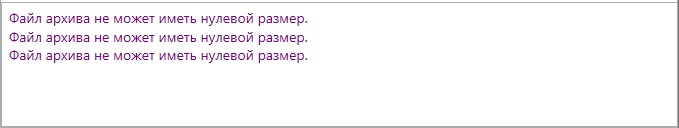


Рисунок 4.3 – Открытие пустого файлов

Как можно заметить, полностью пустые файлы вовсе не могут быть открыты. Теперь рассмотрим открытие второго контейнера (рисунок 4.4).

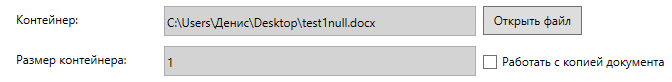


Рисунок 4.4 – Открытие файла с одним пробелом

Как можно заметить, открытие не пустого файла было успешно обработано. Контейнер был открыт, а его размер посчитан.

Рассмотрим третий случай и откроем файл, в который записано 1000 пробелов. Результат представлен на рисунке 4.5.

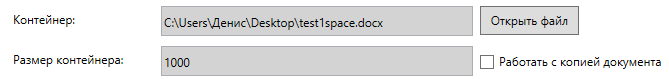


Рисунок 4.5 – Открытие файла с 1000 пробелов

Как и ожидалось, приложение сработало без ошибок. Файл открыт для осаждения сообщения и его размер посчитан верно.

Протестируем открытие поврежденных файлов. Для этого создадим файл формата .txt и переименуем его разрешение в .docx. Результат открытия такого файла представлен на рисунке 4.6.



Рисунок 4.6 – Открытие поврежденного файла

Даже если файл поврежден, приложение позволяет дальше продолжить работу с новыми контейнерами по извлечению и осаждению.

Проверим осаждение сообщения на нескольких примерах. В первом случае попробуем осадить пустое сообщение (рисунок 4.7).

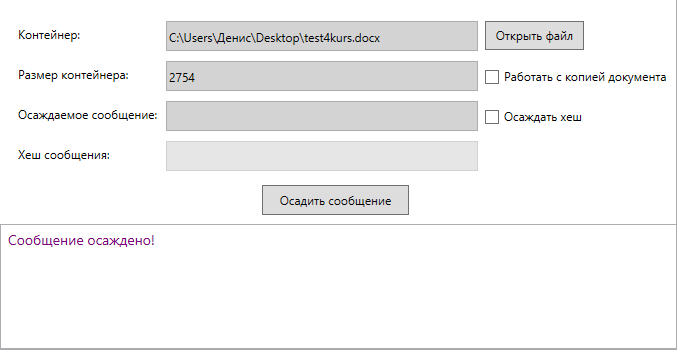


Рисунок 4.7 – Успешное осаждение пустого сообщения

Как можно заметить, сообщение было осаждено успешно, хоть и пустое. Далее попробуем вставить строку произвольной длины. Не будем включать опцию осаждения хеша и заодно проверим эту опцию. Результат осаждения сообщения без контрольного хеша представлен на рисунке 4.8.

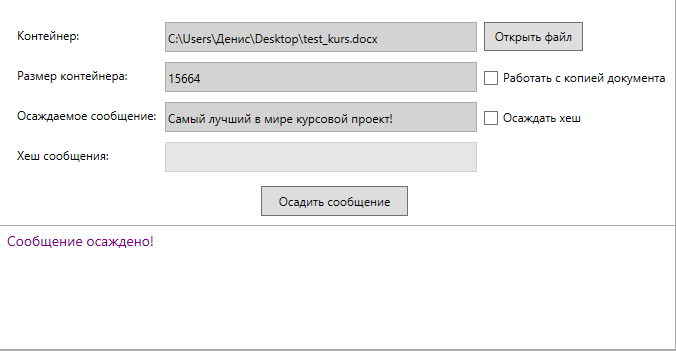


Рисунок 4.8 – Успешное осаждение сообщения без хеша

Как можно заметить по сообщению в окне статуса, сообщение было успешно осаждено, а контейнер открыт для извлечения сообщения.

Также протестируем осаждения того же самого сообщения с хешем. Окно приложения и вывод изображены на рисунке 4.9.

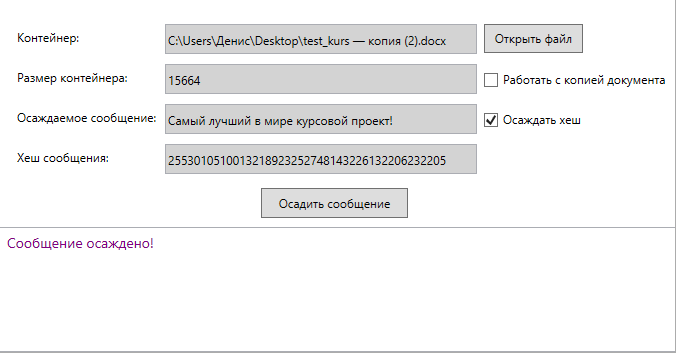


Рисунок 4.9 – Осаждение сообщения с хешем

Создадим контейнер с небольшим размером и попробуем осадить сообщение, превышающее размер контейнера. Тестирование будем проводить без осаждения хеша. Результат выполнения теста отображен на рисунке 4.10.

