Esteganografía en archivos digitales

Digital File Steganography

Trabajo Fin de Grado

Curso 20-21

Autor

Alberto Germán Arias Del Cerro

Director

Enrique Martín Martín

Grado en Ingeniería de Software

Facultad de Informática

Universidad Complutense de Madrid

Esteganografía en archivos digitales
Digital File Steganography

Trabajo de Fin de Grado en Ingeniería de Software

Autor

Alberto Germán Arias Del Cerro

Director

Enrique Martín Martín

Convocatoria: S*eptiembre 2021*

Grado en Ingeniería de Software

Facultad de Informática

Universidad Complutense de Madrid

Agradecimientos

En primer lugar, a mi familia y en especial a mis padres por las duras circunstancias que hemos tenido que vivir antes de llegar a este momento. A los compañeros de universidad y de trabajo con los que he coincidido estos años, porque conocerlos me ha ayudado a transformarme en la persona que soy en diferentes aspectos. A los profesores que me han dado clase durante mi estancia en la universidad, porque gracias a ellos he podido aprender sobre esta bella profesión, en particular a Enrique Martín mi tutor en este proyecto por ayudarme a finalizar este camino y a Antonio Navarro que su método docente y su forma de ver esta profesión me han servido para comprender y afrontar correctamente el mundo laboral de la ingeniería de software.

Resumen

Se ha desarrollado una aplicación que permite elegir entre diferentes métodos esteganográficos basados en la técnica de la modificación del bit menos significativo (LSB, en sus siglas en inglés) para codificar o decodificar un mensaje de texto en una imagen. Los métodos que se pueden utilizar son: LSB simple con una variante de la codificación Elias delta, diferencia de valor de píxel (PVD o pixel value differencing, en sus siglas en inglés), explotación de la modificación de la dirección (EMD o exploiting modification direction, en sus siglas en inglés) y LSB+PVD+EMD que mezcla los tres anteriores. De cada uno de estos métodos se ha realizado un estegoanálisis por par de valores (PoV o *Pairs of values*, en sus siglas en inglés) para medir la capacidad que tienen de ocultarse ante análisis estadísticos cada uno de los métodos. El lenguaje de programación utilizado para el desarrollo ha sido Python y el código de la aplicación está en el siguiente enlace:

https://github.com/agarias/dfs/

Palabras clave

esteganografía, estegoanálisis, PVD, LSB, EMD, codificación Elias delta, Python, PIL, unittest, estegoanálisis por par de valores

Abstract

The project consists of an application that allows choosing among different steganographic methods based on the least significant bit (LSB) modification technique to encode or decode a text message in an image. The methods that can be used are: Simple LSB with an Elias delta encoding variant, Pixel Value Difference (PVD), exploiting modification direction (EMD), and the combination of the previous three (LSB + PVD + EMD). Each of the methods has been subjected to a steganalysis by pairs of values (PoV), which measures the ability of each of the techniques to hide from statistical analysis. The application has been developed in Python and the code can be found at the following link:

https://github.com/agarias/dfs/

Keywords

steganography, steganalysis, PVD, LSB, EMD, Elias delta encoding, Python, PIL, unittest, PoV

Índice

Índice de figuras

Índice de tablas

1. Introducción

Se ha realizado un proyecto sobre la esteganografía en archivos digitales, como veremos es un campo muy extenso y el desarrollo del trabajo se ha centrado en algunas técnicas de la esteganografía en imágenes. En esta sección se exponen los objetivos que se han marcado, el plan de trabajo llevado a cabo y el contenido de la memoria.

1.1. Objetivos

El primer objetivo de este proyecto es realizar una investigación sobre la esteganografía, el estegoanálisis y sus diferentes ramas, la búsqueda de las técnicas esteganográficas que se han implementado, de sistemas o herramientas similares y de métodos de estegoanálisis para valorar y comparar las distintas técnicas.

El objetivo principal es desarrollar una biblioteca en Python de técnicas esteganográficas, para la codificación y decodificación de mensajes de texto en imágenes digitales. En concreto técnicas basadas en la modificación de los bits menos significativos de los píxeles de una imagen. Se han desarrollado las siguientes técnicas:

- **LSB**: Modificación del bit menos significativo, codificando los bits del mensaje con una variante de la codificación Elias delta.

- **PVD**: Diferencia de valor de píxel, está técnica utiliza la diferencia entre un píxel y sus adyacentes para codificar más o menos bits, existen distintas variantes de esta técnica en la literatura.

- **EMD**: Explotación de la modificación de la dirección, se utilizan las modificaciones de píxeles en diferentes direcciones para representar diferentes datos secretos, existen distintas variantes de esta técnica en la literatura.

- **LSB+PVD+EMD**: Es una técnica que combina las tres anteriores, busca diferenciar entre áreas de borde y áreas lisas de una imagen para utilizar la técnica más adecuada, pretendiendo así obtener mayor capacidad de ocultación frente a un análisis de detección y una mayor proporción de información ante las distorsiones de la imagen.

1.2. Plan de Trabajo

A finales de octubre de 2020 se realizó la planificación del trabajo y se decidió utilizar Python para realizar la implementación. Primero se realizó la investigación sobre la esteganografía y las diferentes técnicas esteganográficas decidiendo cuáles se iban a implementar. Seguidamente, se procedió a la implementación de una biblioteca con las técnicas escogidas donde se codifica un mensaje de texto dentro de una imagen. Como parte de la implementación también se desarrollaron test unitarios sobre cada uno de los métodos probando los distintos casos de uso relevantes. El siguiente paso fue investigar sobre el estegoanálisis para poder realizar una valoración y comparativa de las técnicas desarrolladas. Simultáneamente se buscaron herramientas de estegoanálisis externas que pudieran dar una indicación más objetiva sobre los métodos implementados y junto a otras herramientas existentes de esteganografía para otorgar un contexto a nuestra aplicación. Por último, se redactó la memoria del proyecto exponiendo todo lo comentado anteriormente.

Teniendo en cuenta los parones necesarios por los periodos de exámenes el plan de trabajo ha sido el siguiente:

- **Planificación del trabajo** del 22 de octubre de 2020 al 29 de octubre de 2020.

- **Investigación sobre esteganografía y técnicas** del 2 de noviembre de 2020 al 22 de diciembre de 2020, en este período se decidió que se iba a desarrollar LSB, PVD, EMD y LSB+PVD+EMD.

- **Implementación de las técnicas esteganográficas:** del 26 de diciembre al 15 de abril con parón por periodo de exámenes del 5 de enero al 15 de febrero.
    - Implementación del LSB del 26 de diciembre de 2020 al 4 de enero de 2021.
    - Implementación del PVD del 16 de febrero de 2021 al 2 de marzo de 2021.
    - Implementación del EMD del 3 de marzo de 2021 al 15 de marzo de 2021.
    - Implementación de LSB-PVD-EMD del 17 de marzo de 2021 al 22 de marzo de 2021.
    - Diseño y desarrollo de los test de la aplicación del 24 de marzo de 2021 al 12 de abril de 2021.

- **Investigación sobre estegoanálisis y sobre herramientas de estegoanálisis y esteganografía:** del 15 de abril de 2021 al 21 de julio de 2021, con parón por periodos de exámenes del 1 de mayo de 2021 al 15 de julio de 2021.
    - Investigación sobre el estegoanálisis del 15 de abril de 2021 al 30 de abril de 2021.
    - Investigación sobre herramientas y pruebas sobre las mismas del 16 de julio de 2021 al 21 de julio de 2021.

- **Memoria y documentación** del 22 de julio de 2021 al 20 de septiembre de 2021.

1.3. Contenido de la memoria

El documento contine una sección de **preliminares** donde se expone la investigación hecha sobre la esteganografía y el estegoanálisis. A continuación, se explican las **técnicas implementadas** donde exponemos los trabajos teóricos en los que se han basado y la implementación que se ha realizado, con las adaptaciones llevadas a cabo. Seguidamente, una sección donde se muestra el **estegoanálisis** al que se han sometido nuestras técnicas **y los resultados del desarrollo** frente a pruebas de esfuerzo. Finalmente se exponen las **conclusiones** donde tratamos como hemos alcanzado los objetivos, algunas observaciones subjetivas que se han hecho durante todo el proceso de desarrollo y apuntes sobre cómo se puede continuar este trabajo.

2. Introduction

A project has been developed on steganography in digital files, that as we will see, is a very extensive field. The work has focused on steganography in images. This section explains the objectives that have been set, the work plan implemented and the content of the report.

2.1. Objetives

The first objective of this project is to investigate on steganography, steganalysis and its different branches, search for steganographic techniques that have been implemented and similar systems or tools, and identify steganalysis methods in order to evaluate and compare the different techniques.

The second objective is to develop a Python library of steganographic techniques for the encoding and decoding of text messages in digital images. Specifically, techniques based on the modification of the least significant bits of the pixels of images. The following techniques will be developed:

- Modification of the least significant bit **(LSB)**: encoding the message bits with a variant of the Elias delta encoding.

- Pixel value difference **(PVD):** using the difference between a pixel and its adjacent to encode more or fewer bits. There are different variants of this technique in the literature.

- Exploiting modification direction **(EMD):** using pixel modifications in different directions to represent different secret data. There are different variants of this technique in the literature.

- **LSB+PVD+EMD:** combining the previous three, it aims to differentiate between edge and smooth areas of an image to identify the right technique. Therefore, we obtain a higher hide capacity against a detection analysis and a higher level information when we face image distortions.

2.2. Workplan

At the end of October 2020, the planning of the work was done, and it was decided to use Python to develop the implementation. Firstly, we conducted research on steganography and decided the techniques to be implemented. Next, we proceeded to implement a library with the chosen techniques to encode a text message within an image. As part of the implementation, unit tests were developed on each of the methods, testing the several relevant use cases. In the next step, that we conducted the research on the steganalysis to assess and compare of the implemented techniques. We also looked for external steganalysis tools to obtain a more objective report of the techniques, and steganography tools to put our application in context. Finally, we made the project report presenting all the different steps taken so far.

Considering the necessary stops for the exam periods, this is the work plan followed:

- **Work planning** from October 22nd, 2020, to October 29th, 2020.

- **Research on steganography and techniques** from November 2nd, 2020, to December 22nd, 2020, in this period it was decided that LSB, PVD, EMD and LSB + PVD + EMD would be developed**.**

- **Implementation of steganographic techniques:** from December 26th to April 15th with a break for exam period from January 5th to February 15th.

    - Implementation of the LSB from December 26th, 2020, to January 4th, 2021.
    - Implementation of the PVD method from February 16th, 2021, to March 2nd, 2021.
    - Implementation of the EMD method from March 3rd, 2021, to March 15th, 2021.
    - Implementation of the LSB-PVD-EMD method from March 17th, 2021, to March 22nd, 2021.
    - Design and development of the application tests from March 24th, 2021, to April 12th, 2021.

- **Research on steganalysis and on steganalysis and steganography tools:** from April 15th, 2021, to July 21st, 2021, with a break for exam periods from May 1st, 2021, to July 15th, 2021.

    - Research on steganalysis from April 15th, 2021, to April 30th, 2021.
    - Research on tools and tests on them from July 16th, 2021, to July 21st, 2021.

- **Report and documentation** from July 22nd, 2021, to September 20th, 2021.

2.3. Content

The document contains a preliminary section where the research about steganography and steganalysis will be presented. Next, the implemented techniques are explained where the theoretical work on which they have been based and the implementation and the pertinent adaptations, are exposed. Afterwards, a section where the steganalysis and the stress tests to which our techniques have been subjected is shown. Finally, we present the conclusions where the achievement of objectives, some subjective observations throughout the development process and some notes on how this project can continue described in detail.

3. Preliminares

Para exponer adecuadamente el trabajo desarrollado primero se deben explicar algunos conceptos esenciales como la esteganografía y el estegoanálisis. En esta sección se detalla en qué consisten, se realiza un repaso por la historia de la esteganografía, se explican las características de las técnicas esteganográficas y las diferentes clasificaciones que se pueden hacer tanto de la esteganografía como del estegoanálisis, se dedica un apartado a la esteganografía en imágenes (hablando de algunas técnicas concretas) y otro al estegoanálisis por par de valores, se especifican algunas de las aplicaciones actuales de la esteganografía (mencionando algunos estudios de sus aplicaciones potenciales) y finalmente se comentan algunas herramientas existentes de esteganografía y estegoanálisis.

3.1. Esteganografía

Para explicar qué es la esteganografía vamos a empezar hablando de la etimología de la palabra, viene del griego de las palabras *steganos*, que significa oculto o cubierto y *graphia* que significa escritura, por lo que podemos concluir que la palabra significa "escritura oculta", muy adecuado ya que la esteganografía trata el estudio y la aplicación de técnicas que permiten ocultar mensajes u objetos dentro de otros de modo que no se pueda percibir su existencia, y así establecer un canal de comunicación oculto que los observadores que tengan acceso a él puedan pasar por alto. Se pueden confundir esteganografía y criptografía, que etimológicamente tienen un significado similar ya que esta última significaría "escritura secreta", pero la criptografía consiste en hacer ininteligible un mensaje, archivo, … (lo que se conoce como cifrar) y que solo pueda ser leído si se tiene una clave.

El término esteganografía fue utilizado por primera vez por Johannes Trithemius (1462-1516) en el libro con ese mismo nombre publicado en el año 1500, dónde codifico varios mensajes que hasta 1996 no fueron decodificados. Pero la primera evidencia escrita sobre el uso de la esteganografía para enviar mensajes es de mucho antes, se debe a Heródoto de Halicarnaso, historiador y geógrafo griego, que en su obra "Las historias" (en torno al 430 a. C) relata varias historias donde se esconden mensajes en objetos o incluso en personas. Estos y otros antecedentes históricos han sido recogidos en [1, 2] .

En la época actual, la esteganografía se ha vuelto digital. Se busca modificar archivos digitales para ocultar información dentro de ellos, a través de métodos y técnicas computacionales, para que aquellos para los que no va destinada no puedan acceder a ella. Una de las primeras proposiciones del uso de esteganografía digital se da en el conocido como problema de los prisioneros [3], del criptógrafo Gustavus Simmons,

donde con una metáfora de cómo dos prisioneros deben pasarse mensajes para escapar de prisión, pero que deben hacerlo a través de un guardia que no debe enterarse del contenido de estos, ni detectar que pueda haber mensajes ocultos por que podría dejar de enviar los mensajes, explica cómo la esteganografía puede ayudar a establecer un canal encubierto. También habla de tres tipos de guardianes según el poder de acción de que tengan sobre el canal: pasivo (tendría acceso de solo lectura al archivo), activo (podría modificar el archivo provocando que no se pudiera leer la información oculta) o malicioso (podría modificar la información oculta del archivo engañando a quien vaya dirigido). Estos tres guardianes se identifican con los posibles tipos de ataques que nuestros archivos con información oculta pueden sufrir, esto ayudará a definir las características que debe tener una técnica esteganográfica y los distintos tipos de estegoanálisis.

Uno de los primeros métodos para discutir la esteganografía digital se les atribuye a Kurak y McHugh [4] quienes propusieron un método de incrustación en los 4 bits menos significativos, examinaron la degradación y la contaminación de las imágenes y es el comienzo de lo que ahora se conoce como la esteganografía basada en imágenes, lo que se ha aplicado en este trabajo. En la figura 1 se muestra un diagrama con las diferentes disciplinas de ocultación de información digital que existen actualmente según Cheddad *et al.* [2].

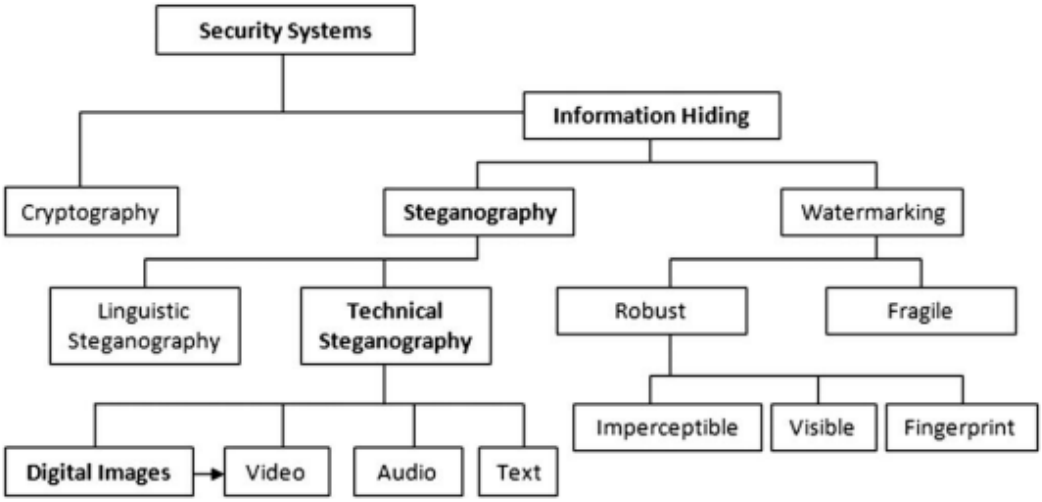

Figura 1 Clasificación de las técnicas de ocultación digitales (fuente [2])

Hemos comentado las diferencias entre la criptografía y la esteganografía, pero como aparece en la figura 1 también existen las marcas de agua (*Watermarking* en inglés) como un tipo de técnica de ocultación. A nivel técnico se puede afirmar que es una forma de esteganografía que utiliza técnicas digitales de enmascaramiento y filtrado para ocultar información dentro de un archivo. Pero el objetivo es distinto ya que en la esteganografía tradicional se busca una comunicación encubierta, y en las marcas de agua se busca añadir información adicional al archivo original. Se suelen usar para añadir a imágenes o documentos derechos de autor, propiedad, licencias o firmas.