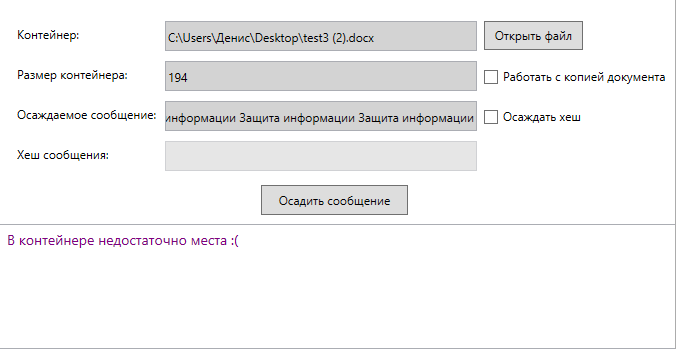


Рисунок 4.10 – Осаждение сообщения, превышающего размер контейнера

Далее протестируем функционал извлечения сообщения из стегоконтейнера. Проверим извлечение сообщения с осажденным хешем на основе одного из ранее созданных файлов. В данном случае ожидается, что будет извлечено сообщение и осажденный хеш. Контрольный хеш будет вычислен на основе извлеченного сообщения и сравнен с осажденным (рисунок 4.11).

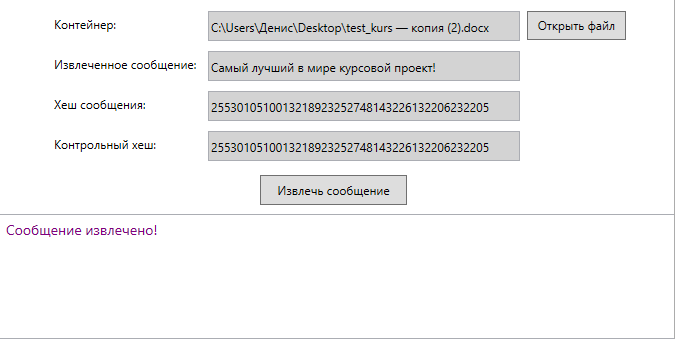


Рисунок 4.11 – Извлечения сообщения с хешем

Как можно заметить, сообщение было извлечено, а также вычислен контрольный хеш. Он идентичен хешу осажденному в контейнере.

Также проверим извлечение сообщения, осажденного без вычисления контрольного хеша. В данном случае извлекается только сообщение и вычисляется на его основе хеш. Данный тест представлен на рисунке 4.12.

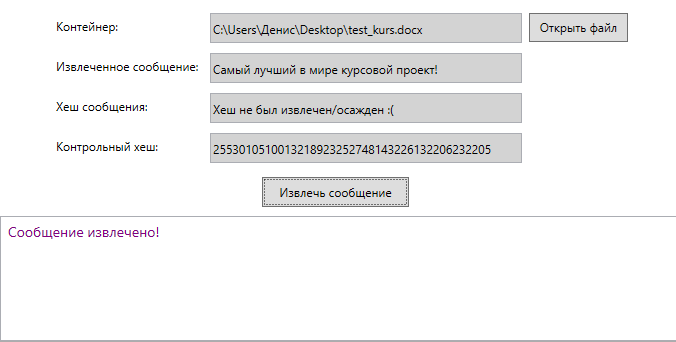


Рисунок 4.12 – Извлечение сообщения без хеша

Протестируем так же извлечение поврежденного сообщения. Для этого возьмем ранее созданный контейнер с осажденным сообщением и удалим некоторую его часть. После чего снова попытаемся извлечь сообщение с хешем. На рисунке 4.13 видно, что сообщение повреждено и отличается от начального.

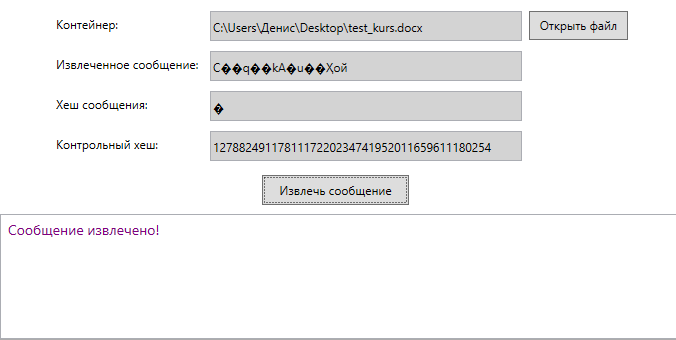


Рисунок 4.13 – Извлечение поврежденного сообщения

Также иногда может случиться ситуация, когда стегоконтейнер не может быть открыт, так как используется в других целях системой. В таком случае пользователь получит результат, представленный на рисунке 4.14).

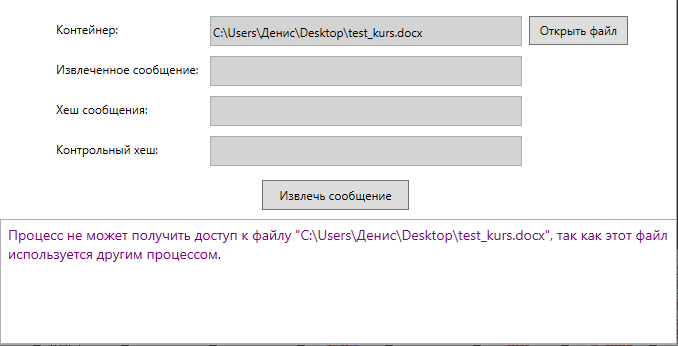


Рисунок 4.14 – Попытка использовать открытый контейнер

1. Руководство пользователя

Основной вид приложения изображен на рисунке 5.1.

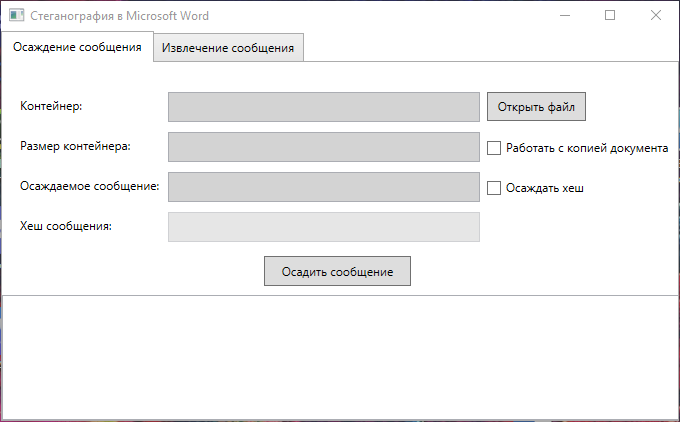


Рисунок 5.1 – Основной вид приложения

Для осаждения сообщения в приложении существуют соответствующая секция. Чтобы начать работу с предпочитаемым контейнером, нужно нажать кнопку «Открыть файл», после чего в диалоговом окне найти директорию, выбрать нужный файл. В любой директории будут отображаться только файлы формата \*.doc и \*.docx (рисунок 5.2).

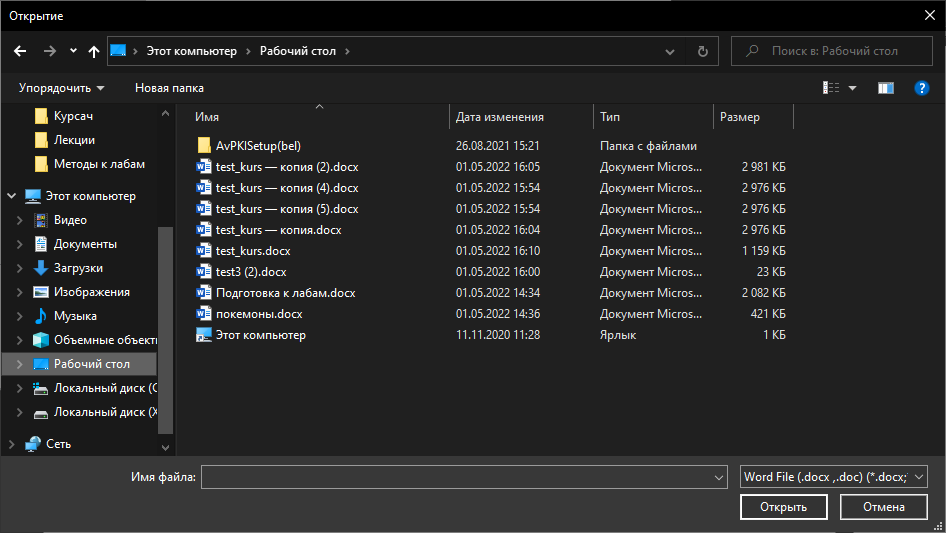


Рисунок 5.2 – Выбор контейнера

После выбора контейнера, можно начать осаждение сообщения.

Чтобы избежать повреждения исходного файла, присутствует функция «Работать с копией документа». Она должна быть выбрана галочкой предварительно, до открытия контейнера (рисунок 5.3).



Рисунок 5.3 – Функция «Работать с копией документа»

Если контейнер открыт успешно, в поле «Контейнер» отобразится путь к файлу, а в поле «Размер контейнера» будет отображаться максимальное количество бит, которое может быть записано в файл (рисунок 5.4).

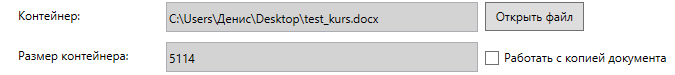


Рисунок 5.4 – Успешное открытие контейнера

В случае неудачного открытия файла, в окне статуса будет отображена причина неудачного открытия. Чаще всего это связано с поврежденными файлами, либо старым форматом документов.

Для проверки целостности сообщения предусмотрена функция осаждения хеша сообщения. Она может быть включена в любой момент, до осаждения сообщения, как показано на рисунке 5.5.