3.1.1. Características de una técnica esteganográfica

Una técnica esteganográfica se compone de un método de embebido, incrustación, ocultación o codificación donde se ocultará la información dentro del archivo original, al que también llamaremos archivo portador. También de un método de extracción o decodificación donde se obtendrá la información oculta en un archivo al que llamaremos estego-archivo.

Según los objetivos que se persigan con la inclusión de la información en un archivo digital, en una técnica esteganográfica se dará más valor a unas u otras características. Antes de diferenciar entre tipos de técnicas se explicará cuáles son las características que deben presentar según Al-Ani *et al.* [5]:

- **Capacidad**: Es la cantidad de información oculta que se puede esconder por cantidad de información que hay en el archivo portador. En archivos de video y de audio también se puede utilizar la información oculta por tiempo de reproducción, dependiendo de lo que se necesite medir.

- **Robustez**: Es la capacidad de resistencia que tiene una técnica ante la manipulación de un estego-archivo, la cual no busca la información oculta. Por ejemplo, el proceso de compresión de un archivo, aplicación de filtros a una imagen o la eliminación del ruido en un archivo de audio.

- **Indetectabilidad o invisibilidad estadística**: Es la capacidad de que la información oculta no sea detectada ante análisis estadísticos o de que no haya desviaciones estadísticas que permitieran poder decir que existe información oculta.

- **Invisibilidad o transparencia perceptiva**: Es la capacidad de que la información oculta no sea detectada a los sentidos humanos. Esta cualidad es casi indispensable en cualquier técnica de esteganografía. Si el estego-archivo queda tan distorsionado que permite detectar que hay ocultación tampoco se cumpliría esta característica.

- **Seguridad**: Es la capacidad de resistencia que tiene una técnica al intento de extracción o manipulación de la información oculta de un atacante que desconozca el método de extracción.

3.1.2. Clasificación de las técnicas esteganográficas

Existen diversos tipos de clasificación de las técnicas esteganográficas en función de diferentes factores:

- Como se ha mostrado en la figura 1, una de las clasificaciones más simples de las técnicas esteganográficas es según el tipo de archivo portador (definido en la sección 3.1.1), que llamaremos para simplificar **por tipo de portador**.

- Otra de las clasificaciones que se llevan haciendo desde más tiempo es la llamada **por tipo de esteganografía**, serían pura, de clave privada y de clave pública. Ya en el problema de los prisioneros [3], antes mencionado, se define la esteganografía pura y de clave secreta como posibles soluciones al problema. Y a lo largo de los años la esteganografía de clave secreta se ha acabado dividiendo en de clave privada y de clave pública.

- Está la clasificación más utiliza, que es según la manera en la que se oculta la información en el archivo portador y que llamaremos para simplificar **por el tipo de modificación.**

3.1.2.1. Clasificación por tipo de portador

La clasificación por tipo de portador es básicamente por el tipo de archivo, es decir técnicas para documentos de texto, audio, video o imagen.

- **Documentos de texto**: Se suelen usar técnicas que utilizan espacios en blanco o tabulaciones ya que en la mayoría de los editores de texto no son detectables al ojo humano.

- **Audio**: Una de las técnicas en audio con mayor eficacia es el espectro ensanchado, en la que se añaden ruidos a la señal donde se oculta la información. También son muy útiles las técnicas basadas en ecos, donde se oculta la información añadiendo sonido extra a los ecos de un archivo, esto además suele provocar que se mejore la calidad del audio por lo que levanta aún menos sospechas.

- **Imágenes:** En general las técnicas usadas en imágenes suelen basarse en la modificación del bit menos significativo (LSB), la mayor desventaja del método es el hecho de que sea el más usado y conocido, y ya hay muchos análisis sobre las marcas que dejan en las imágenes este tipo de técnicas y muchas de las diferencias entre unas y otras es como evitan estos análisis con la dispersión del mensaje. En las imágenes de mayor calidad, con mayor resolución, es más fácil ocultar las huellas y por lo tanto ocultar la información de forma óptima. Pero por el contrario son técnicas poco robustas ya que un simple cambio de formato o una compresión suelen dañar inevitablemente el contenido de la información oculta y que resulte irrecuperable.

- **Video**: Una técnica muy usada es la modificación de los coeficientes de la DCT (la transformada de coseno discreta) que modifica cada una de las imágenes del vídeo, sin que pueda ser perceptible para el ojo humano.

3.1.2.2. Clasificación por tipo de esteganografía

Como ya se ha mencionado esta clasificación deriva del problema del prisionero [3] y se diferenciaran tres categorías esteganografía pura, esteganografía de clave privada y esteganografía de clave pública, aunque estas dos últimas se engloban dentro la esteganografía de clave secreta.

- **Esteganografía pura**: Estas técnicas utilizarán algoritmos que establecen una manera fija de incorporar la información en el archivo portador para obtener el estego-archivo. Las técnicas que se han implementado en este trabajo formarían parte de esta categoría.

- **Esteganografía de clave secreta**: Estas técnicas utilizarán algoritmos que estarán parametrizados por una clave secreta que defina como aplicar el algoritmo. Emisor y receptor deben establecer previamente tanto el algoritmo como la clave, el método de extracción consiste en aplicar el algoritmo y la clave secreta necesarios en el estego-archivo recibido para obtener la información oculta.

    - **Esteganografía de clave privada**: La información se oculta usando una clave privada que proporciona más seguridad la cual no se transmite ya que previamente deben conocerla tanto emisor como receptor. Un ejemplo claro de este tipo de serían los intercambios de clave DH o ECDH que se utilizan en el protocolo TLS de la capa de transporte

    - **Esteganografía de clave pública:** Se requiere una clave privada y una pública, la pública se usa para ocultar la información y la privada para recuperarla. En este caso es necesario transmitir la clave a través del canal. Un ejemplo claro de este tipo de serían los intercambios de clave RSA que se utilizan en el protocolo TLS de la capa de transporte, aunque ya este tipo de intercambio se está dejando de utilizar.

3.1.2.3. Clasificación por el tipo de modificación

En la clasificación de las técnicas esteganográficas por el tipo de modificación que se realiza en el archivo portador, tendremos seis categorías, de acuerdo con Al-Ani *et al.* [5]. Aunque según esta clasificación algunas técnicas existentes pueden no encajar exactamente en una categoría o pueden encajar en varias de ellas.

- **Técnicas de sustitución:** Estas técnicas intentan usar las partes menos importantes del archivo portador por bits con la información a ocultar. El receptor puede obtener la información si conoce en que partes se han modificado, y como estas modificaciones son muy pequeñas en comparación con el tamaño del archivo, el emisor no espera que sea detectado por un atacante. Las técnicas de modificación del LSB mencionadas en la sección 3.1.2.1 entrarían en esta categoría. Estas técnicas son simples de desarrollar y pueden tener una gran

capacidad además de tener una buena invisibilidad perceptiva, sin embargo, son poco robustas, muy susceptibles a la modificación del estego-archivo y tienen una baja indetectabilidad ya que tienden a dejar muchas marcas de su aplicación a nivel estadístico.

- **Técnicas de transformación de dominios:** Estas técnicas ocultan el mensaje en un área significativa del archivo portador, lo que las hace más robustas, pero se busca que sigan teniendo una alta invisibilidad, por lo tanto, lo que se busca es distribuir la información oculta a través de diferentes bits de orden. Una de las técnicas que formarían en esta categoría sería la mencionada en la sección 3.1.2.1 en las técnicas de video, la modificación de los coeficientes de la DCT. En la actualidad estos son los tipos de técnicas más robustos.

- **Técnicas de espectro ensanchado:** Estas técnicas usan medios de transmisión en los que la señal ocupa un ancho de banda superior al mínimo necesario para enviar la información. El ensanchamiento de banda se logra mediante un código que es independiente de los datos, con una recepción sincronizada usando el código en el receptor para la difusión y posterior recuperación de los datos. Dado que las señales de dispersión tienden a ser difíciles de eliminar, las técnicas de espectro ensanchando obtienen un nivel considerable de robustez. En el ocultamiento de información, generalmente se utilizan dos variantes especiales de este tipo de técnicas: secuencia directa y esquema de salto de frecuencia. En el esquema de secuencia directa, la señal secreta se propaga mediante una constante llamada tasa de envío, modulada con una señal pseudoaleatoria y agregada a la cobertura. En los esquemas de salto de frecuencia, la frecuencia de la señal portadora se altera de manera que salta rápidamente de una frecuencia a otra. Estas técnicas se utilizan ampliamente en el contexto de las marcas de agua.

- **Técnicas de distorsión**: Estas técnicas, a diferencia de las anteriores, requieren el conocimiento de cómo es el archivo portador para poder realizar la decodificación de la información oculta. El emisor aplica una secuencia de modificaciones en el archivo portador para obtener un estego-archivo. Se elige una secuencia de modificación de tal de manera que corresponda a un mensaje secreto específico a transmitir. El receptor mide la diferencia en el archivo portador para reconstruir la secuencia de modificación, que corresponde al mensaje secreto. La mayoría de los métodos de ocultación basados en texto son del tipo de distorsión (es decir, la disposición de las palabras o el diseño de un documento pueden revelar información). Los archivos HTML son buenos candidatos para incluir espacios, pestañas y saltos de línea adicionales. Los navegadores web ignoran estos espacios y líneas "extra" y pasan desapercibidos hasta que se revela el origen de la página web.

- **Técnicas de generación de portadores:** Estas técnicas al contrario que las anteriores en lugar de modificar un archivo portador y generar un estego-archivo con la información a ocultar, a partir de la información a ocultar generan un archivo portador nuevo que contiene la información oculta. Así que el único propósito de estos archivos es tapar la comunicación secreta.

- **Técnicas de esteganografía estadística:** Estas técnicas utilizan la existencia de esquemas de esteganografía de “bits a 1”. Funcionan modificando el archivo portador de tal manera que algunas características estadísticas cambien significativamente si se transmite un "1”, de lo contrario el portador no es modificado. Un archivo portador se divide en m bloques separados, desde B_1 hasta B_m . Un bit oculto, m_i, se inserta en el bloque i al colocar "1" en B_i si m_i = 1. De lo contrario, el bloque no se cambia en el proceso de incorporación.

3.1.3. Esteganografía en imágenes digitales

Vamos a dar una visión general de las técnicas esteganográficas más importantes en imágenes digitales. Los formatos de imagen más populares son GIF, JPEG y PNG y la mayoría de las técnicas desarrolladas se establecieron para explotar las estructuras de estos formatos, con algunas excepciones que utilizan el formato de mapa de bits (BMP) por su estructura de datos simple.

Definimos el método de incrustación según la figura 2 : Sea C la imagen portadora y C’ la estego-imagen. Sea K una clave opcional (utilizada para cifrar el mensaje o para generar un ruido pseudoaleatorio) y sea M el mensaje que queremos comunicar. Em es un acrónimo de embebido y Ex de extracción. Por lo tanto:

$$Em: C \oplus K \oplus M \rightarrow C' \ \therefore Ex: (Em(c,k,m) \approx m, \forall c \in C, k \in K, m \in M$$

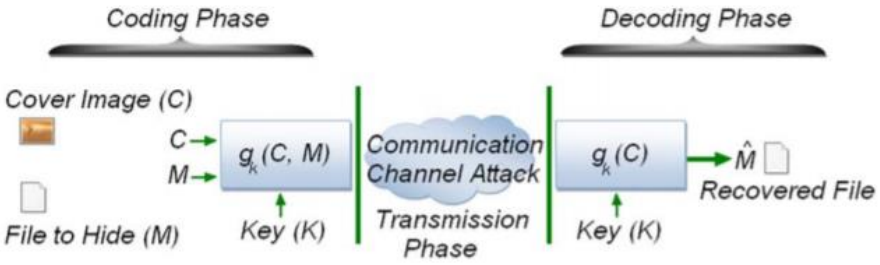

Figura 2 Ilustración del método de incrustación genérico: C denota imagen portadora, M denota los datos a ocultar (fuente [2])

Dividiremos la esteganografía en imágenes en cuatro categorías, esteganografía que aprovecha los formatos de la imagen, esteganografía en el dominio espacial (que generalmente utilizan una técnica de reemplazo directo de bit menos significativo (LSB)), esteganografía en el dominio de la frecuencia y esteganografía adaptativa. Esta categorización es la realizado por Cheddad *et al.* [2] y también aseguran que no es una clasificación estándar. Por ejemplo, los métodos adaptativos pueden aplicarse en los dominios espacial o de frecuencia; por lo tanto, se pueden considerar casos especiales de estos.

3.1.3.1. Esteganografía aprovechando el formato

Se puede lograr ocultar información en una imagen digital de maneras muy simples, como ejecutar un comando del sistema operativo que copie el mensaje de un fichero de texto en el archivo de la imagen, en los sistemas Windows sería un comando de este estilo: **C> Copy Cover.jpg /b + Message.txt /b Stego.jpg.** La idea detrás de esto es aprovechar el reconocimiento de fin de archivo (EOF), insertando el mensaje después de la etiqueta EOF. Cuando se visualiza Stego.jpg usando cualquier aplicación de edición de fotos simplemente mostrará la imagen ignorando cualquier cosa que venga después de la etiqueta EOF. Sin embargo, cuando se abre en el bloc de notas, por ejemplo, nuestro mensaje se revela después de mostrar algunos datos. Desafortunadamente, esta sencilla técnica no resistiría ningún tipo de edición de la estego-imagen ni ningún estegoanálisis.

Otra implementación simple y fácil de detectar, es agregar datos ocultos en la información de archivo extendido de la imagen (EXIF), que es un estándar utilizado por los fabricantes de cámaras digitales para almacenar información en el archivo de imagen. Esta es la información de metadatos sobre la imagen y se ubica en el encabezado del archivo. El agente especial Paul Alvarez [6] propuso utilizar tales encabezados en el análisis de evidencia digital para combatir la pornografía infantil. Este método no es confiable ya que adolece de los mismos inconvenientes que el método EOF, por eso si debemos usar este tipo de técnicas es aconsejable cifrar los datos que vamos a ocultar para que sea más difícil que un atacante los descifre.

3.1.3.2. Esteganografía en el dominio espacial

En estas técnicas, se modifican los datos secretos y el medio de cobertura en el dominio espacial, lo que implica la codificación al nivel de los LSB. Aunque estas técnicas puedan resultar más simples, tienen un impacto mayor en comparación con las técnicas de las otras categorías.

Uno de los métodos más usados en las técnicas de domino espacial es modificar el mapa de bits, cambiando el valor de los que menos afectan la apariencia de la imagen. Los colores que utilizan una imagen también se ven representados por la cantidad de bits que disponga, esto significa que si la imagen posee 3 bytes existen un color rojo, uno verde y uno azul, lo que al combinarse pueden dar cualquier color, formando una paleta de colores. La esteganografía basada en la paleta de colores explota la transición suave de la rampa en los colores, lo que hace que las modificaciones no sean perceptibles. Los LSB aquí se modifican en función de sus posiciones en el índice de la paleta.

Johnson y Jajodia [7] estaban a favor de utilizar BMP o GIF en lugar de imágenes JPEG, a pesar de que su resistencia a los contraataques estadísticos es débil, debido al algoritmo de compresión JPEG que no admite una incrustación LSB directa en el

dominio espacial. La ocultación de datos basada en histogramas es otro esquema de uso común. Li et al. [8] proponen la ocultación de datos sin pérdidas utilizando el valor de diferencia de los píxeles adyacentes, se clasifica en algoritmos de incrustación de datos "±1". Aprovecha la correlación entre píxeles adyacentes que eventualmente da como resultado un histograma compacto que se caracteriza por una distribución gaussiana normal. En lugar de considerar la imagen completa, Tsai et al. [9] dividían la imagen en bloques de 5 x 5 donde la imagen residual se calcula mediante predicción lineal (otro término para la diferencia de píxeles adyacentes). Tales esquemas tienen la ventaja de recuperar la imagen portadora original de la estego-imagen. Si bien esta conservación puede ser necesaria en determinadas aplicaciones, como la formación de imágenes médicas, en general, la esteganografía no se ocupa de dicha recuperación. Potdar et al. [10] utilizaron una técnica de dominio espacial para soportar los ataques de recorte de imágenes. La lógica detrás de su trabajo es dividir la imagen portadora en subimágenes y comprimir y cifrar la información a ocultar. Esta información resultante se subdivide a su vez y se integran en esas partes de la imagen. Para recuperar los datos, se aplica un polinomio de interpolación de Lagrange junto con un algoritmo de cifrado. El coste computacional de esta técnica es alto, y el número de subimágenes (n) y el valor umbral (k) no se establecieron en valores óptimos, dejando al lector adivinar los valores. Si establecemos n en 32, por ejemplo, eso significa que se necesitan 32 claves públicas junto con 32 personas y 32 subimágenes, lo que resulta poco práctico. Además, la redundancia de datos que pretendían eliminar se produce en su estego-imagen.

3.1.3.3. Esteganografía en el dominio de la frecuencia

La falta de robustez y seguridad (3.1.1) de las técnicas LSB en dominio espacial, provocó que los investigadores se preguntarán dónde podían aplicarlo y empezarán a desarrollar las técnicas dentro del dominio de la frecuencia.

Los cambios pequeños en el LSB de un píxel en una imagen JPEG pueden detectarse de forma fiable. Por eso para imágenes JPEG es mejor usar técnicas en el dominio de la frecuencia. Los experimentos sobre los coeficientes de transformada de coseno discreta (DCT) mostraron resultados prometedores y redirigieron la atención de los investigadores hacia este tipo de imagen. De hecho, actuar al nivel de DCT hace que la esteganografía sea más robusta y menos propensa a ataques estadísticos.