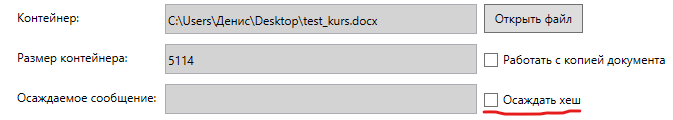


Рисунок 5.5 – Функция «Осаждать хеш»

Далее для осаждения сообщения нужно вписать его в поле «Осаждаемое сообщение» и нажать кнопку «Осадить сообщение». В случае успешного завершения процесса, окно статуса отобразит сообщение об успешном осаждении сообщения, а также в поле хеша, если была выбрана опция его осаждения, отобразится контрольный хеш (рисунок 5.6).

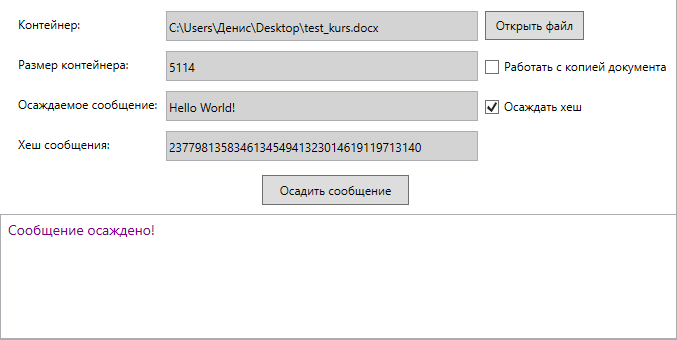


Рисунок 5.6 – Успешное осаждание сообщения

Далее можно осадить сообщение в новый контейнер, для этого достаточно снова нажать на кнопку «Открыть файл» и повторить действия.

Для извлечения сообщения, нужно перейти на вкладку «Извлечение сообщения» и открыть контейнер с сообщением. В случае успешно открытого сообщения в поле «Контейнер» будет отображен путь к файлу. Если же файл открыт с ошибкой, текст ошибки будет также отображен в окне статуса.

Далее для извлечения сообщения достаточно нажать кнопку «Извлечь сообщение». В случае успеха, сообщение появится в соответствующем поле, а также будет вычислен хеш извлеченного сообщения и помещен в поле рядом с уже извлеченным хешем, если таковой осаждался (рисунок 5.7)

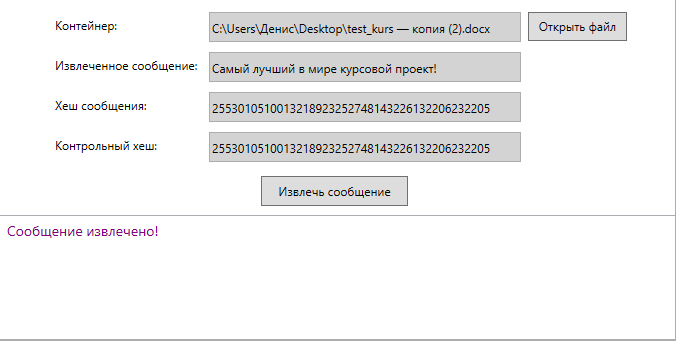


Рисунок 5.7 – Успешно извлеченное сообщение

Затем можно извлечь сообщение из другого контейнера, для этого достаточно снова нажать на кнопку «Открыть файл» и повторить действия.

В случае получения какой-либо ошибки, она будет отображена в окне статуса. Пример изображен на рисунке 5.8.

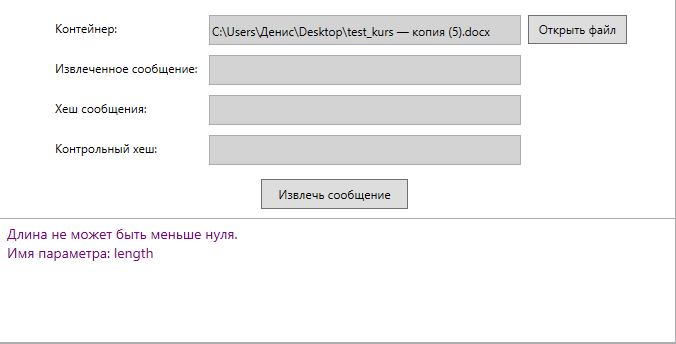


Рисунок 5.8 – Ошибка при извлечении данных из контейнера

Заключение

В ходе выполнения курсового проекта была достигнута поставленная цель по разработке приложения для осаждения и извлечения информации в контейнеры формата \*.doc и \*.docx. Были созданы необходимые классы, функции и пользовательский интерфейс.

При разработке проекта выполнены следующие задачи:

* реализовать возможность осаждения и извлечения сообщения в контейнеры формата \*.doc и \*.docx;
* реализовать возможность работы с копией документа;
* реализовать возможность осаждения хеша сообщения;
* составить руководство пользователя.

В процессе выполнения проекта были рассмотрены различные текстовые стеганографические методы. Был проведен их анализ и сравнение. В результате чего для реализации был выбран текущий метод.

На следующем этапе было спроектировано приложение, а также выбраны технологии для его реализации. Этими технологиями стали система для построения клиентских приложений WPF, язык программирования C#, библиотека для работы с контейнером Open XML, а также IDE Visual Studio.

Результатом третьего этапа стало приложение, рассчитанное для использования в ОС Windows. Приложение предоставляет все регламентированные функции, а также графический интерфейс.

По результатам тестирования на следующих этапах были устранены и исправлены всевозможные ошибки, а также проверена работоспособность, и отказоустойчивость программного средства.

Было составлено руководство пользователя.

По результатам выполнения программы можно сделать вывод о том, что разработанная программа работает должным образом и требования технического задания полностью выполнены.

Список используемых источников

1. Синтаксические методы [Электронный ресурс]. Режим доступа: https://studfile.net/preview/4452629/page:139/. Дата доступа: 30.04.2022
2. Example of Word-Shift coding [Электронный ресурс]. Режим доступа: https://www.researchgate.net/figure/Example-of-word-shift-coding-a-the-top-text-line-has-added-spacing-before-the-for-b\_fig1\_228672143. Дата доступа: 30.04.2022
3. Example of Line-Shift coding [Электронный ресурс]. Режим доступа: https://www.researchgate.net/figure/Example-of-line-shift-coding-The-second-line-has-been-shifted-up-by-inch\_fig1\_3559650. Дата доступа: 30.04.2022
4. Блинова, Е. А. Применение нескольких стеганографических методов для осаждения скрытых данных в электронных текстовых // Е. А. Блинова, А. А. Сущеня // Системный анализ и прикладная информатика. – 2019. – № 2. – С. 32-38.
5. Руководство по Open XML [Электронный ресурс]. Режим доступа: https://docs.microsoft.com/en-us/office/open-xml/open-xml-sdk. Дата доступа: 02.05.2022
6. Руководство по WPF [Электронный ресурс]. Режим доступа: https://metanit.com/sharp/wpf. Дата доступа: 02.05.2022