La DCT [11] es una transformada basada en la transformada de Fourier discreta, pero utilizando únicamente números reales, expresa una secuencia finita de varios puntos como resultado de la suma de distintas señales sinusoidales (con distintas frecuencias y amplitudes). Se usa ampliamente en compresión de video e imágenes, por ejemplo, en la compresión con pérdida de JPEG que se explica detalladamente en [2]. Li y Wang [12] presentaron una técnica que modifica la tabla de cuantificación (QT) del algoritmo de compresión con pérdida de JPEG e inserta los bits ocultos en los coeficientes de frecuencia media. Su QT modificado se muestra en la figura 3, esta nueva versión del QT les da 36 coeficientes en cada bloque de 8 x 8 para

incrustar sus datos secretos en los que se obtiene una carga útil razonable. Su trabajo fue motivado por un trabajo previamente publicado por Chang *et al.* [13]. La esteganografía basada en la compresión DCT JPEG pasa por diferentes pasos, como se muestra en la figura 4.

8	1	1	1	1	1	1	1
1	1	1	1	1	1	1	55
1	1	1	1	1	1	69	56
1	1	1	1	1	87	80	62
1	1	1	1	68	109	103	77
1	1	1	64	81	104	113	92
1	1	78	87	103	121	120	101
1	92	95	98	112	100	103	99

Figura 3 Tabla de cuantificación (QT) modificada por Li y Wang (fuente [2])

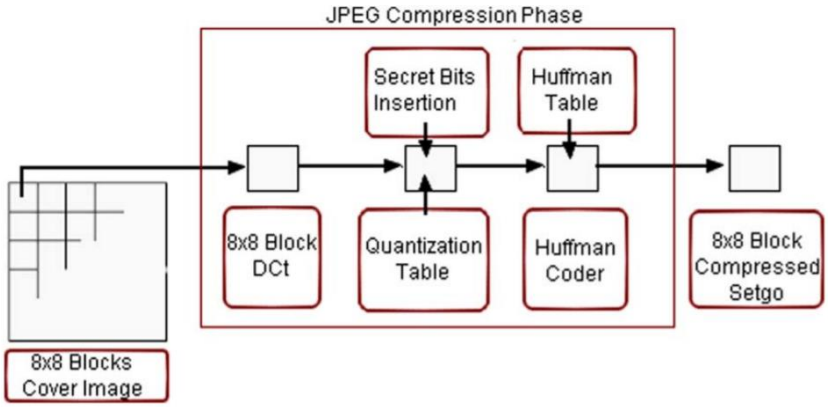

Figura 4 Diagrama de flujo que muestra el proceso general de incorporación al dominio de la frecuencia (fuente [2])

Otro de los métodos relacionado con la modificación de la tabla de cuantificación de la figura 3 es la denominada cuantificación perturbada (PQ), que tiene como objetivo lograr una alta eficiencia, con una distorsión mínima, en lugar de una gran capacidad. A cada coeficiente en el bloque DCT se le asigna un valor escalar que corresponde a cuánto impacto tendría en la imagen portadora, y luego se puede establecer una regla de selección para filtrar los coeficientes que mejor se comporten, dando así un algoritmo de menor capacidad, pero alta indetectabilidad.

En [2] se explican las técnicas mencionadas en esta sección y otras basadas en la DCT que han sido desarrolladas para mejorar la indetectabilidad, como "JSteg", "OutGuest" y "F5", pero estos desarrollos también han provocado una escalada, y han surgido técnicas de detección para atacarlos, "X2-test", "X2-test extendido" o la propuesta de Fridrich *et al.* [14] específica para "F5". Además, en [2] se explican varias técnicas de esteganografía en la transformada de ondículas discreta (DWT) que también formarían parte de la esteganografía en el dominio de la frecuencia.

3.1.3.4. Esteganografía Adaptativa

La esteganografía adaptativa es un caso especial de los dos tipos anteriores. También se conoce como "la incorporación con reconocimiento de estadísticas" o "el enmascaramiento perceptual". Este método toma ciertas características estadísticas globales de la imagen antes de intentar interactuar con sus coeficientes LSB / DCT. Las estadísticas dictarán dónde realizar los cambios. Se caracteriza por una selección adaptativa de píxeles en función de la imagen portadora y la selección de píxeles en un bloque con desviación estándar grande, esto último está destinado a evitar áreas de color uniforme también denominadas áreas lisas. Este comportamiento hace que la esteganografía adaptativa busque imágenes con ruido existente o agregado deliberadamente e imágenes que demuestran la complejidad del color. Wayner dedicó un capítulo completo de [15] a lo que llamó "la vida en el ruido", señalando la utilidad de la ocultación de datos en el ruido. Se ha demostrado que es robusto con respecto a la compresión, el recorte y el procesamiento de imágenes. En [2] se especifican varios trabajos y técnicas relacionadas con la esteganografía adaptativa.

3.1.4. Aplicaciones de la Esteganografía

Resulta evidente que la ocultación de mensajes usando la esteganografía puede tener fines legítimos o ilegítimos, que pueden ser beneficiosos para la proteger la privacidad de las comunicaciones, ser vehículos para perpetrar actos criminales o pueden ser de interés para burlar posibles censuras y para esquivar posibles restricciones en el acceso a Internet. Para dejar patente la importancia y la utilidad de su desarrollo, mencionamos algunos de los usos actuales de la esteganografía que se exponen Cheddad *et al.* [2].

Mejorar la solidez de los motores de búsqueda de imágenes, las identificaciones inteligentes en tarjetas de identidad donde los detalles de las personas están integrados en sus fotografías. La sincronización de audio y video, la circulación segura de datos secretos por parte de las empresas, la transmisión de televisión, los paquetes TCP/IP (por ejemplo, se puede incrustar una identificación única en una imagen para analizar el tráfico de red de usuarios particulares), la autenticación y seguridad en los protocolos de red.

La empresa Japonesa Fujitsu en 2007 estuvo desarrollando una aplicación para codificar datos en una imagen impresa y que estos se pudieran decodificar con un móvil en menos de un segundo, con la idea de conectar los materiales impresos al mundo web. La intención era transformar el esquema de color de la imagen antes de imprimir en sus componentes de tono, saturación y valor (HSV), y luego incrustarlo en el dominio de tono al que los ojos humanos no son sensibles. Esta aplicación se puede utilizar para recetas médicas, envoltorios de alimentos, vallas publicitarias, tarjetas de visita, medios impresos como revistas y folletos, y para reemplazar códigos de barras.

Cheddad *et al.* [2] también recogen las proposiciones de algunos investigadores sobre los usos potenciales de la esteganografía, por ejemplo, en la información médica donde se considera necesaria una separación confidencial entre los datos de imágenes de pacientes y otra información del paciente como el médico que les atiende, el nombre, la dirección y otros detalles personales, pero sin embargo debe mantenerse un vínculo entre ambos. Incrustar la información del paciente en la imagen podría ser una medida de seguridad útil y ayudaría a resolver estos problemas de confidencialidad. La esteganografía proporcionaría la máxima garantía de autenticación que ninguna otra herramienta de seguridad puede garantizar. También se han presentado técnicas para registros electrónicos de pacientes basadas en la ocultación de datos bipolares de múltiples bases.

La confianza en la integridad de las imágenes visuales se ha visto arruinada por la tecnología digital contemporánea. Esto llevó a una mayor investigación relacionada con el análisis forense de documentos digitales. Cheddad *et al.* [16] propusieron un esquema de seguridad que protege los documentos escaneados de la falsificación utilizando técnicas de auto incrustación. El método no solo señala la falsificación, sino que también permite a los expertos legales o forenses acceder al documento original a pesar de haber sido manipulado.

3.2. Estegoanálisis

Si bien la esteganografía consiste en técnicas para ocultar información, el estegoanálisis se define como la disciplina dedicada al estudio de detectar o estimar la información oculta en un estego-archivo con poco o ningún conocimiento de la técnica esteganográfica utilizada. Para poder decir que un estegoanálisis ha tenido éxito basta con que sea capaz de detectar la existencia de información oculta y no es necesario obtener la información en sí, ya que la función principal de la esteganografía es ocultar la comunicación. Los algoritmos estegoanalíticos más precisos son capaces incluso de determinar el tamaño de la información oculta.

Según explican Chandramouli y Subbalakshmi [17], una tarea importante para el estegoanálisis es definir qué características típicas tendría un estego-archivo genérico, ya que cada tipo de técnica puede dejar marcas muy diferentes. Si bien es posible diseñar una técnica de estegoanálisis razonablemente buena para un algoritmo esteganográfico específico, el objetivo a largo plazo es desarrollar un marco de estegoanálisis que pueda funcionar eficazmente al menos para una clase de métodos esteganográficos. La tendencia en estegoanálisis parece sugerir dos enfoques extremos, en el primero pocas suposiciones estadísticas sobre la imagen en investigación, las estadísticas se aprenden utilizando una gran base de datos de imágenes de entrenamiento y en el segundo se asume un modelo paramétrico para la imagen y sus estadísticas se calculan para la detección.

Para Muñoz *et al.* [18] las herramientas de estegoanálisis pueden facilitar al ciudadano corriente la capacidad de reforzar la seguridad de su entorno informático, la detección de la ocultación de los rastros locales en una máquina después de un ataque informático, la detección de herramientas de hacking ocultadas por un atacante en su máquina (por ejemplo, usando la fragmentación interna de un fichero), la detección de la existencia de troyanos, ...

Se ha mencionado a lo largo del documento en varias ocasiones, el ataque a nuestros estego-archivos. Esto es otra manera, habitual en la literatura, de referirse al estegoanálisis ya que de alguna forma se busca "destruir" el algoritmo esteganográfico descubriendo o evitando la comunicación oculta como si atacarás el estego-archivo. En los siguientes apartados se exponen diferentes clasificaciones que puede tener el estegoanálisis y en concreto se explica el estegoanálisis por par de valores que se ha utilizado para atacar a las técnicas que se han desarrollado.

3.2.1. Clasificación de los métodos estegoanalíticos

De la misma manera que en las técnicas esteganográficas, hay diferentes formas de dividir las técnicas de estegoanálisis. Muchos autores han realizado la suya propia, pero se van a mencionar solo algunas de ellas.

La clasificación más básica que encontramos es la división entre estegoanálisis manual y estadístico. El manual consiste en buscar manualmente diferencias entre el archivo portador y el estego-archivo como su propio nombre indica, buscando cambios en la estructura para localizar la información oculta. Esto implica varios problemas, el primero que necesitamos el archivo original que difícilmente se podrá disponer y el segundo que de esta manera descifrar la información oculta u obtener información de esta, como por ejemplo el tamaño, es casi imposible. El estegoanálisis estadístico consiste en un análisis estadístico del estego-archivo como su propio nombre indica, para hallar indicios de que se ha ocultado información en él. En imágenes, por ejemplo, se estudia la frecuencia de distribución de colores.

Francia y Gomez [19] recogen la clasificación que hace Wayner [15] sobre los posibles ataques que puede recibir un método esteganográfico. Para definir esta clasificación se usan distintos parámetros, la información de la que dispone el atacante y la forma de atacar el estego-archivo:

- ***Stego-Only attack***: El atacante dispone solo del estego-archivo, este ataque se basa en un análisis estadístico del contenido del mismo archivo.

- ***Known-Message attack***: Para este tipo de ataque se necesita conocer el estego-archivo y el archivo portador. Este ataque trata de analizar las diferencias de los dos archivos con la intención de extraer la información oculta.

- ***Multiple encoded files attack***: Este tipo de ataque se realiza cuando se esconde información en varios archivos, se intenta extraer información de cada uno para reconstruir el mensaje completo.

- ***Known-Cover attack***: Para este tipo de ataque se necesita tener el algoritmo esteganográfico utilizado, y con ese conocimiento previo se extrae la información oculta.

- ***Destroy everything attack***: En este tipo de ataque simplemente se sobrescribe todo el estego-archivo para evitar que pueda llegar la información oculta a su destinatario.

- ***Random tweaking attack:*** El atacante puede alterar aleatoriamente partes del archivo en un intento de hacer que la información oculta sea indescifrable.

- ***Add new information attack***: El atacante escribe en partes del archivo con la intención de sobrescribir en partes con la información oculta.

- ***Reformat attack***: Este tipo de ataque se basa en guardar el archivo en un formato diferente esperando que el cambio de formato destruya la información oculta. Un ejemplo sería tan simple como reformatear un archivo de imagen JPEG en un archivo de imagen GIF.

- ***Compression attack***: Sabiendo que los programas de compresión normalmente descartarían información extraña, el atacante ejecutaría un programa de compresión utilizando el archivo como entrada.

Chandramouli y Subbalakshmi [17] proponen dos enfoques diferentes para la clasificación del estegoanálisis de imágenes, aunque también indican que son aplicables a otros tipos de datos. En el primero, el criterio que tienen en cuenta es la estrategia que se va a aplicar para hacer el estegoanálisis y los grupos serían los siguientes: estegoanálisis basado en el aprendizaje supervisado, estegoanálisis basado en identificación ciega y estegoanálisis estadístico paramétrico. También existiría un grupo de técnicas híbridas que combinan más de uno de los anteriores, y según la combinación que hagan obtendrán unas ventajas e inconvenientes. El segundo enfoque que proponen se basa en el tipo de información que se analizará del estego-archivo para realizar el estegoanálisis, en este caso solo hay dos grupos: el estegoanálisis basado en la diversidad espacial y el estegoanálisis basado en la diversidad temporal: el estegoanálisis basado en la diversidad de espacial busca información en el dominio espacial que se repite en varias formas y en diferentes ubicaciones espaciales (por ejemplo, diferentes bloques dentro de una imagen o, en diferentes imágenes) y estegoanálisis basado en diversidad temporal que busca la información esteganográfica que aparece repetidamente a lo largo del tiempo, se realizará para archivos de video o audio.

En referencia a la división por tipos de estrategia se puede decir que cada uno de los grupos tiene ventajas y problemas que Chandramouli y Subbalakshmi [17] también han analizado y los recogemos en los siguientes apartados.

3.2.1.1. Estegoanálisis basado en el aprendizaje supervisado

Estas técnicas emplean estrategias de dos fases: fase de formación y fase de prueba. En la fase de formación, se genera una base de ejemplos de tipo $\{(d_i, t_i)\}$ donde d_i indica una propiedad de la estego-imagen y t_i indica si hay información oculta o no, de esta forma se aprende la mejor regla para indicar si hay o no información oculta. En la fase de prueba, una vez realizado el aprendizaje se proporcionan imágenes de entrada para decidir si hay información oculta presente o no. Hay algunos métodos de estegoanálisis que no utilizan directamente este tipo de aprendizaje clásico, sino que los datos de entrenamiento se utilizan para calcular un modelo de regresión para un conjunto de características seleccionadas, y este modelo se utiliza para hacer el estegoanálisis. Algunos factores a favor de los algoritmos de estegoanálisis basados en el aprendizaje son los siguientes:

- Se ha observado que el estegoanálisis basado en el aprendizaje funciona bastante bien utilizando características como estadísticas de coeficientes de ondas, métricas de calidad de imagen, ...

- Al entrenar para un algoritmo de incrustación específico, se puede lograr una detección razonablemente precisa. Ya que se utilizan múltiples ejemplos, no es necesario asumir modelos estadísticos previos para las imágenes. Se aprende un modelo promediando varios ejemplos.

- A partir de las técnicas basadas en el aprendizaje se pueden construir detectores universales.

- La no estacionariedad de las imágenes no plantea un problema importante debido al proceso de promediado.

- Dado que el aprendizaje automático ha sido un área de investigación activa durante varios años, existe una teoría y una metodología general bien desarrolladas.

Este tipo técnicas también están limitadas por los siguientes factores:

- Se debe entrenar por separado para cada algoritmo de incrustación. Esto puede llevar mucho tiempo y, en ocasiones, resultar poco práctico.

- La elección de las características adecuadas para entrenar al algoritmo es un paso crítico. Si las características seleccionadas no son apropiadas para el algoritmo de incrustación específico, entonces la detección puede fallar por

completo. No existe una regla sistemática para la selección de funciones. Es principalmente un método heurístico, de prueba y error.

- Algunos algoritmos tienen varios parámetros que debe elegir el estegoanalista. Esta podría ser una tarea abrumadora. No existe una manera sencilla de elegir estos parámetros.

- Cualquier método basado en el entrenamiento adolece del clásico equilibrio entre el sesgo y la varianza. Es decir, el algoritmo se puede entrenar muy bien para dar una precisión muy alta para las imágenes de entrenamiento, pero puede perder la capacidad de generalización para funcionar en las imágenes de prueba.

- Es extremadamente difícil o incluso imposible identificar partes de la imagen donde un mensaje está oculto, extracción de mensajes, ... El objetivo final de estas técnicas es llegar a una decisión simple: presencia o ausencia de información oculta.

3.2.1.2. Estegoanálisis basado en identificación ciega

Los métodos de identificación ciega plantean el problema como la identificación del sistema. Se explotan algunas propiedades estadísticas como la independencia del portador y el mensaje secreto, entre otras. El algoritmo de incrustación se representa como un canal y el objetivo es invertir este canal para identificar el mensaje oculto. Algunas de las ventajas de utilizar técnicas basadas en la identificación ciega son las siguientes:

- En esta formulación no hay datos de entrenamiento. Cada imagen se analiza individualmente según las estadísticas calculadas. Las estadísticas verdaderas estimadas de la imagen están disponibles para el estegoanalista en lugar de un promedio como en el estegoanálisis basado en el aprendizaje. Por lo tanto, las estadísticas calculadas reflejan las características de la imagen con mayor precisión.