ПРИЛОЖЕНИЕ А

Файл MainWindow.xaml

|  |
| --- |
| <Window x:Class="WordSteganography.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:WordSganography"  mc:Ignorable="d"  Title="Стеганография в Microsoft Word" Height="429" Width="694">  <TabControl>  <TabItem>  <TabItem.Header>  <StackPanel Orientation="Horizontal">  <TextBlock Margin="4">Осаждение сообщения</TextBlock>  </StackPanel>  </TabItem.Header>  <TabItem.Content>  <Grid>  <Label Content="Контейнер:" HorizontalAlignment="Left" Margin="11,26,0,0" Width="81" Height="30" HorizontalContentAlignment="Left" VerticalContentAlignment="Center" VerticalAlignment="Top"/>  <TextBox AllowDrop="True" Name="FilePath" HorizontalAlignment="Left" Margin="164,28,0,0" Width="312" Height="30" Padding="0,7,0,0" VerticalAlignment="Top" Background="LightGray"/>  <Button Content="Открыть файл" HorizontalAlignment="Right" Margin="0,28,90,0" Height="29" VerticalAlignment="Top" Width="99" Click="openInputFile"/>  <Label Content="Размер контейнера:" HorizontalAlignment="Left" Margin="11,66,0,0" Width="126" Height="30" HorizontalContentAlignment="Left" VerticalContentAlignment="Center" VerticalAlignment="Top"/>  <TextBox AllowDrop="True" Name="containerSizeField" HorizontalAlignment="Left" Margin="164,68,0,0" Width="312" Height="30" Padding="0,7,0,0" VerticalAlignment="Top" Background="LightGray"/>  <CheckBox Content="Работать с копией документа" Name="isCopyNeed" HorizontalAlignment="Right" Margin="0,69,3,0" Width="187" Height="30" VerticalAlignment="Top" VerticalContentAlignment="Center"/>  <Label Content="Осаждаемое сообщение:" HorizontalAlignment="Left" Margin="11,106,0,0" Width="156" Height="30" HorizontalContentAlignment="Left" VerticalContentAlignment="Center" VerticalAlignment="Top"/>  <TextBox AllowDrop="True" Name="messageField" HorizontalAlignment="Left" Margin="164,108,0,0" Width="312" Height="30" Padding="0,7,0,0" VerticalAlignment="Top" Background="LightGray"/>  <CheckBox Content="Осаждать хеш" Name="isHashNeed" HorizontalAlignment="Right" Margin="0,109,3,0" Width="187" Height="30" VerticalAlignment="Top" VerticalContentAlignment="Center" Checked="IsHashNeed\_Checked" Unchecked="IsHashNeed\_Unchecked"/>  <Label Content="Хеш сообщения:" HorizontalAlignment="Left" Margin="11,146,0,0" Width="106" Height="30" HorizontalContentAlignment="Left" VerticalContentAlignment="Center" VerticalAlignment="Top"/>  <TextBox IsReadOnly="True" IsEnabled="False" AllowDrop="True" Name="hashField" HorizontalAlignment="Left" Margin="164,148,0,0" Width="312" Height="30" Padding="0,7,0,0" VerticalAlignment="Top" Background="LightGray"/>  <Button Content="Осадить сообщение" HorizontalAlignment="Right" Margin="0,192,265,0" Height="30" VerticalAlignment="Top" Width="147" Click="inputMessage"/>  <TextBox IsReadOnly="True" Name="statusField1" HorizontalAlignment="Left" Margin="-2,0,-2,-2" TextWrapping="Wrap" VerticalAlignment="Bottom" Height="125" Width="684" Text="" Foreground="Purple" FontSize="14" Padding="5"></TextBox>  </Grid>  </TabItem.Content>  </TabItem>  <TabItem>  <TabItem.Header>  <StackPanel Orientation="Horizontal">  <TextBlock Margin="4">Извлечение сообщения</TextBlock>  </StackPanel>  </TabItem.Header>  <TabItem.Content>  <Grid>  <Label Content="Контейнер:" HorizontalAlignment="Left" Margin="49,26,0,0" Width="81" Height="30" HorizontalContentAlignment="Left" VerticalContentAlignment="Center" VerticalAlignment="Top"/>  <TextBox IsReadOnly="True" AllowDrop="True" Name="outputFilePath" HorizontalAlignment="Left" Margin="208,28,0,0" Width="312" Height="30" Padding="0,7,0,0" VerticalAlignment="Top" Background="LightGray"/>  <Button Content="Открыть файл" HorizontalAlignment="Right" Margin="0,28,46,0" Height="29" VerticalAlignment="Top" Width="99" Click="openOutputFile"/>  <Label Content="Извлеченное сообщение:" HorizontalAlignment="Left" Margin="49,66,0,0" Width="155" Height="30" HorizontalContentAlignment="Left" VerticalContentAlignment="Center" VerticalAlignment="Top"/>  <TextBox IsReadOnly="True" AllowDrop="True" Name="outputMessageField" HorizontalAlignment="Left" Margin="208,68,0,0" Width="312" Height="30" Padding="0,7,0,0" VerticalAlignment="Top" Background="LightGray"/>  <Label Content="Хеш сообщения:" HorizontalAlignment="Left" Margin="49,106,0,0" Width="156" Height="30" HorizontalContentAlignment="Left" VerticalContentAlignment="Center" VerticalAlignment="Top"/>  <TextBox IsReadOnly="True" AllowDrop="True" Name="outputHashField" HorizontalAlignment="Left" Margin="208,108,0,0" Width="312" Height="30" Padding="0,7,0,0" VerticalAlignment="Top" Background="LightGray"/>  <Label Content="Контрольный хеш:" HorizontalAlignment="Left" Margin="49,146,0,0" Width="121" Height="30" HorizontalContentAlignment="Left" VerticalContentAlignment="Center" VerticalAlignment="Top"/>  <TextBox IsReadOnly="True" AllowDrop="True" Name="controlHashField" HorizontalAlignment="Left" Margin="208,148,0,0" Width="312" Height="30" Padding="0,7,0,0" VerticalAlignment="Top" Background="LightGray"/>  <Button Content="Извлечь сообщение" HorizontalAlignment="Right" Margin="0,192,265,0" Height="30" VerticalAlignment="Top" Width="147" Click="outputMessage"/>  <TextBox IsReadOnly="True" Name="statusField2" HorizontalAlignment="Left" Margin="-2,0,-2,-2" TextWrapping="Wrap" VerticalAlignment="Bottom" Height="125" Width="684" Text="" Foreground="Purple" FontSize="14" Padding="5"></TextBox>  </Grid>  </TabItem.Content>  </TabItem>  </TabControl>  </Window> |