- Con este tipo de técnicas es posible extraer la información oculta en lugar de una simple detección de su presencia.

- Dado que el marco de identificación ciega es bastante general, se pueden detectar varios algoritmos esteganográficos modelándolos dentro de este marco.

- Es posible derivar resultados analíticos que sugieran la viabilidad de un estegoanálisis exitoso para ciertos tipos de modelos estadísticos de la imagen portadora y la información oculta.

También existen algunos problemas con este tipo de técnicas, como se describe a continuación:

- Se sabe que las imágenes digitales son estadísticamente no estacionarias. Esto causa problemas prácticos en la implementación de algoritmos basados en el modelo de identificación ciega, ya que la identificación ciega asume inherentemente la estacionariedad de los datos.

- Cuando se viola la condición de estacionariedad, se necesita un esfuerzo adicional para que el estegoanálisis funcione. Esto puede necesitar algunos enfoques heurísticos, como el cálculo estadístico basado en ventanas móviles, suposición de estacionariedad por partes, ...

3.2.1.3. Estegoanálisis estadístico paramétrico

Estas técnicas tienden a asumir un modelo estadístico paramétrico para la imagen portadora, la estego-imagen y la información oculta. El estegoanálisis se formula como un problema de prueba de hipótesis, por ejemplo, H0: sin mensaje (hipótesis nula) y H1: mensaje presente (hipótesis alternativa). Luego, se diseña un algoritmo de detección estadística para probar entre las dos hipótesis. Suponiendo que un modelo de distribución de probabilidad paramétrico está disponible para el algoritmo, se encuentran las siguientes ventajas en esta clase de técnicas:

- La teoría de detección estadística paramétrica es un área temática bien desarrollada. Por lo tanto, muchos de los resultados conocidos en esta área se pueden aplicar de manera sencilla para investigar las reglas de detección de la información oculta.

- La característica de funcionamiento del receptor especifica el rendimiento del algoritmo de estegoanálisis. Esta es una curva con probabilidad de falsas alarmas en el eje X y probabilidad de detección en el eje Y. Por tanto, las tasas de error alcanzables se pueden deducir fácilmente.

- Un estegoanalista tiene control sobre la probabilidad de error de detección deseada. Los umbrales de detección se pueden calcular en forma cerrada para una restricción de probabilidad de error dada. A veces, incluso puede ser posible especificar restricciones tanto en la falsa alarma como en la probabilidad de detección.

- También es posible estimar la clave secreta, las ubicaciones de los mensajes, la longitud del mensaje, etc.

Algunos de los inconvenientes del estegoanálisis basado en estadísticas paramétricas son los siguientes:

- Por naturaleza, estas técnicas son sensibles a inexactitudes en las estimaciones estadísticas de ciertos parámetros. Así que si las estadísticas estimadas no reflejan fielmente las estadísticas de la imagen podría dar resultados inexactos.

- Asumir antecedentes probabilísticos es un tema polémico. Por lo general, se eligen subjetivamente. Por lo tanto, esto implica un mayor grado de participación del usuario en el proceso de estegoanálisis.

- La no estacionariedad estadística de las imágenes digitales plantea un serio problema práctico.

3.2.2. Estegoanálisis por par de valores

La técnica principal para detectar esteganografía LSB se conoce como **estegoanálisis por pares de valores** (PoV, del inglés «***Pairs of Values***») y fue propuesta en Westfeld y Fitzmann [20]. En concreto, un byte puede representar 256 valores (de 0 a 255). Si se agrupan estos 256 valores por pares contiguos, se obtienen 128 posibilidades: [0,1], [2,3], [4,5], …, estos pares de valores se denominan puntos de vista en la secuela. Si los bits utilizados para sobrescribir los bits menos significativos se distribuyen por igual, las frecuencias de ambos valores de cada PoV se vuelven iguales. La idea del ataque estadístico es comparar la distribución de frecuencia esperada con alguna distribución de muestra observada en el archivo portador potencialmente cambiado. Un punto crítico es cómo obtener la distribución de frecuencia esperada, esta frecuencia no debe derivarse de la muestra aleatoria, porque esta muestra aleatoria podría haber sido cambiada por operaciones esteganográficas. Pero en la mayoría de los casos no tenemos el original para comparar o derivar la frecuencia esperada. En el original, la frecuencia esperada es la media aritmética de las dos frecuencias en un punto de vista. Debido a que la función de incrustación sobrescribe los bits menos significativos, no cambia la suma de estas dos frecuencias. El recuento tomado de la frecuencia de valor impar se transfiere a la frecuencia de valor par correspondiente en cada PoV, y viceversa. Como la suma permanece constante, la media aritmética es la misma en ambos, el portador y el estego-archivo correspondiente. Este hecho nos permite obtener la distribución de frecuencias esperadas de la muestra aleatoria. Así que no necesitamos el medio de transporte original para el ataque.

El grado de similitud de la distribución de la muestra observada y la distribución de frecuencia esperada es una medida de la probabilidad de que se haya producido alguna incrustación. El grado de similitud se determina mediante el test χ^2, esta prueba realiza en un mapeo de observaciones en categorías y sigue los siguientes pasos:

- **Paso 1**: Supondremos que hay k categorías y que tenemos una muestra aleatoria de observaciones. Cada observación debe pertenecer a una única categoría. Las categorías son todos los índices de la paleta, cuyo color se coloca en un índice uniforme dentro de la paleta ordenada. Nos concentramos en los valores impares de los puntos de vista del portador. Su frecuencia mínima esperada debe ser mayor que 4, así podemos unificar categorías para mantener esta condición.

- **Paso 2:** La frecuencia esperada en la categoría i después de incrustar la información de forma uniforme es:

$$n_i^* = \frac{|\{\text{colour}|\text{sortedIndexOf(colour)} \in \{2i, 2i+1\}\}|}{2}$$

Figura 5 Fórmula de la frecuencia esperada (fuente [20])

- **Paso 3:** La frecuencia medida de ocurrencia en la muestra aleatoria es:

$$n_i = |\{\text{colour}|\text{sortedIndexOf(colour)} = 2i\}|$$

Figura 6 Fórmula de la frecuencia medida (fuente [20])

- **Paso 4:** El valor χ^2, se da como $\chi^2_{k-1} = \sum_{i=1}^{k} \frac{(n_i - n_i^*)^2}{n_i^*}$

- **Paso 5:** Se calcula la probabilidad de que las distribuciones de n_i y n_i^* sean iguales. realiza la integración de la función de densidad:

$$p = 1 - \frac{1}{2^{\frac{k-1}{2}}\Gamma(\frac{k-1}{2})} \int_0^{\chi^2_{k-1}} \mathrm{e}^{-\frac{x}{2}} x^{\frac{k-1}{2}-1} \mathrm{d}x$$

Figura 7 Fórmula de la probabilidad (fuente [20])

Este test permite descartar que la imagen analizada contenga información oculta mediante esteganografía LSB.

3.3. Herramientas existentes

Actualmente hay muchas herramientas de esteganografía, se centran en ocultar archivos dentro de otros y antes que ocultar mensajes de texto, como se ha realizado en nuestra implementación, aunque también podemos encontrar algunas herramientas en el mercado que lo hacen. En los siguientes puntos presentamos un listado de aplicaciones esteganográficas existentes:

- **Xiao Steganography:** Es un software gratuito que se puede utilizar para ocultar archivos en imágenes BMP o en archivos WAV. Para leer el mensaje oculto de este archivo, tendrá que volver a utilizar este software. Este software leerá el archivo y decodificará el archivo oculto de él. No puede extraer el archivo oculto de ningún otro software.

- **Image Steganography:** Es también un software gratuito para ocultar información en archivos de imagen. Puede ocultar mensajes de texto o archivos dentro de un archivo de imagen. Se puede usar la opción de decodificación de la misma herramienta para decodificar el archivo o mensaje oculto de la imagen.

- **Camouflage:** Es una buena herramienta de esteganografía te permite ocultar cualquier tipo de archivo dentro de otro. No hay ningún tipo de restricción en el software para ocultar información. El uso de la herramienta también es simple y fácil. Para extraer sus datos confidenciales del archivo, haga clic con el botón derecho y seleccione Desmarcar. También puede establecer una contraseña para encriptar los datos ocultos dentro del archivo. Ahora el proyecto ya no está en desarrollo.

- **SteganPEG:** Permite ocultar cualquier tipo de archivo en un archivo de imagen JPG. Puede adjuntar cualquier archivo y dar la contraseña para ocultar dentro de un archivo JPG. Solo puede extraer su archivo de esa imagen JPG con la misma herramienta. La interfaz del software es simple con menos opciones. Por lo tanto, uno puede usar fácilmente esta herramienta para ocultar datos en un archivo de imagen JPG.

- **SSuite Picsel:** Es una aplicación portátil gratuita para ocultar el texto dentro del archivo de imagen. Sin embargo, tiene un enfoque diferente. Utiliza el archivo de imagen como clave para proteger el texto oculto dentro de una imagen. Esta herramienta puede ocultar texto dentro de un archivo de imagen. Sin embargo, para ocultar y mostrar el texto dentro de una imagen, debe ingresar otra imagen como clave.

También podemos encontrar herramientas de estegoanálisis, de diversos tipos, algunas destruyen la información oculta, otras son capaces de detectarla y otras son capaces de extraerla, enunciamos algunas de cada tipo:

- **Destruyen la información oculta:** 2Mosaic, StirMark Benchmark, Phototile.

- **Detectan la información oculta:** Steganography Analyzer Real-Time Scanner (Analiza paquetes de red), StegDeteect.

- **Extraen la información oculta:** StegBreak y Stego-Suite.

4. Técnicas implementadas

En la sección anterior hemos analizado la esteganografía, viendo las características que tienen las técnicas esteganográficas y algunas clasificaciones que se pueden hacer de ellas, analizando algunos trabajos teóricos de métodos esteganográficos para imágenes. Todo esto nos sirve como punto de partida para contextualizar el desarrollo realizado, una biblioteca Python con diversas técnicas esteganográficas: **LSB** (modificación del bit menos significativo con una derivada de la codificación Elias delta), **PVD** (diferencia de valor de píxel), **EMD** (explotación de la modificación de la dirección) y **LSB+PVD+EMD** (es una técnica que combina las tres anteriores).

Mediante los distintos apartados de esta sección se explican las técnicas desarrolladas y todo lo necesario para entender el trabajo realizado. Primero exponemos los trabajos teóricos en los que se han basado las técnicas desarrolladas y los distintos conceptos que se hayan utilizado. A continuación, describimos la estructura de la implementación desarrollada además de analizar las particularidades y modificaciones realizadas en nuestras técnicas con respecto a los trabajos teóricos. Para finalizar, comentamos los test unitarios realizados para probar nuestras técnicas.

4.1. Planteamiento teórico de las técnicas implementadas

La primera técnica que se ha implementado es una técnica de modificación del bit menos significativo en cada componente del color RGB de cada píxel de una imagen, aplicando una derivación de la codificación Elias delta a la información a ocultar, la llamaremos a partir de ahora técnica LSB. La codificación Elias delta es una codificación universal que codifica números enteros positivos y que fue desarrollada por Peter Elias [21]. Se deben seguir una serie de pasos simples para codificar un número $X \geq 1$, que son los siguientes:

1. Descomponer número X a binario en N dígitos binarios.
2. Descomponer número N a binario en L dígitos binarios.
3. Escriba L-1 ceros.
4. Continúe escribiendo la codificación binaria de N.
5. Continúe escribiendo la codificación binaria de X menos el bit inicial.

Si queremos codificar por ejemplo el 5 siguiendo los pasos primero lo pasamos a binario 101, tenemos 3 dígitos por lo tenemos que pasar 3 a binario, es decir 11. Tenemos 2 dígitos. Escribimos 2-1 ceros, es decir un 0. Añadimos la codificación de 3 011 y luego añadimos el numero en binario sin el bit inicial 01101 sería 5 en codificación Elias Delta. Cuando queremos decodificar un entero codificado en Elias delta se deben realizar los siguientes pasos:

1. Lea todos los ceros hasta llegar al primer uno. Ese número de ceros será L.

2. Después del primer uno lea los L siguientes dígitos.

3. Con el uno y los L dígitos binarios leídos obtendremos el numero N+1 en binario. Lo pasamos a decimal y le restamos uno obteniendo el número N.

4. Ponemos un uno y le agregamos los siguientes N dígitos, con lo que obtendremos el número en binario y lo pasamos a decimal. Y ya tendríamos el número decodificado.

Si por algún motivo estos pasos no se pudieran seguir, si no hay suficientes dígitos o hay demasiados, significaría que no se ha realizado una codificación Elias delta o se ha realizado erróneamente. Para ver un ejemplo de decodificación correcta usaremos el numero codificado 00111001010. Primero leeremos todos los ceros antes del primer uno, en este caso tenemos 2, después leeremos los 2 dígitos siguientes después del primer uno que son 11. Añadiendo al primer uno lo que hemos leído obtenemos la codificación en binario de los N+1 dígitos siguientes que tenemos que leer, 111 que es 7 en decimal por lo tanto tenemos que coger los 6 dígitos siguientes en este caso 001010. Agregando a un uno los dígitos obtenidos obtenemos el número en binario 1001010 que es el número 74 en decimal.

Esta técnica según las diferentes clasificaciones explicadas en la sección 3.1.2 es una técnica de **esteganografía pura**, ya que no se usan claves para codificar la información, y es una **técnica de sustitución** ya que sustituye los datos de los LSB en los píxeles de la imagen por los datos por los datos que queremos ocultar y también formaría parte de la **esteganografía en el domino espacial**.

El resto de las técnicas se han implementado usando como referencia el trabajo de Pradhan *et al.* [22] que presentan una técnica de sustitución LSB combinada con las técnicas de diferencia de valor de píxel, PVD, y la técnica de la explotación de la modificación de la dirección, EMD. A la técnica conjunta se la llama LSB+PVD+EMD, en la literatura se pueden encontrar diferentes proposiciones de las técnicas que la componen y son referenciadas en [22]. PVD es una técnica que utiliza la diferencia entre un píxel y sus adyacentes para codificar más o menos bits y EMD es una técnica que utiliza modificaciones en diferentes direcciones para representar diferentes datos secretos.

En concreto en [22] proponen dos variantes de la técnica LSB+PVD+EMD, una para bloques de píxel 2x2 y otra para bloques de píxeles 3x3, nuestro trabajo se centra en la variante de bloques 2x2. Como cualquier otra técnica esteganográfica tiene un

procedimiento de codificación o incrustación y otro de decodificación o extracción, como explicamos en la sección 3.1.1. El proceso de incrustación de la información oculta consta de los siguientes pasos:

1. La imagen se recorre de manera rectangular de izquierda a derecha y de arriba abajo, y se divide en bloques no superpuestos de tamaño 2×2. En la figura 8 (a) vemos un bloque de muestra.

2. Para cada bloque se calcula la diferencia de valor de píxel media, con la siguiente fórmula $d = (1/3)\ \sum_{i=1}^{3}|P_c - P_i|$. Si d es mayor que 15 entonces se dice que el bloque es un área de borde, si no es un área lisa.

3. En un área de borde, la incrustación se realiza mediante sustitución de LSB + PVD y en un área lisa, la incrustación se realiza mediante sustitución de LSB + EMD.

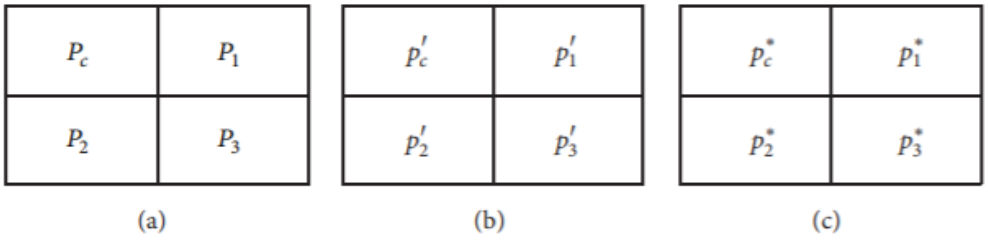

Figura 8 (a) bloque de píxeles del portador, (b) bloque de píxeles de la estego-imagen (c) bloque de píxeles de la estego-imagen usada en la extracción. (fuente [22])

Range, $\{l_i, u_i\}$	$R_1 = \{0,7\}$	$R_2 = \{8,15\}$	$R_3 = \{16,31\}$	$R_4 = \{32,63\}$	$R_5 = \{64,127\}$	$R_6 = \{128,255\}$
No of bits to be hidden, n_i	3	3	3	3	4	4

Tabla 1 Tabla de rangos tipo 1 (fuente [22])

Range, $\{l_i, u_i\}$	$R_1 = \{0,7\}$	$R_2 = \{8,15\}$	$R_3 = \{16,31\}$	$R_4 = \{32,63\}$	$R_5 = \{64,127\}$	$R_6 = \{128,255\}$
No of bits to be hidden, n_i	3	3	4	5	6	6

Tabla 2 Tabla de rangos tipo 2 (fuente [22])

4.1.1. Algoritmo incrustación LSB+PVD

El primer LSB del píxel P_c se sustituye por el bit 1, para actuar como indicador durante la extracción. Los otros 2 LSB se sustituyen por 2 bits de datos. Se obtiene un nuevo valor del píxel P_c'. Suponemos que el valor decimal de los tres LSB de P_c' es s_1 y el valor decimal de los tres LSB de P_c es i_1 . Se calcula un valor de diferencia $df_1 = i_1 - s_1$ y P_c' se optimiza mediante:

$$p_c' = \begin{cases} p_c' + 2^3, & \text{if } \mathrm{df}_1 > 2^{3-1},\ 0 \le \left(p_c' + 2^3\right) \le 255 \\ p_c' - 2^3, & \text{if } \mathrm{df}_1 < -2^{3-1},\ 0 \le \left(p_c' - 2^3\right) \le 255 \\ p_c', & \text{otherwise} \end{cases}$$

Figura 9 Fórmula que optimiza el valor de P_c' según el valor de la diferencia (fuente [22])

Ahora se calculan las diferencias para el resto de los píxeles con respecto al píxel origen del bloque, $d_i = |P_c' - P_i|$, donde i toma los valores 1, 2 y 3. El número de bits ocultos, n_i en cada píxel dependerá de d_i y en que rango esté dentro la tabla de rangos, que dependiendo de la implementación que se desee hacer se puede elegir entre la tabla 1 y la tabla 2. Se convierten los n_i bits ocultos a su equivalente en decimal, ds_i , se calcula el valor $d_i' = l_i - ds_i$. Para todos los P_i se calculan dos posibles valores $P_i'' = P_c' - d_i'$ y $P_i''' = P_c' + d_i'$ y se selecciona un valor para P_i' siguiendo la siguiente fórmula:

$$p_i' = \begin{cases} p_i'', & \text{if } \left|p_i - p_i''\right| < \left|p_i - p_i'''\right|,\ 0 \le p_i'' \le 255 \\ p_i''', & \text{otherwise} \end{cases}$$

Figura 10 Formula que calcula P_i' en el método PVD (fuente [22])

4.1.2. Algoritmo incrustación LSB+EMD

El primer LSB del píxel P_c se sustituye por el bit 0, para actuar como indicador durante la extracción. Los otros 2 LSB se sustituyen por 2 bits de datos. Se obtiene un nuevo valor de este píxel P_c'. Suponemos que el valor decimal de los tres LSB de P_c' es s_1 y el valor decimal de los tres LSB de P_c es i_1. Se calcula el valor de diferencia $df_1 = i_1 - s_1$ y P_c' se optimiza por la fórmula de la figura 9.

Suponemos P_k para el resto de los píxeles del bloque donde k toma los valores 1, 2 y 3 y para cada P_k ocultamos 2 bits de datos, que los pasamos a su equivalente en decimal como m_k . Ahora calculamos $P_k'' = P_k$ + x donde x es un valor entre $\{-3,-2,-1,0\}$ y $P_k'' \bmod 4 = m_k$ y también se calcula $P_k''' = P_k$ + x donde x es un valor entre $\{3,2,1\}$ y $P_k''' \bmod 4 = m_k$, si para ningún valor de x se satisface que

$P_k''' \ mod\ 4 = m_k$ entonces P_k''' = -10. Ahora calculamos el valor P_k' por la fórmula de la figura 11. La figura 8 (b) representa la salida del bloque de píxeles en la estego-imagen.

$$p_k' = \begin{cases} p_k'', & \text{if } \{(p_k''' < 0 \text{ or } p_k''' > 255),\ 0 \le p_k'' \le 255\} \text{ or } \{0 \le (p_k'', p_k''') \le 255,\ |p_k - p_k''| \le |p_k - p_k'''|\} \\ p_k''', & \text{if } \{(p_k'' < 0 \text{ or } p_k'' > 255),\ 0 \le p_k''' \le 255\} \text{ or } \{0 \le (p_k'', p_k''') \le 255,\ |p_k - p_k'''| \le |p_k - p_k''|\} \end{cases}$$

Figura 11 Fórmula que calcula P_k' en el método EMD (fuente [22])

4.1.3. Algoritmo Extracción LSB+PVD+EMD

El procedimiento de extracción de la información oculta en una estego-imagen consta de los siguientes pasos:

1. La imagen se recorre de manera rectangular de izquierda a derecha y de arriba abajo, y se divide en bloques no superpuestos de tamaño 2×2. En la figura 8 (c) se muestra un bloque de muestra.

2. Se comprueba el LSB de P_c^*, si el valor es 1, se utiliza el procedimiento de extracción de LSB+PVD. Este procedimiento consiste en extraer los 2 siguientes bits menos significativos de P_c^* y calcular $d_i^* = |P_c^* - P_i^*|$ y $s_i^* = d_i^* - l_i$ donde i toma los valores 1, 2 y 3, d_i^* está dentro de un rango de la tabla de rangos utilizada o tabla 1 o tabla 2 , y donde l_i es el límite inferior del rango al que pertenece d_i^*. Ahora al convertir cada s_i^* a número binario y se han obtenido los bits ocultos en esos píxeles.

3. Si el valor del LSB de P_c^* es 0, entonces se utiliza el procedimiento de extracción de LSB+EMD. Este procedimiento consiste en extraer los 2 siguientes bits menos significativos de P_c^*, para el resto de los píxeles del bloque P_1^*, P_2^*, P_3^*, se realiza el módulo de 4, $m_k = P_k^* \ mod\ 4$, donde k toma los valores de 1, 2 y 3. Ahora convertimos cada m_k a número binario y hemos obtenido los bits ocultos en esos píxeles.

Esta técnica conjunta de LSB+PVD+EMD aparte de situarse en los mismos tipos de la anterior técnica de LSB **esteganografía pura, técnica de sustitución y esteganografía en el dominio espacial de la imagen**, también forma parte la **esteganografía adaptativa.** Debido a que está eligiendo entre áreas lisas y de borde para utilizar un método más agresivo (que oculta más bits de datos) o menos, ya utilicemos la tabla 1 o tabla 2 como tabla de rangos.

En el propio resumen de [22] los autores comentan "Las capacidades de ocultación y la proporción máxima de señal ruido (PSNR) de ambas técnicas se encuentran mejoradas. Los resultados de los experimentos demuestran que el análisis del

histograma de la diferencia de píxeles (PDH) y el análisis de RS [23] no pueden detectar estas técnicas propuestas".

4.2. Exposición de la implementación

El objetivo de este proyecto era realizar una biblioteca de técnicas esteganográficas para imágenes en Python, las técnicas elegidas para empezar esta biblioteca han sido las expuestas anteriormente **LSB**, **PVD**, **EMD** y **LSB+PVD+EMD**. Para realizar operaciones con imágenes se ha usado la librería de PIL de Python, se han hecho pruebas sobre imágenes PNG y BMP comprobando que los métodos funcionan correctamente, y sabemos que no funciona sobre imágenes JPG, el resto de los formatos no se han probado. La estructura que se ha elegido para la implementación ha sido la creación de módulos Python para cada uno de los métodos desarrollados, con un mínimo de una función para codificar un mensaje de texto dado en una imagen dada, que crea una nueva estego-imagen con el mensaje embebido utilizando la técnica correspondiente, y otra función que decodifique un mensaje oculto con la técnica correspondiente en una estego-imagen dada. Además, se han creado varios módulos de utilidades, uno de ellos con funciones transversales que no tienen relación directa con ninguna de las técnicas esteganográficas en concreto (myutils.py) como por ejemplo abrir una imagen o transformar un mensaje de texto a bits. El otro módulo creado (myconstants.py) es una lista de constantes usadas. También hay un main (dfsetegomain.py) donde con una serie de comandos se pueden probar las funciones de codificación y decodificación de cada una de las técnicas. Un esquema de la estructura de directorios de la aplicación sería el siguiente:

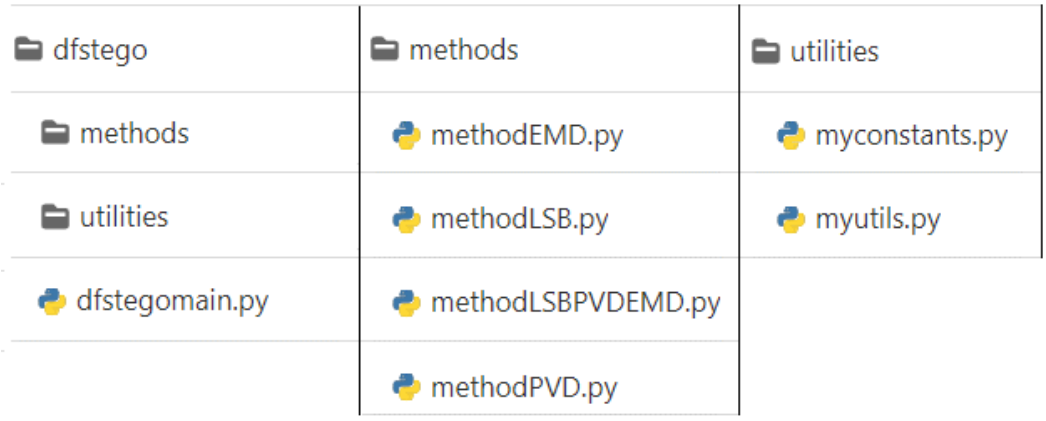

Figura 12 Esquema de la estructura de directorios de la aplicación.

4.2.1. Implementación Método LSB

Se ha realizado una implementación de sustitución LSB básica, recorriendo la matriz de píxeles de la imagen de izquierda a derecha y de arriba abajo, sustituyendo en cada una de las componentes del color del píxel (rojo, verde y azul) el LSB por los bits del mensaje dado. El mensaje de texto se codifica en UTF-8 y al decimal que nos devuelve la función estándar de Python le añadimos una codificación binaria con una variante de la codificación Elias delta explicada en la sección 4.1.

La variación con respecto al algoritmo original es en el paso 3 donde en lugar de añadir L-1 ceros añadimos directamente L ceros, y el paso 5 donde en lugar de añadir la codificación binaria de X sin el bit inicial lo que hacemos es añadirla con el bit inicial. Así para decodificar el mensaje oculto lo que hacemos es recorrer la matriz de píxeles de la imagen de izquierda a derecha y de arriba abajo y por las componentes de color rojo, verde y azul. Mientras el LSB sea 0, contamos los L ceros leídos, y recorremos las siguientes L componentes para obtener el número N de bits del mensaje oculto según indica la codificación Elias delta explicada en la sección 4.1. Finalmente recorremos las N siguientes componentes para obtener los bits del mensaje, pasamos los bits obtenidos a decimal y decodificamos el UTF-8 para obtener el mensaje de texto oculto.

La decisión técnica de añadir la codificación Elias delta al método de sustitución LSB fue debido a que en la versión inicial del método se indicaba en el primer píxel de la imagen de manera oculta, el tamaño en bits del mensaje. Esto suponía un problema por dos motivos, el primero estábamos indicando el tamaño del mensaje oculto de una manera demasiado evidente y el segundo nos limitaba el tamaño del mensaje demasiado, añadiendo esta variante conseguimos que el propio procedimiento de decodificación nos indique cuando acabar de aplicar el algoritmo, aumenta la capacidad y se oculta mejor el mensaje. Las variaciones con respecto a la codificación teórica se realizaron para evitar tener que añadir operaciones o modificaciones en los datos y así simplificar el código.

4.2.2. Implementación Método PVD

Para realizar la implementación del método PVD nos hemos basado en la incrustación LSB + PVD explicada en la sección 4.1.1 pero en lugar de solo en las áreas de borde, en toda la imagen. Por motivos de simplificación y para una mejor reutilización del código se ha tomado la decisión técnica de realizar algunas modificaciones sobre la teoría. En concreto en P_c en lugar de codificar dos bits de datos ocultos, no se hace, para simplificar la finalización del algoritmo en el método LSB+PVD+EMD.

Al igual que el método LSB codificamos un mensaje de texto en UTF-8 y pasamos el decimal que nos da a binario, y este es la información que vamos a ocultar en nuestra

imagen. En este método al contrario que el método LSB, solo incrustamos en la componente de color roja de cada píxel, pero como se explica en el paso 1 de la técnica LSB+PVD+EMD recorremos la matriz de píxeles de izquierda a derecha y de arriba abajo en bloques de 2x2 píxeles. Sustituimos el LSB de P_c por el bit 1, indicando que en este bloque se codifica parte del mensaje, y aplicamos la fórmula de la figura 9 para decidir el valor de P_c' , sumando o restando 8 dependiendo de las condiciones. Con este valor guardado se procede a calcular el valor del resto de píxeles del bloque. Calculamos d_i cómo se indica en la sección 4.1.1 y se decide cuantos bits se van a codificar en el píxel según la tabla 1, que hemos decidido seguir como tabla de rangos. En este punto realizamos una modificación respecto al trabajo de Pradhan *et al.* [22] ya que sustituimos el LSB de la componente azul del píxel al bit 1, indicando que en este píxel se codifica parte del mensaje y que no es el último, y si es el último en el que se codifica información, es decir, no queda más bits por codificar después de este píxel, sustituimos el LSB de la componente azul del píxel al bit 0. Si estamos en el segundo caso en la componente verde se codifican en los dos bits LSB los bits codificados de 0 a 3, y en la componente roja en los LSB que se indique en la componente verde se sustituyen por los del mensaje. Si estamos en el primer caso continuamos aplicando el algoritmo como se indica en la sección 4.1.1 obteniendo el valor de P_i' según la fórmula de la figura 10. Repetimos esto hasta que acabemos de guardar todo el mensaje o se acabe la imagen. Si no hemos agotado los bloques potenciales a codificar sustituimos el LSB de P_c del bloque siguiente al último que guarda información oculta con el bit 0 para indicar que ya no hay más bits codificados.

Para decodificar el mensaje se recorre la imagen como se explica en el paso 1 del algoritmo de extracción de la sección 4.1.3, bloque a bloque. Se comprueba que el LSB de P_c^* del bloque es del valor 1, si es así se procede a extraer los bits en el bloque, si el valor es 0 acabamos el algoritmo. Para extraer los bits codificados en cada píxel, primero comprobamos que el LSB de la componente azul del píxel P_i^* tiene valor 1, si es así se aplicaría el paso 2 del algoritmo de extracción en lo que al P_i^* se refiere, y si tiene valor 0 se obtendrían los 2 bits LSB de la componente verde de P_i^* para ver cuántos bits están codificados en la componente roja y luego extraeríamos esos bits de esta, añadiéndolos a los bits de salida, además de indicar que el algoritmo finaliza. Después los bits obtenidos del algoritmo se pasarían a decimal y decodificaríamos UTF-8 para obtener el mensaje de texto oculto.

Las modificaciones con respecto al trabajo teórico se han realizado debido a que no se indica ninguna manera de finalizar el algoritmo en la exposición teórica. Se decidió diferenciar la finalización del algoritmo con una implementación muy diferente a la definida.

4.2.3. Implementación Método EMD

Para realizar la implementación del método EMD nos hemos basado en la incrustación LSB + EMD explicada en la sección 4.1.2, pero en lugar solo en las áreas lisas, en toda la imagen. Al igual que el método PVD en P_c no se codifican bits ocultos por los mismos motivos. Como en los métodos anteriores se codifica el mensaje de texto en UTF-8 y pasamos el decimal que nos da a binario, siendo esta la información que vamos a ocultar en la imagen. Como se explica en el paso de 1 de la técnica LSB+PVD+EMD recorremos la matriz de píxeles de izquierda a derecha y de arriba abajo en bloques de 2x2 píxeles y ocultaremos bits de información oculta en la componente roja de estos. Sustituimos los dos bits LSB de P_c por los bits 00, indicando que en este bloque se codifica parte del mensaje, y aplicamos la fórmula de la figura 9 para decidir el valor de P_c'. Para calcular el valor del resto de píxeles del bloque aplicamos el algoritmo como se indica en la sección 4.1.2 con la excepción que en el LSB de la componente azul del píxel sustituimos al bit 0, indicando que en este píxel se codifica parte del mensaje y que no es el último. Si es el último píxel en el que se codifica información, sustituimos el LSB de la componente azul del píxel al bit 1. Si estamos en el segundo caso en la componente verde se codifica el LSB para indicar si hay 0 o 1 bits codificados, y en la componente roja los LSB que se indiquen en la componente verde se sustituyen por los del mensaje. Si no hemos agotado los bloques potenciales a codificar sustituimos el LSB de P_c del bloque siguiente al último que guarda información oculta con el bit 1 para indicar que ya no hay más bits codificados.

Para decodificar el mensaje se recorre la imagen como se explica en el paso 11 del algoritmo de extracción de la sección 4.1.3, bloque a bloque. Se comprueba que los dos bits LSB de P_c^* del bloque son del valor 00, si es así se procede a extraer los bits en el bloque, si no acabamos el algoritmo. Para extraer los bits codificados en cada píxel, primero comprobaríamos que el LSB la componente azul del píxel P_i^* tiene valor 0, si es así se aplicaría el paso 3 del algoritmo de extracción en lo que al P_i^* se refiere, y si tiene valor 0 se obtiene el LSB de la componente verde de P_i^* para ver cuántos bits están codificados en la componente roja y luego extraeríamos los bits indicados de esta, añadiéndolos a los bits de salida, además de indicar que el algoritmo finaliza. Después los bits obtenidos del algoritmo se pasarían a decimal y decodificaríamos UTF-8 para obtener el mensaje de texto oculto.

Las modificaciones con respecto al trabajo teórico se han realizado debido a que no se indica ninguna manera de finalizar el algoritmo en la exposición teórica. Se decidió diferenciar la finalización del algoritmo con una implementación muy diferente a la definida.