ПРИЛОЖЕНИЕ Б

Файл WordAsXML.cs

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using DocumentFormat.OpenXml.Packaging;  using DocumentFormat.OpenXml.Wordprocessing;  using System.Security.Cryptography;  using System.Collections;  using DocumentFormat.OpenXml;  using System.IO;  using Microsoft.Office.Interop.Word;  namespace WordSteganography  {  public class WordAsXML  {  public WordAsXML()  {  }  public string CreateDocCopy(string FilePathStr)  {  string sourceFile = FilePathStr;  string destinationFile;  if (FilePathStr.ToLower().EndsWith(".doc"))  {  destinationFile = FilePathStr.Replace(".doc", "v2.doc");  }  else  {  destinationFile = FilePathStr.Replace(".docx", "v2.docx");  }  try  {  File.Copy(sourceFile, destinationFile, true);  }  catch (IOException iox)  {  Console.WriteLine(iox.Message);  }  return destinationFile;  }  public bool ValidateFile(string FilePathStr)  {  if (FilePathStr.ToLower().EndsWith(".doc"))  return true;  else  return false;  }  public string ConvertDocxToDoc(string FilePathStr)  {  Application word = new Application();  string newFileName = null;  if (FilePathStr.ToLower().EndsWith(".docx"))  {  var sourceFile = new FileInfo(FilePathStr);  try  {  var document = word.Documents.Open(sourceFile.FullName);  newFileName = sourceFile.FullName.Replace(".docx", ".doc");  document.SaveAs2(newFileName, WdSaveFormat.wdFormatDocumentDefault,  CompatibilityMode: WdCompatibilityMode.wdCurrent);  }  catch  {    }  finally  {  word.ActiveDocument.Close();  word.Quit();  File.Delete(FilePathStr);  }  return newFileName;  }  else  {  throw new Exception("Convert .docx to .doc failed");  }  }  public string ConvertDocToDocx(string FilePathStr)  {  Application word = new Application();  string newFileName = null;  if (FilePathStr.ToLower().EndsWith(".doc"))  {  var sourceFile = new FileInfo(FilePathStr);  try  {  var document = word.Documents.Open(sourceFile.FullName);  newFileName = sourceFile.FullName.Replace(".doc", ".docx");  document.SaveAs2(newFileName, WdSaveFormat.wdFormatXMLDocument, ReadOnlyRecommended: false,  CompatibilityMode: WdCompatibilityMode.wdWord2013);  }  catch  {  }  finally  {  word.ActiveDocument.Close();  word.Quit();  }  File.Delete(FilePathStr);  return newFileName;  }  else  {  throw new Exception("Convert .doc to .docx failed");  }  }  public string InsertMessageToFile(string FilePathStr, BitArray message, bool isHashNeed)  {  int counter = 0;  using (WordprocessingDocument doc = WordprocessingDocument.Open(FilePathStr, true))  {    Body mainPart = doc.MainDocumentPart.Document.Body;  var runs = mainPart.Descendants<Run>().ToList();  foreach (Run run in runs)  {  var text = run.GetFirstChild<Text>();  if (text != null)  {  if (text.Text != null & text.Text != " ")  {  string[] words = text.Text.Split(' ');  for (int i = 0; i < words.Count(); i++)  {  string word = words[i];  var newRun = (Run)run.Clone();  string newWord = word;  Text newRunText = newRun.GetFirstChild<Text>();  newRunText.Text = newWord;  run.Parent.InsertBefore(newRun, run);  Run newSpaceRun = new Run();  string space = (i < words.Count() - 1 ? " " : "");  RunProperties runProperties = new RunProperties();  if (space == " ")  {  if (counter < message.Count)  {  if (message.Get(counter))  {  runProperties.AppendChild(new Position() { Val = "2" });  counter++;  }  else  {  runProperties.AppendChild(new Position() { Val = "-2" });  counter++;  }  }  }  runProperties.AppendChild(new DocumentFormat.OpenXml.Wordprocessing.Languages() { Val = "ru-RU" });  newSpaceRun.AppendChild(runProperties);  Text newSpace = newSpaceRun.AppendChild(new Text(space));  newSpace.Space = SpaceProcessingModeValues.Preserve;  run.Parent.InsertBefore(newSpaceRun, run);  }  run.Remove();  }  }  if (counter >= message.Count)  break;  }  doc.Save();  doc.Close();  }  return null;  }  public string GetMessageFromFile(string FilePathStr)  {  string result="";  string lastResult = "";  using (WordprocessingDocument doc = WordprocessingDocument.Open(FilePathStr, true))  {  Body body = doc.MainDocumentPart.Document.Body;  foreach(Run item in body.Descendants<Run>().ToList())  {  if(item.InnerText==" " & item.GetFirstChild<RunProperties>()!=null )  if(item.GetFirstChild<RunProperties>().GetFirstChild<Position>() != null) {  {  if (item.GetFirstChild<RunProperties>().GetFirstChild<Position>().Val == "2")  result += "1";  else if (item.GetFirstChild<RunProperties>().GetFirstChild<Position>().Val == "-2")  {  result += "0";  }  else  {  break;  }  }  }  }  var bitArr = new BitArray(result.Select(c => c == '1').ToArray());  lastResult = ByteArrayToMessage(bitArr);  }  return lastResult;  }  public BitArray MessageToByteArray(string message)  {  byte[] bytes;  bytes = Encoding.UTF8.GetBytes(message);  BitArray bit = new BitArray(bytes);  return bit;  }  public string ByteArrayToMessage(BitArray bites)  {  byte[] strArr = new byte[bites.Length / 8];  for(int i = 0; i < bites.Length / 8; i++)  {  for(int index = i \* 8, m = 1; index < i \* 8 + 8; index++, m \*= 2)  {  strArr[i] += bites.Get(index) ? (byte)m : (byte)0;  }  }  return Encoding.UTF8.GetString(strArr);  }  public int ContainerSize(string FilePathStr)  {  int counter = 0;  string text = null;  using (WordprocessingDocument doc = WordprocessingDocument.Open(FilePathStr, true))  {  var body = doc.MainDocumentPart.Document.Body;  text += body.InnerText;  foreach(var item in text)  {  if (item == ' ')  counter++;  }  }  return counter;  }  public string HashCounter(string message)  {  string result = null;  byte[] byteMessage = Encoding.ASCII.GetBytes(message);  MD5 md5 = new MD5CryptoServiceProvider();  byte[] hashMessage = md5.ComputeHash(byteMessage);  result = hashMessage.Aggregate("", (current, next) => current + next);  return result;  }  }  } |