4.2.4. Implementación Método LSB+PVD+EMD

El método LSB+PVD+EMD se ha implementado como una mezcla de los tres métodos anteriores. Codificamos el mensaje de texto como en el método LSB explicado en la sección 4.2.1, aplicamos los pasos 1 y 2 de la técnica LSB+PVD+EMD explicada en 4.1, y si al calcular d es un área de borde aplicamos el algoritmo de incrustación explicado en la sección 4.2.2 en el método PVD y si es una área lisa aplicamos el algoritmo de incrustación explicado en la sección 4.2.3. en el método EMD. Para finalizar el algoritmo sustituimos los dos bits LSB de P_c por los bits 10, para poder realizar esta finalización se tomó la decisión técnica de no ocultar dos bits de datos en P_c en los métodos PVD y EMD, como dicta la técnica descrita por Pradhan *et al.* [22].

Para decodificar el mensaje aplicamos el paso 1 de la técnica de extracción descrita en 4.1.3, comprobamos el LSB de P_c^*, si el valor es 1, se utiliza el procedimiento de extracción de LSB+PVD descrito en la sección 4.2.2, si el valor es 0 se utiliza el procedimiento de extracción de LSB+EMD descrito en la sección 4.2.3. Cuando finalizamos esta parte tenemos los bits del mensaje con nuestra variante de la codificación Elias delta. Para decodificarlo procedemos a recorrer estos bits de manera secuencial, sumamos el número de los L primeros ceros hasta llegar al primer uno, luego recorremos los L dígitos binarios siguientes obteniendo el valor de los N bits que debería ocupar el mensaje oculto y seguidamente recorremos los N siguientes bits que pasaremos a decimal. Así tenemos el mensaje codificado en UTF-8, lo decodificamos y obtenemos el mensaje oculto en la imagen.

4.3. Pruebas unitarias sobre los módulos desarrollados

Para comprobar que nuestros métodos funcionan correctamente en todos los casos, se ha desarrollado una batería de pruebas sobre nuestros módulos, para cada una de las funciones relevantes para la decodificación y codificación. Se ha usado la librería unittest de Python, usando la clase TestCase para cada grupo de pruebas que hemos realizado. Dependiendo de cada módulo se ha realizado una división diferente de los grupos de pruebas, en los próximos apartados explicamos la división por cada módulo.

4.3.1. Módulo methodLSB

La implementación del método LSB explicado en la sección 4.2.1 se ha realizado en el módulo Python ***methodLSB.py***. Además, en este módulo también se ha añadido la funcionalidad para realizar la variante de la codificación y decodificación Elias delta que se utiliza en el método LSB+PVD+EMD explicado en la sección 4.2.4.

Para comprobar esta funcionalidad se ha generado la batería de pruebas que podemos ver en la figura 13, como se puede observar se han generado 5 clases TestCase, que son cada uno de los grupos de pruebas que se ha considerado para este módulo.

- ***MethodLSBTest:*** Contiene los *test* que comprueban la codificación y decodificación de un mensaje de texto a la variante desarrollada de la codificación Elias delta.

- ***MethodLSBChangeBitTest:*** Contiene los *tests* que comprueban todos los casos de modificación de un píxel para el método LSB desarrollado.

- ***MethodLSBDecodeEliasMessageTest:*** Contiene los *tests* que comprueban la decodificación de un mensaje de texto codificado con el método LSB desarrollado en la matriz de píxeles de una estego-imagen.

- ***MethodLSBDecodeMessageToPictureTest:*** Contiene los *tests* que comprueban la decodificación de un mensaje de texto codificado con el método LSB desarrollado en una estego-imagen, incluyendo *tests* sobre la existencia de la imagen.

- ***MethodLSBEncodeMessageToPictureTest:*** Contiene los *tests* que comprueban la codificación de un mensaje de texto de un mensaje de texto codificado con el método LSB desarrollado en una imagen portadora, incluyendo *tests* sobre la existencia de la imagen.

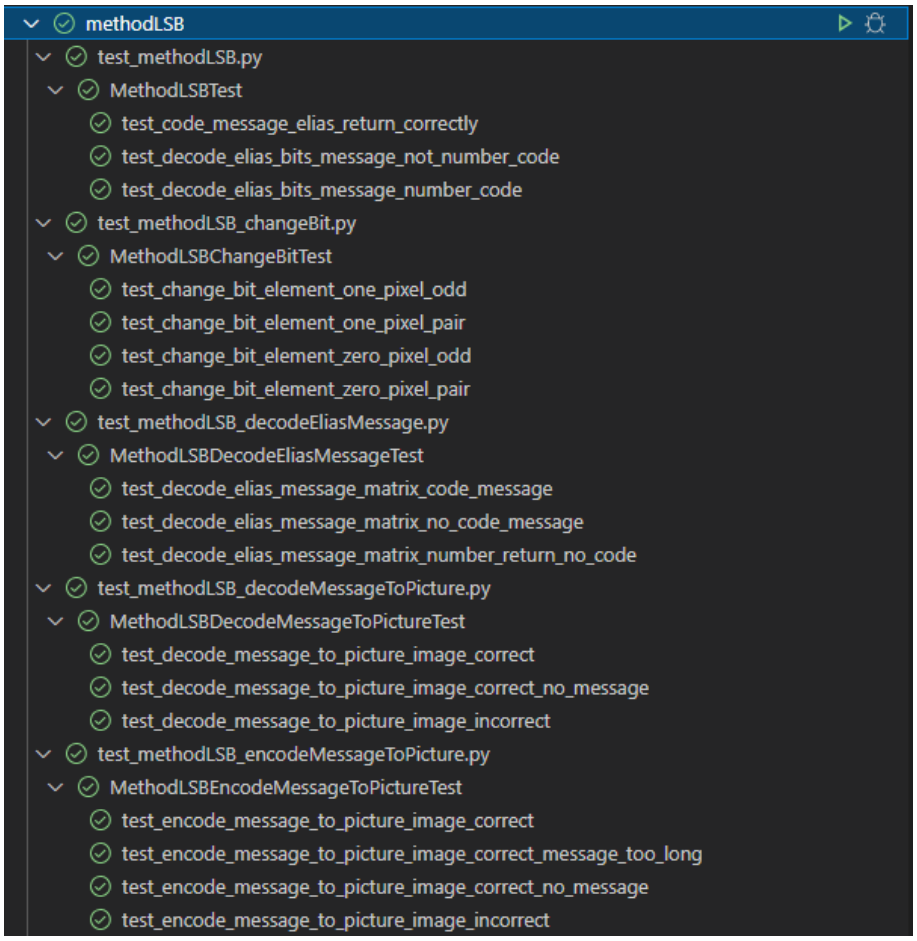

Figura 13 Batería de pruebas implementada para el módulo methodLSB.py

4.3.2. Módulo methodPVD

La implementación del método PVD explicado en la sección 4.2.2 se ha realizado en el módulo Python ***methodPVD.py***. Para comprobar esta funcionalidad se ha generado la batería de pruebas que podemos ver en la figura 14, como se puede observar se han generado 4 clases TestCase, que son cada uno de los grupos de pruebas que se ha considerado para este módulo.

- ***MethodPVDTest:*** Contiene los *tests* que comprueban las funciones de codificación de los píxeles, los bloques y la matriz de píxeles de una estego-imagen en el método PVD desarrollado, ya sean píxeles en los que se tengan que codificar 3 o 4 bits, haya que codificar solo en algunos bits de un boque, y otros casos de uso.

- ***MethodPVDDecodeMessageToPictureTest***: Contiene los *tests* que comprueban la decodificación de un mensaje de texto codificado con el método PVD desarrollado en una estego-imagen, incluyendo *tests* sobre la existencia de la imagen.

- ***MethodPVDEncodeMessageToPictureTest***: Contiene los *tests* que comprueban la codificación de un mensaje de texto de un mensaje de texto codificado con el método PVD desarrollado en una imagen portadora, incluyendo *tests* sobre la existencia de la imagen.

- ***MethodPVDGetbitsTest:*** Contiene los *tests* que comprueban las funciones de decodificación de los píxeles, los bloques y la matriz de píxeles de una estego-imagen en el método PVD desarrollado, ya haya píxeles 3 o 4 bits codificados, solo haya codificados algunos bits de un píxel, y otros casos de uso.

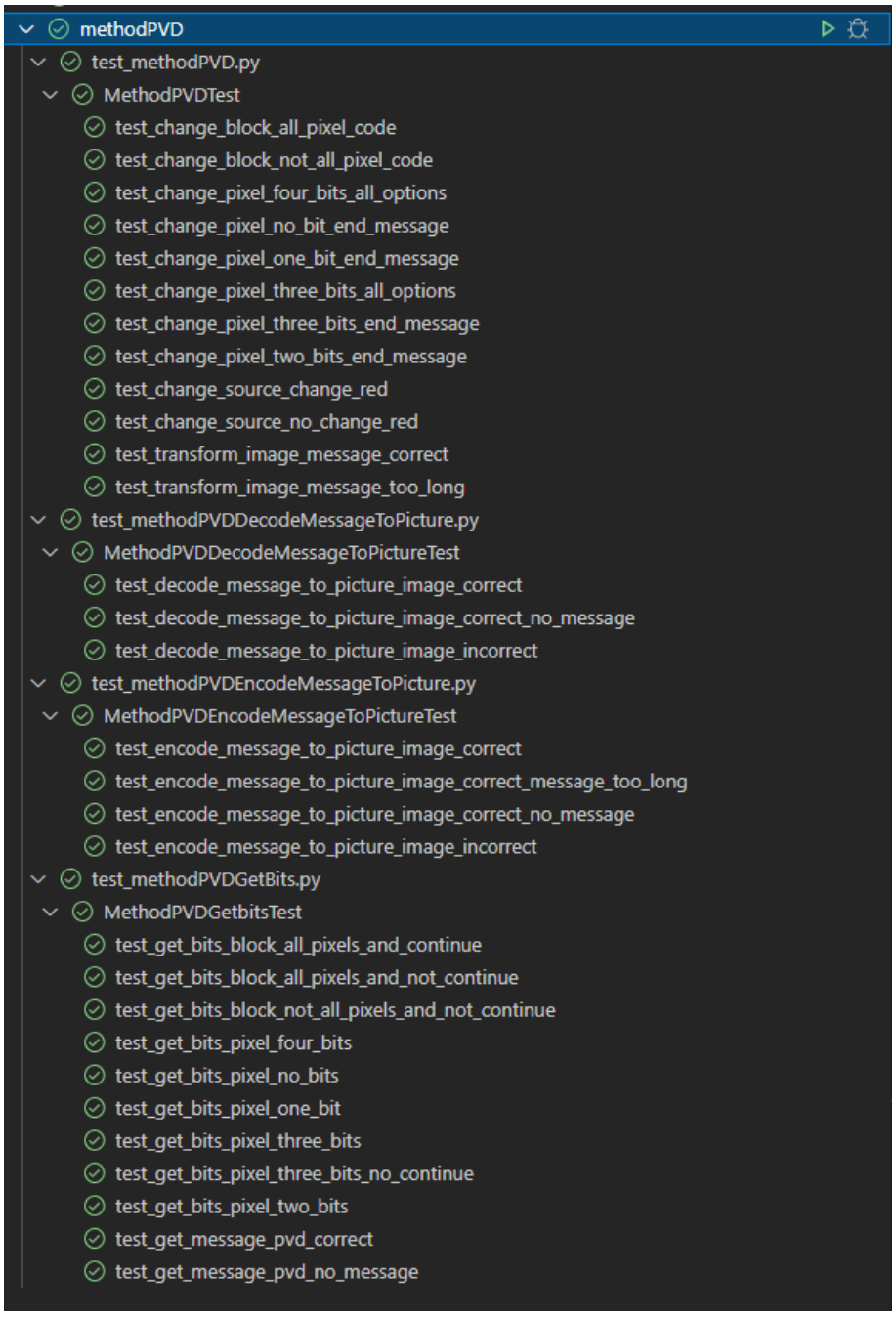

Figura 14 Batería de pruebas para el módulo methodPVD.py

4.3.3. Módulo methodEMD

La implementación del método EMD explicado en la sección 4.2.3 se ha realizado en el módulo Python ***methodEMD.py***. Para comprobar esta funcionalidad se ha generado la batería de pruebas que podemos ver en la figura 15, como se puede observar se han generado 4 clases TestCase, que son cada uno de los grupos de pruebas que se ha considerado para este módulo.

- ***MethodEMDTest:*** Contiene los *tests* que comprueban las funciones de codificación de los píxeles, los bloques y la matriz de píxeles de una estego-imagen en el método EMD desarrollado, ya sean bloques en los que se tengan que codificar todos los píxeles, haya que codificar solo un bit de un píxel, y otros casos de uso.

- ***MethodEMDDecodeMessageToPictureTest***: Contiene los *tests* que comprueban la decodificación de un mensaje de texto codificado con el método EMD desarrollado en una estego-imagen, incluyendo *tests* sobre la existencia de la imagen.

- ***MethodEMDEncodeMessageToPictureTest***: Contiene los *tests* que comprueban la codificación de un mensaje de texto de un mensaje de texto codificado con el método EMD desarrollado en una imagen portadora, incluyendo *tests* sobre la existencia de la imagen.

- ***MethodPVDGetbitsTest:*** Contiene los *tests* que comprueban las funciones de decodificación de los píxeles, los bloques y la matriz de píxeles de una estego-imagen en el método EMD desarrollado, ya haya píxeles con 1 o 2 bits codificados, solo haya codificados algunos píxeles codificados en un bloque, y otros casos de uso.

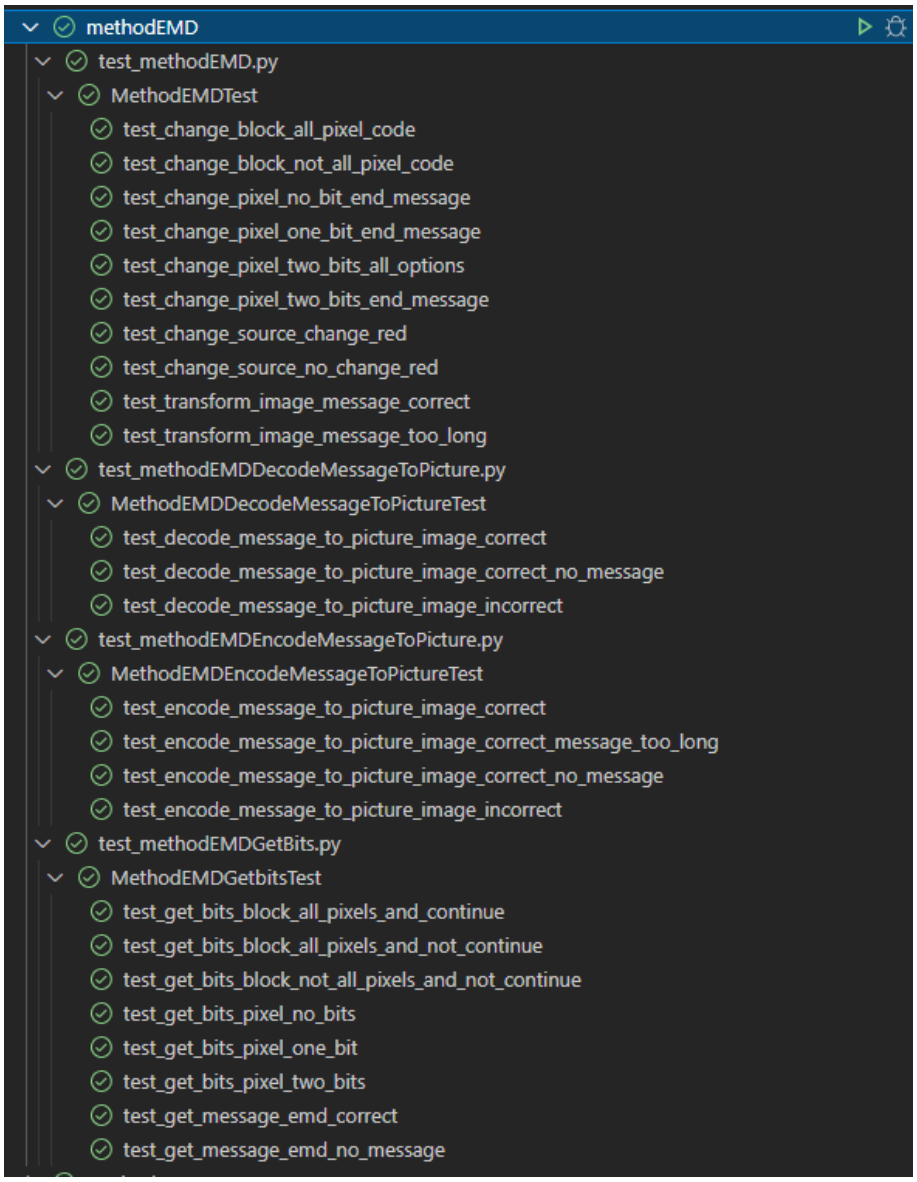

Figura 15 Batería de pruebas para el módulo methodEMD.py

4.3.4. Módulo methodLSBPVDEMD

La implementación del método LSB+PVD+EMD explicado en la sección 4.2.4 se ha realizado en el módulo Python ***methodLSBPVDEMD.py***. Este módulo llama a funciones de los módulos de los otros métodos para aplicar su algoritmo por eso las pruebas se han centrado sobre todo en comprobar que se utilizaba el método PVD y el método EMD en las situaciones adecuadas, podemos ver la batería de pruebas diseñada en la, como se puede observar se han generado 4 clases TestCase, que son cada uno de los grupos de pruebas que se ha considerado para este módulo.

- ***MethodLSBPVDEMDTest:*** Contiene los *tests* que comprueban que se codifica correctamente un bloque con el método PVD o con el método EMD.

- ***MethodLSBPVDEMDDecodeMessageToPictureTest***: Contiene los *tests* que comprueban la decodificación de un mensaje de texto codificado con el método LSD+PVD+EMD desarrollado en una estego-imagen, incluyendo *tests* sobre la existencia de la imagen.

- ***MethodLSBPVDEMDEncodeMessageToPictureTest***: Contiene los *tests* que comprueban la codificación de un mensaje de texto de un mensaje de texto codificado con el método LSD+PVD+EMD desarrollado en una imagen portadora, incluyendo *tests* sobre la existencia de la imagen.

- ***MethodLSBPVDEMDGetMessageTest:*** Contiene los *tests* que comprueban las funciones de decodificación en la matriz de píxeles de una estego-imagen en el método LSB+PVD+EMD desarrollado.

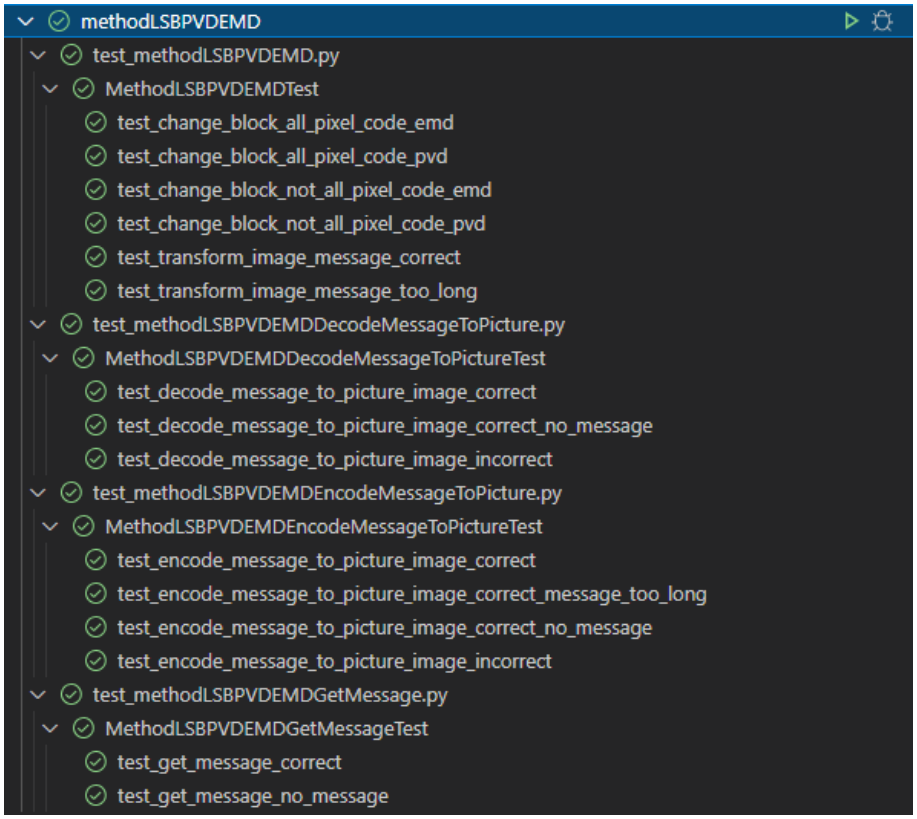

Figura 16 Batería de pruebas para el módulo methodLSBPVDEMD.py

5. Estegoanálisis y resultados del desarrollo

En la sección 4 se han explicado las técnicas que se han desarrollado y se han comentado las pruebas unitarias que se han realizado sobre la implementación, pero no se han analizado los resultados de estas técnicas. En esta sección expondremos el estegoanálisis por par de valores o PoV (3.2.2) que hemos aplicado sobre los métodos desarrollados. Además, estos han sido sometidos a pruebas de esfuerzo, calculando el tiempo de codificación y decodificación con distintos tamaños de imágenes y porcentaje de píxeles codificados en la estego-imagen, que también exponemos en esta sección.

5.1. Estegoanálisis por par de valores aplicado

El estegoanálisis que se ha realizado sobre nuestras técnicas esteganográficas ha sido el de PoV. En los objetivos del proyecto no estaba el de implementar una técnica de estegoanálisis, por ese motivo se ha buscado alguna herramienta o técnica implementada para llevarlo a cabo, en concreto se ha utilizado la implementación de Jesús Díaz [24]. El estegoanálisis por par de valores realizado lo que nos indica es la probabilidad de que una imagen tenga información oculta al analizar una de sus coordenadas de color, rojo, verde o azul, con porcentajes altos por encima del 30% se puede casi asegurar que hay información oculta, porque al someter al estegoanálisis a imágenes sin mensaje oculto a veces hemos obtenido en alguna de las componentes porcentajes del 27% o 28%, lo que nos indica que esos porcentajes no son del todo concluyentes. Para realizarlo se han generado mensajes de texto aleatorios, con los caracteres del alfabeto latino y los caracteres numéricos, de diferentes tamaños que se han calculado para que aproximadamente en cada técnica ocupen un determinado porcentaje de píxeles en la estego-imagen después de realizar la codificación. Después a las estego-imágenes generadas a partir de ocultar estos mensajes con las distintas técnicas se las ha sometido al estegoanálisis por par de valores.

Los resultados obtenidos se han organizado en tablas por el porcentaje de píxeles codificados: un 2% (figura 17), un 25% (figura 18), un 75% (figura 19) y un 98%(figura 20). Lo que se puede ver con estos resultados es que para las imágenes de alta resolución el estegoanálisis por par de valores no es capaz de detectar que la imagen ha sido modificada, sin importar el porcentaje de píxeles modificados en la imagen. Tan solo para las imágenes de menor tamaño que se han usado 150 x 100 píxeles, se

ha podido observar que en el método LSB aumenta la probabilidad de que haya información oculta cuanto más aumenta el porcentaje de píxeles codificados, llegando a casi el 92 % cuando se ha codificado el 98% de los píxeles. Para el resto de las técnicas solo podemos decir que resisten el estegoanálisis por par de valores ya que no hay un patrón reconocible en los resultados a medida que se aumenta la cantidad de píxeles modificados. Los resultados mostrados en las tablas son en imágenes BMP, pero también se ha probado con otros formatos de imagen como PNG y los resultados no varían significativamente.

Algoritmo	Tamaño imagen	Probabilidad PoV para Rojo	Probabilidad PoV para Verde	Probabilidad PoV para Azul
LSB	150x100 p	0,2038259	0,2723042	0,0023822
PVD	150x100 p	0,1309821	0,2709153	0,0031997
EMD	150x100 p	0,2307192	0,2754649	0,0010042
LSB+PVD+EMD	150x100 p	0,2722542	0,2830069	0,0010319
LSB	450x300 p	2,050593 e-37	4,355951 e-68	9,703187 e-40
PVD	450x300 p	1,178851 e-39	3,166324 e-69	1,553027 e-36
EMD	450x300 p	1,805391 e-35	3,256066 e-69	5,814401 e-45
LSB+PVD+EMD	450x300 p	1,583864 e-35	3,115405 e-69	1,532359 e-43
LSB	1000x666 p	1,92087 e-254	0,0	0,0
PVD	1000x666 p	6,12175 e-262	0,0	0,0
EMD	1000x666 p	4,30399 e-242	0,0	0,0
LSB+PVD+EMD	1000x666 p	2,42875 e-244	0,0	0,0
LSB	2048x1364 p	0,0	0,0	0,0
PVD	2048x1364 p	0,0	0,0	0,0
EMD	2048x1364 p	0,0	0,0	0,0
LSB+PVD+EMD	2048x1364 p	0,0	0,0	0,0

Figura 17 Tabla de resultados de estegoanálisis PoV con imágenes BMP con un porcentaje de ocultación del 2% de los píxeles de la imagen para los métodos implementados.

Algoritmo	Tamaño imagen	Probabilidad PoV para Rojo	Probabilidad PoV para Verde	Probabilidad PoV para Azul
LSB	150x100 p	0,4205858	0,12050315	0,0056749
PVD	150x100 p	9,2398508 e-6	0,26825780	7,162007 e-57
EMD	150x100 p	0,0022112	0,27533949	4,152924 e-78
LSB+PVD+EMD	150x100 p	0,0334682	0,28332845	5,586068 e-57
LSB	450x300 p	4,334438 e-21	1,877558 e-52	6,853932 e-40
PVD	450x300 p	3,22747 e-126	3,116885 e-69	0,0
EMD	450x300 p	1,722335 e-49	3,115405 e-69	0,0
LSB+PVD+EMD	450x300 p	3,24669 e-169	0,0	0,0
LSB	1000x666 p	0,0	0,0	0,0
PVD	1000x666 p	0,0	0,0	0,0
EMD	1000x666 p	0,0	0,0	0,0
LSB+PVD+EMD	1000x666 p	0,0	0,0	0,0
LSB	2048x1364 p	0,0	0,0	0,0
PVD	2048x1364 p	0,0	0,0	0,0
EMD	2048x1364 p	0,0	0,0	0,0
LSB+PVD+EMD	2048x1364 p	0,0	0,0	0,0

Figura 18 Tabla de resultados de estegoanálisis PoV con imágenes BMP con un porcentaje de ocultación del 25% de los píxeles de la imagen para los métodos implementados.

Algoritmo	Tamaño imagen	Probabilidad PoV para Rojo	Probabilidad PoV para Verde	Probabilidad PoV para Azul
LSB	150x100 p	0,8202356	1,9077090 e-6	0.5576841
PVD	150x100 p	6,787046 e-66	0,28693532	0,0
EMD	150x100 p	1,25997 e-137	0,28238161	0,0
LSB+PVD+EMD	150x100 p	1,614454 e-48	0,28076725	2,35019 e-260
LSB	450x300 p	1,3662607 e-6	3,379440 e-17	2,648861 e-34
PVD	450x300 p	0,0	3,294318 e-69	0,0
EMD	450x300 p	0,0	3,115405 e-69	0,0
LSB+PVD+EMD	450x300 p	0,0	3,115405 e-69	0,0
LSB	1000x666 p	3,212764 e-93	2,15732 e-187	0,0
PVD	1000x666 p	0,0	0,0	0,0
EMD	1000x666 p	0,0	0,0	0,0
LSB+PVD+EMD	1000x666 p	0,0	0,0	0,0
LSB	2048x1364 p	0,0	0,0	0,0
PVD	2048x1364 p	0,0	0,0	0,0
EMD	2048x1364 p	0,0	0,0	0,0
LSB+PVD+EMD	2048x1364 p	0,0	0,0	0,0

Figura 19 Tabla de resultados de estegoanálisis PoV con imágenes BMP con un porcentaje de ocultación del 75% de los píxeles de la imagen para los métodos implementados

Algoritmo	Tamaño imagen	Probabilidad PoV para Rojo	Probabilidad PoV para Verde	Probabilidad PoV para Azul
LSB	150x100 p	0,9199183	0,92340948	0.0436464
PVD	150x100 p	2,58138 e-101	0,27946118	0,0
EMD	150x100 p	3,37187 e-258	0,28977558	0,0
LSB+PVD+EMD	150x100 p	1,32142 e-177	0,28623056	0,0
LSB	450x300 p	4,2996125 e-7	2,048411 e-17	9,496215 e-17
PVD	450x300 p	0,0	3,115405 e-69	0,0
EMD	450x300 p	0,0	3,115405 e-69	0,0
LSB+PVD+EMD	450x300 p	0,0	4,015444 e-69	0,0
LSB	1000x666 p	1,05360 e-136	2,55645 e-175	0,0
PVD	1000x666 p	0,0	0,0	0,0
EMD	1000x666 p	0,0	0,0	0,0
LSB+PVD+EMD	1000x666 p	0,0	0,0	0,0
LSB	2048x1364 p	0,0	0,0	0,0
PVD	2048x1364 p	0,0	0,0	0,0
EMD	2048x1364 p	0,0	0,0	0,0
LSB+PVD+EMD	2048x1364 p	0,0	0,0	0,0

Figura 20 Tabla de resultados de estegoanálisis PoV con imágenes BMP con un porcentaje de ocultación del 98% de los píxeles de la imagen para los métodos implementados

5.2. Resultados de pruebas de esfuerzo

La realización del propio estegoanálisis, al tener que generar imágenes en distintos tamaños con distintos porcentajes de píxeles codificados, era una prueba de esfuerzo en sí misma. Al calcular el tiempo que tarda en ejecutarse tanto la codificación como la decodificación y comprobar que se obtiene correctamente el mensaje oculto, se tendría una prueba de esfuerzo completa. Los resultados obtenidos ante esta prueba de esfuerzo se han organizado en tablas por el porcentaje de píxeles codificados: un 2% (figura 21), un 25% (figura 22), un 75% (figura 23) y un 98%(figura 24). Gracias a estas pruebas se han podido ver varios problemas en las técnicas desarrolladas.

El primer problema que podemos ver es el tiempo de ejecución de la decodificación para los métodos PVD, EMD y LSB+PVD+EMD cuando hay grandes cantidades de datos. En la tabla que muestra la codificación al 25% de la imagen en la figura 22, ya se puede observar que el tiempo de decodificación del método PVD para imágenes de alta resolución 2048x1364 píxeles son 222 segundos aproximadamente, lo cual ya es un tiempo excesivo sobre todo si lo comparamos con la codificación que son unos 3.6 segundos aproximadamente, los métodos EMD y LSB+PVD+EMD también dan tiempos relativamente altos 40 y 28 segundos respectivamente. Si ya vamos a la figura 24 con un porcentaje de codificación del 98% de una imagen de 2048x1364, los tiempos de ejecución son de 4.571 segundos para PVD y en torno a los 3.000 para los métodos EMD y LSB+PVD+EMD. Ante estos resultados hemos ejecutado un perfilador de la librería estándar de Python sobre nuestros algoritmos de decodificación para ver en qué funciones se aumentaba el tiempo de ejecución. Lo que nos ha indicado es que en las funciones que van añadiendo los bits de datos a la variable se salida es donde está el problema. Esta situación empieza a darse cuanto se intenta codificar información por encima de los 180.000 caracteres, que ya es una cantidad importante de información para ocultar, en este punto se tarda en torno a los 3 minutos en decodificar el mensaje. Pero si tenemos en cuenta, por ejemplo, que la obra "El ingenioso hidalgo don Quijote de la Mancha" tiene 2.034.611 de caracteres contando espacios en blanco y signos de puntuación, se puede codificarla entre 15 y 20 imágenes de 1000x666 tardando en la decodificación entre 20 o 30 segundos por imagen, lo que parece una circunstancia más aceptable.

El segundo problema que se ha podido observar es que en algunas ocasiones no se consigue decodificar el mensaje codificado para el método PVD e indirectamente podemos ver que también afecta al método LSB+PVD+EMD, ya que usa el primero en su ejecución. Está pasando cuando se intenta codificar el 75% de la imagen en adelante para imágenes de 450x300 y de 1000x666. Las deducciones que hemos hecho es que en alguna combinación de caracteres con algún color específico hace que la codificación o decodificación no sea correcta, pero por ahora no se ha encontrado la causa concreta de este problema.

Algoritmo	Tipo	Tamaño imagen	Tiempo Ejecución	Tamaño Mensaje
LSB	codificación	450x300 p	0.08918	371
LSB	decodificación	450x300 p	0.01396	371
PVD	codificación	450x300 p	0.05611	371
PVD	decodificación	450x300 p	0.01895	371
EMD	codificación	450x300 p	0.06309	248
EMD	decodificación	450x300 p	0.01565	248
LSB+PVD+EMD	codificación	450x300 p	0.06822	248
LSB+PVD+EMD	decodificación	450x300 p	0.01546	248
LSB	codificación	1000x666 p	0.14469	1862
LSB	decodificación	1000x666 p	0.02792	1862
PVD	codificación	1000x666 p	0.13772	1862
PVD	decodificación	1000x666 p	0.03690	1862
EMD	codificación	1000x666 p	0.12563	1242
EMD	decodificación	1000x666 p	0.05957	1242
LSB+PVD+EMD	codificación	1000x666 p	0.13795	1242
LSB+PVD+EMD	decodificación	1000x666 p	0.03889	1242
LSB	codificación	2048x1364 p	0.44683	7820
LSB	decodificación	2048x1364 p	0.34226	7820
PVD	codificación	2048x1364 p	0.63223	7820
PVD	decodificación	2048x1364 p	0.25306	7820
EMD	codificación	2048x1364 p	0.48162	5213
EMD	decodificación	2048x1364 p	0.34058	5213
LSB+PVD+EMD	codificación	2048x1364 p	0.56183	5213
LSB+PVD+EMD	decodificación	2048x1364 p	0.16437	5213

Figura 21 Tabla que mide el tiempo de ejecución de los algoritmos implementados para codificación del 2% de los píxeles de la imagen. El tiempo de ejecución en segundos y el tamaño del mensaje en caracteres.

Algoritmo	Tipo	Tamaño imagen	Tiempo Ejecución	Tamaño Mensaje
LSB	codificación	450x300 p	0.20345	9335
LSB	decodificación	450x300 p	0.07051	9335
PVD	codificación	450x300 p	0.34343	9335
PVD	decodificación	450x300 p	0.19748	9335
EMD	codificación	450x300 p	0.29974	6223
EMD	decodificación	450x300 p	0.29974	6223
LSB+PVD+EMD	codificación	450x300 p	0.23537	6223
LSB+PVD+EMD	decodificación	450x300 p	0.17291	6223
LSB	codificación	1000x666 p	0.81347	46714
LSB	decodificación	1000x666 p	0.44879	46714
PVD	codificación	1000x666 p	1.06804	46714
PVD	decodificación	1000x666 p	3.05725	46714
EMD	codificación	1000x666 p	0.70921	31143
EMD	decodificación	1000x666 p	1.28776	31143
LSB+PVD+EMD	codificación	1000x666 p	1.29890	31143
LSB+PVD+EMD	decodificación	1000x666 p	1.88671	31143
LSB	codificación	2048x1364 p	2.84735	196189
LSB	decodificación	2048x1364 p	1.73033	196189
PVD	codificación	2048x1364 p	3.59571	196189
PVD	decodificación	2048x1364 p	222.584	196189
EMD	codificación	2048x1364 p	5.00461	130992
EMD	decodificación	2048x1364 p	40.0587	130992
LSB+PVD+EMD	codificación	2048x1364 p	4.34971	130992
LSB+PVD+EMD	decodificación	2048x1364 p	27.8355	130992

Figura 22 Tabla que mide el tiempo de ejecución de los algoritmos implementados para codificación del 25% de los píxeles de la imagen. El tiempo de ejecución en segundos y el tamaño del mensaje en caracteres.