Файл MainWindow.xaml.cs

|  |
| --- |
| using Microsoft.Win32;  using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Data;  using System.Windows.Documents;  using System.Windows.Input;  using System.Windows.Media;  using System.Windows.Media.Imaging;  using System.Windows.Shapes;  namespace WordSteganography  {  /// <summary>  /// Логика взаимодействия для MainWindow.xaml  /// </summary>  public partial class MainWindow : Window  {  public static bool isFileDoc { get; set; }  public static string FilePathStr { get; set; }  public static string encodedFilePathStr { get; set; }  public static string CopyPathStr { get; set; }  public static int ContainerSize { get; set; }  public static int MessageSize { get; set; }  public static int BitHashSize { get; set; }  public static string Hash { get; set; }  public static BitArray BitHash { get; set; }  public static BitArray BitMessage { get; set; }  public static string MessageAndHash { get; set; }  public static BitArray BitMessageAndHash { get; set; }  public MainWindow()  {  InitializeComponent();  }  private void openInputFile(object sender, RoutedEventArgs e)  {  try  {  OpenFileDialog FileDialog = new OpenFileDialog(); // open file from a directory  FileDialog.Filter = "Word File (.docx ,.doc)|\*.docx;\*.doc";  if (FileDialog.ShowDialog() == true)  FilePathStr = FileDialog.FileName;  FilePath.Text = FilePathStr;  WordAsXML helper = new WordAsXML(); // create a work copy and validate .doc/.docx  if (isCopyNeed.IsChecked == true)  {  FilePathStr = helper.CreateDocCopy(FilePathStr);  }  isFileDoc = helper.ValidateFile(FilePathStr);  if (isFileDoc)  {  FilePathStr = helper.ConvertDocToDocx(FilePathStr);  }  ContainerSize = helper.ContainerSize(FilePathStr); // count file spaces and print the result  containerSizeField.Text = ContainerSize.ToString();  }  catch (Exception ex)  {  statusField1.Text += ex.Message + "\n";  statusField2.Text += ex.Message + "\n";  }  }  private void inputMessage(object sender, RoutedEventArgs e)  {  try  {  statusField1.Text = "";  statusField2.Text = "";  if (FilePathStr == null)  {  throw new Exception("Файл не выбран :(");  }  WordAsXML helper = new WordAsXML();  if (isHashNeed.IsChecked == true)  {  Hash = helper.HashCounter(messageField.Text);  hashField.Text = Hash;  MessageAndHash = messageField.Text + " " + Hash;  }  else  {  MessageAndHash = messageField.Text + " " + null;  }  BitMessageAndHash = helper.MessageToByteArray(MessageAndHash);  if (ContainerSize >= BitMessageAndHash.Count)  {  if (isHashNeed.IsChecked == true)  helper.InsertMessageToFile(FilePathStr, BitMessageAndHash, true);  else  helper.InsertMessageToFile(FilePathStr, BitMessageAndHash, false);  if (isFileDoc)  {  helper.ConvertDocxToDoc(FilePathStr);  isFileDoc = false;  }  statusField1.Text = "Сообщение осаждено!\n";  statusField2.Text = "Сообщение осаждено!\n";  }  else  {  throw new Exception("В контейнере недостаточно места :(");  }  }  catch (Exception ex)  {  statusField1.Text += ex.Message + "\n";  statusField2.Text += ex.Message + "\n";  }  }  private void openOutputFile(object sender, RoutedEventArgs e)  {  try  {  OpenFileDialog FileDialog = new OpenFileDialog();  FileDialog.Filter = "Word File (.docx ,.doc)|\*.docx;\*.doc";  if (FileDialog.ShowDialog() == true)  encodedFilePathStr = FileDialog.FileName;  outputFilePath.Text = encodedFilePathStr;  }  catch (Exception ex)  {  statusField1.Text += ex.Message + "\n";  statusField2.Text += ex.Message + "\n";  }  }  private void outputMessage(object sender, RoutedEventArgs e)  {  try  {  if (encodedFilePathStr == null)  {  throw new Exception("Файл не выбран");  }  WordAsXML helper = new WordAsXML();  string result = helper.GetMessageFromFile(encodedFilePathStr);  if (result != null)  {  var lastSpacePos = result.LastIndexOf(' ');  string message = result.Substring(0, lastSpacePos);  string hash = result.Substring(lastSpacePos + 1, result.Length - lastSpacePos - 1);  outputMessageField.Text = message;  if (hash != "")  {  outputHashField.Text = hash;  controlHashField.Text = helper.HashCounter(message);  statusField1.Text += "Сообщение извлечено!" + "\n";  statusField2.Text += "Сообщение извлечено!" + "\n";  }  else  {  outputHashField.Text = "Хеш не был извлечен/осажден :(";  controlHashField.Text = helper.HashCounter(message);  }  }  else  {  throw new Exception("Сообщение не было извлечено");  }  }  catch (Exception ex)  {  statusField1.Text += ex.Message + "\n";  statusField2.Text += ex.Message + "\n";  }  }  private void IsHashNeed\_Checked(object sender, RoutedEventArgs e)  {  hashField.IsEnabled = true;  }  private void IsHashNeed\_Unchecked(object sender, RoutedEventArgs e)  {  hashField.IsEnabled = false;  }  }  } |