Algoritmo	Tipo	Tamaño imagen	Tiempo Ejecución	Tamaño Mensaje
LSB	codificación	450x300 p	0.56647	28002
LSB	decodificación	450x300 p	0.32213	28002
PVD	codificación	450x300 p	0.77891	28002
PVD	decodificación	450x300 p	0.88364	0
EMD	codificación	450x300 p	0.47978	18669
EMD	decodificación	450x300 p	0.58440	18669
LSB+PVD+EMD	codificación	450x300 p	0.62233	18669
LSB+PVD+EMD	decodificación	450x300 p	0.63330	18669
LSB	codificación	1000x666 p	2.16720	140130
LSB	decodificación	1000x666 p	1.00432	140130
PVD	codificación	1000x666 p	2.67041	140130
PVD	decodificación	1000x666 p	39.8942	0
EMD	codificación	1000x666 p	2.23502	93420
EMD	decodificación	1000x666 p	9.73894	93420
LSB+PVD+EMD	codificación	1000x666 p	2.53620	93420
LSB+PVD+EMD	decodificación	1000x666 p	8.48632	0
LSB	codificación	2048x1364 p	6.15499	588515
LSB	decodificación	2048x1364 p	10.0734	588515
PVD	codificación	2048x1364 p	10.5057	588515
PVD	decodificación	2048x1364 p	3005.42	588515
EMD	codificación	2048x1364 p	6.32856	392344
EMD	decodificación	2048x1364 p	1594.03	392344
LSB+PVD+EMD	codificación	2048x1364 p	8.49613	392344
LSB+PVD+EMD	decodificación	2048x1364 p	1346.17	392344

Figura 23 Tabla que mide el tiempo de ejecución de los algoritmos implementados para codificación del 75% de los píxeles de la imagen. El tiempo de ejecución en segundos y el tamaño del mensaje en caracteres.

Algoritmo	Tipo	Tamaño imagen	Tiempo Ejecución	Tamaño Mensaje
LSB	codificación	450x300 p	0.46035	36591
LSB	decodificación	450x300 p	0.53707	36591
PVD	codificación	450x300 p	0.91138	36591
PVD	decodificación	450x300 p	0.83651	0
EMD	codificación	450x300 p	0.44133	24394
EMD	decodificación	450x300 p	0.62362	24394
LSB+PVD+EMD	codificación	450x300 p	0.44645	24394
LSB+PVD+EMD	decodificación	450x300 p	0.63273	0
LSB	codificación	1000x666 p	2.11163	183107
LSB	decodificación	1000x666 p	1.13370	183107
PVD	codificación	1000x666 p	2.73004	183107
PVD	decodificación	1000x666 p	133.475	0
EMD	codificación	1000x666 p	2.20019	122072
EMD	decodificación	1000x666 p	20.5118	122072
LSB+PVD+EMD	codificación	1000x666 p	2.74735	122072
LSB+PVD+EMD	decodificación	1000x666 p	15.5499	122072
LSB	codificación	2048x1364 p	7.07394	769011
LSB	decodificación	2048x1364 p	7.68615	769011
PVD	codificación	2048x1364 p	11.1198	769011
PVD	decodificación	2048x1364 p	4575.41	769011
EMD	codificación	2048x1364 p	8.30631	512674
EMD	decodificación	2048x1364 p	2891.36	512674
LSB+PVD+EMD	codificación	2048x1364 p	10.5459	512674
LSB+PVD+EMD	decodificación	2048x1364 p	3206.91	512674

Figura 24 Tabla que mide el tiempo de ejecución de los algoritmos implementados para codificación del 98% de los píxeles de la imagen. El tiempo de ejecución en segundos y el tamaño del mensaje en caracteres.

6. Conclusiones

El principal objetivo de este proyecto era realizar una biblioteca en Pyhton de métodos o técnicas esteganográficas que ocultaran un mensaje de texto en una imagen. Previamente había que hacer una investigación sobre la esteganografía y el estegoanálisis. Se puede decir que todos los objetivos se han cumplido.

- Se han implementado cuatro módulos en Pyhton, que aplican cada uno una de las técnicas de esteganografía explicadas en la sección 4, los módulos desarrollados son: **methodLSB.py**, **methodPVD.py**, **methodEMD.py** y **methodLSBPVDEMD.py.** Se ha comprobado el funcionamiento correcto de todos ellos, salvo por los apuntes hechos en la sección 5.2.

- Los resultados de la investigación sobre la esteganografía y el estegoanálisis se pueden ver en la sección 3 de este documento.

Aunque los objetivos del proyecto se hayan alcanzado, hay algunas funcionalidades que podrían mejorarse o añadirse, ya que el campo de la esteganografía es muy extenso. También hay algunas observaciones que se han hecho durante el desarrollo del proyecto que son dignas de mencionar. En los siguientes apartados comentamos estas observaciones y el trabajo futuro que se puede realizar tomando nuestro proyecto como punto de partida.

6.1. Observaciones

La elección de Python para la implementación de este proyecto no fue trivial, a parte del propio interés del autor en realizar un proyecto en Python. Antes de conocer en detalle la esteganografía, se hizo la suposición de que se necesitarían operaciones matemáticas complejas para poder implementar algunas de las técnicas. Es conocido que Python contiene una gran variedad de librerías que realizan operaciones matemáticas complejas de manera muy optimizada y con muchos años de pruebas y desarrollo. Este punto fue un factor muy importante para acabar utilizando Python como lenguaje de programación para desarrollar esta biblioteca.

Otra observación que se ha realizado durante el proyecto es la importancia de los test unitarios en una aplicación, y este desarrollo es una prueba fehaciente de ello. Se consiguió descubrir un error de gran importancia en la implementación que las pruebas de desarrollo no fueron capaces de ver, gracias al desarrollo de los test unitarios. En concreto se detectó que se estaban realizando operaciones con enteros de 8 bits como si fueran enteros, y cuando se salían de rango, ya fuera un número

negativo o superior a 255 obteníamos valores no deseados que provocaban que la implementación no funcionase correctamente. Como las pruebas de desarrollo se lanzaban directamente sobre imágenes comunes, no encontrábamos los casos donde se producían los valores que se salían de rango. Pero cuando tuvimos una batería de test con las distintas posibilidades se consiguió detectar y corregir un error que se producía de forma sistemática.

6.2. Trabajo futuro

Durante las diferentes fases del desarrollo de este proyecto, han surgido posibles funcionalidades que podrían mejorar y completar nuestra biblioteca. Las exponemos en los siguientes puntos:

- Encontrar los casos concretos donde se dan los problemas del método PVD vistos en las pruebas de esfuerzo y comentados en la sección 5.2 y aportar una solución a ellos.

- Buscar una solución para mejorar el rendimiento de los algoritmos de decodificación PVD, EMD y LSB+PVD+EMD con mensajes de texto muy grandes (si es posible).

- Probar otros formatos de imagen aparte de PNG y BMP y modificar el desarrollo para que soporte las imágenes JPG si es posible, y si no desarrollar métodos alternativos para los formatos de imagen que no se soporten.

- Desarrollar una versión de los métodos PVD, EMD y LSB+PVD+EMD para tamaños de bloque 3x3 en base al trabajo de [22], o parametrizar nuestros métodos para que puedan realizarse con distintos tamaños.

- Desarrollar una versión de los métodos PVD, y LSB+PVD+EMD usando como tabla de rangos la tabla 2 o parametrizar nuestros métodos para que puedan realizarse con la tabla 1 y la tabla 2.

- Desarrollar una versión parametrizada del método LSB, donde podamos elegir los bits por coordenada que se ocultan.

- Desarrollar versiones de los métodos desarrollados más precisas con respecto al planteamiento teórico y hacer comparativas entre las diferentes versiones.

- Desarrollar versiones de nuestras técnicas que permitan a parte de guardar mensajes de texto otros tipos de información digital, como por ejemplo otras imágenes.

- Desarrollar nuevos métodos para ampliar nuestra librería, ya entren en la misma categoría, pero preferiblemente de otras, como métodos en el dominio de la frecuencia, de transformación de dominios, de generación de portadores, técnicas hibridas ...

- Investigar y desarrollar métodos para otros tipos de archivo como audio, video o documentos, ampliando nuestra librería no solo a técnicas esteganográficas en imágenes sino a todo tipo de archivos digitales.

- Desarrollar una nueva aplicación para acceder a nuestras técnicas ya sea de comandos o con una interfaz de usuario pero que sea más amigable y de mayor facilidad de uso.

- Desarrollar un API REST que permita acceder mediante diferentes peticiones a nuestros métodos y poder incorporarlos a diferentes aplicaciones.

- Someter a otras herramientas de estegoanálisis a los métodos desarrollados, permitiéndonos hacer una mejor comparativa entre ellos.

- Desarrollar técnicas propias de estegoanálisis sobre todo las referenciadas en [22], para comprobar sus tesis sobre las técnicas desarrolladas.

7. Conclusions

The main objective of this project was to create a library in Python of steganographic methods or techniques that would hide a text message in an image. Previously, research had to be conducted on steganography and steganalysis. It can be said that all the objectives have been met.

- The following four modules have been implemented in Python: **methodLSB.py**, **methodPVD.py**, **methodEMD.py** and **methodLSBPVDEMD.py**, each one applying a steganography technique explained in section 4. The correct behaviour all of them has been checked, except for remarks made in section 5.2.

- The results of the research on steganography and steganalysis can be observed in section 3 of this document.

Although the objectives of the project have been achieved, there are some functionalities that could be improved or added, since the field of steganography is very extensive. There are also some observations that have been made during the development of the project that are worth mentioning. They will be discussed in the following sections.

7.1. Notes

The choice of Python for the implementation of this project was not trivial. Apart from the author's own interest in developing a Python project, we made the assumption that complex mathematical operations would be needed to be able to implement some of the techniques. Python is known to contain a great variety of libraries that perform complex mathematical operations in a highly optimized way and a long track record in testing and development. This point was a very important factor to select Python as a programming language to develop this library.

Another observation that has been made during the project is the importance of unit tests in an application, and this development is a reliable proof of it. In fact, it was thanks to the implementation of unit tests that we discovered a great importance error in programming, that development tests were not able to identify. Specifically, it was detected that operations with 8-bit integers were carried out as if they were integers, and when they went out of range, whether it was a negative number or greater than 255, we obtained unwanted values that made the implementation not work correctly. As the development tests were done directly on common images, we could not find the cases where the values were out of range. However, since we had

a battery of tests with the different possibilities, we were able to identify and correct an error that was occurring in a systematic way.

7.2. Future work

During the different phases of the development of this project, possible functionalities have emerged that could improve and complete this library. We will expose them in the following points:

- Find the specific cases where the problems of the PVD method seen in the stress tests and discussed in section 5.2 happen and provide a solution to them.

- Find a solution to improve the performance of the decode algorithms of the PVD, EMD and LSB + PVD + EMD with very large text messages (if possible).

- Test image formats other than PNG and BMP and modify the development to support JPG images if possible, and if not develop alternative methods for not supported format images.

- Develop a version of the PVD, EMD and LSB + PVD + EMD methods for 3x3 blocks based on the work of [22], or parameterize our methods so that they can be used with different type of block.

- Develop a version of the PVD, and LSB + PVD + EMD methods using Tabla 2 as a table of ranges or parameterize our methods so that they can be used with Tabla 1 or Tabla 2.

- Develop a parameterized version of the LSB method, where we can choose the bits per coordinate that are hidden.

- Develop more accurate versions of the developed methods with respect to the theoretical approach and make comparisons between the different versions.

- Develop versions of our techniques that allow us to hide other type of digital information, apart from saving text messages, such as other images.

- Develop new methods to expand our library, whether they are the same category, but preferably others, such as methods in the frequency domain, domain transformation, covered generation, hybrid techniques, ...

- Research and develop methods for other types of files such as audio, video, or documents, expanding our library not only to steganographic techniques in images but also types of digital files.

- Develop a new application to access our techniques either with commands or with a user interface, but one that is friendlier and easier to use.

- Develop a REST API that allows access to our methods through different requests and to be able to incorporate them into different applications.

- Use other steganalysis tools to attack the developed methods, which allows us to make a better comparison among them.

- Develop own steganalysis techniques, especially those referenced in [22], to check their conclusions on the developed techniques.

Bibliografía

[1] J. Fridrich, "Steganography throughout history," en *Steganography in Digital Media: Principles, Algorithms, and Applications*: Cambridge University Press, 2010, pp. 3–7.

[2] A. Cheddad, J. Condell, K. Curran y P. Mc Kevitt, "Digital image steganography: Survey and analysis of current methods," *Signal Processing*, Marzo.2010, pp. 728–752, Marzo.2010.

[3] G. J. Simmons, "The Prisoners' Problem and the Subliminal Channel," presentado en *In Advances in Cryptology*, Santa Barbara, 1983, pp. 51–67.

[4] C. Kurak y J. McHugh, "A Cautionary Note On Image Downgrading," presentado en *Proceedings Eighth Annual Computer Security Application Conference*, San Antonio Texas, 1992.

[5] Z. K. Al-Ani, A. A. Zaidan, B. B. Zaidan y H. O. Alanazi, "Overview: Main Fundamentals for Steganography," *Journal of Computing*, 2010, pp. 158–165, 2010.

[6] P. Alvarez, "Using Extended File Information (EXIF) File Headers in Digital Evidence Analysis," *International Journal of Digital Evidence*, 2004, pp. 1–5, 2004.

[7] N. F. Johnson y S. Jajodia, "Exploring steganography: seeing the unseen," *IEEE Computer*, 1998, pp. 26–34, 1998.

[8] Z. Li, X. Chen, X. Pan y X. Zeng, "Lossless Data Hiding Scheme Based on Adjacent Pixel Difference," presentado en *2009 International Conference on Computer Engineering and Technology*, 2009, pp. 145–152.

[9] P. Tsai, Y. C. Hu y H. L. Yeh, "Reversible image hiding scheme using predictive coding and histogram shifting," *Signal Process*, 2009, pp. 1129–1143, 2009.

[10] V. M. Potdar, S. Han y E. Chang, "Fingerpnnted Secret Sharing Steganography for Robustness against Image Cropping Attacks," presentado en *IEEE Third International Conference on Industrial Informatics*, Perth, 2005.

[11] W. H. Chen, C. H. Smith y S. C. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform," *IEEE TRANSACTIONS ON COMMUNICATIONS*, 1977, pp. 1004–1009, 1977.

[12] X. Li y J. Wang, "A steganographic method based upon JPEG and particle swarm optimization algorithm," *Information Sciences*, 2007, pp. 3099–3109, 2007.

[13] C. C. Chang, T. S. Chen y L. Z. Chung, "A steganographic method based upon JPEG and quantization table modification," *Information Sciences*, 2002, pp. 123–138, 2002.

[14] J. Fridrich, M. Goljan y D. Hogea, "Steganalysis of JPEG Images: Breaking the F5," en *Information Hiding 2002*

[15] P. Wayner, *Disappearing Cryptography 2nd Edition Information Hiding: Steganography & Watermarking*: Morgan Kaufmann, 2002.

[16] A. Cheddad, J. Condell, K. Curran y M. K. Paul, "A secure and improved self-embedding algorithm to combat digital," *Signal Processing*, 2009, pp. 2324–2332, 2009.
[17] R. Chandramouli y K. P. Subbalakshmi, "Current trends in steganalysis: a critical survey," presentado en *ICARCV 2004 8th Control, Automation, Robotics and Vision Conference*, Kunming, 2004, pp. 964–967.
[18] A. Muñoz Muñoz, J. Carracedo Gallardo y S. Sánchez García, "Detection of distributed steganographic information in social networks," presentado en *2008 Euro American Conference on Telematics and Information Systems*, New York, NY, USA, 2008.
[19] G. A. Francia y T. S. Gomez, "Steganography obliterator: an attack on the least significant bits," presentado en *2006 Information Security Curriculum Development*, Kennesaw, Georgia, USA, 2006.
[20] A. Westfeld y A. Pfitzmann, "Attacks on Steganographic Systems," presentado en *International workshop on information hiding*, Dresden, 1999.
[21] P. Elias, "Universal codeword sets and representations of the integers," *IEEE Transactions on Information Theory*, 1975, pp. 194–203, 1975.
[22] A. Pradhan, K. R. Sekhar y G. Swain, "Digital Image Steganography Using LSB Substitution, PVD, and EMD," *Mathematical Problems in Engineering*, 2018, pp. 1–11, 2018.
[23] J. Fridrich, M. Goljan y R. Du, "Reliable detection of lsb steganography in color and grayscale images," presentado en *2001 Workshop on Multimedia and Security: New Challenges*, New York, NY, USA, 2001, pp. 27–30.
[24] J. Díaz. Consultado en 2021. [Online]. Disponible en https://www.incibe-cert.es/blog/esteganografia-y-estegoanalisis-basicos.