Proyecto 02 Esteganografía

Universidad Nacional Autónoma de México

Facultad de Ciencias

Modelado y Programación

Integrantes:
Hannia Laura López Ceballos
Mónica Miranda Mijangos

17 de diciembre de 2021

1. Desarrollo

1.1. Definición del Problema

La estaganografía es un conjunto de técnicas algorítmicas para ocultar un conjunto de datos en otro de forma que los primeros no sean evidentes, permanezcan íntegros y sean recuperables mientras que los segundos, a los que se les denomina portadores, no deben verse alterados de manera evidente.

Un método común es ocultar mensajes de texto claro en archivos de imágenes. Se aprovecha en este caso el hecho de que, con frecuencia, las imágenes no requieren de preservar todos los datos que contienen para ser útiles.

Un método usado para almacenar un archivo de texto en una imagen consiste en usar el bit menos significativo de cada byte de datos de la imagen para guardar un bit de los datos a ocultar. Por supuesto se supone que la cantidad de bytes de datos de la imagen es suficiente para guardar el archivo de texto respectivo. Típicamente cada pixel de una imagen es una terna (o cuarteta si existe un canal alfa para la transparencia) de bytes, uno para especificar el tono del pixel en el canal rojo, otro para el canal verde y un tercero para el canal azul, los cuales toman valores de 0 hasta 255. Si se cambia el valor del bit menos significativo de cada uno de estos canales, el color del pixel será cambiado de manera imperceptible para quien observa a simple vista la imagen, entonces cada pixel puede alojar 3 bytes (recordemos que un byte son 8 bits) de información, uno por cada nivel de color. Es decir, por cada pixel hay, el menos, tres bits útiles para almacenar datos ocultos.

De esta manera, la idea para esconder información en una imagen será descomponer nuestro texto convirtiendolo en bits y escondiendo un bit por cada nivel de color, en el bit menos significativo. Pero ¿cómo recuperamos está infromación? solo debemos hacer el proceso inverso, calculando el valor decimal de cada ocho bits para así, obtener el carácter que le corresponde.

¿Es esto seguro? Sí, de alguna forma. Sin embargo, podemos ser muy predecibles en nuestro proceso de codificación lo cuál hace que nustra información pueda ser fácilmente vulnerada. Es por eso que lo recomendable sería encriptar el texto antes de codificarlo, así quién descubra lo que hemos ocultado, no sabrá cuál es el mensaje original.

1.2. Análisis del Problema

- Requsitos funcionales:
  - ¿Qué debe entregar como salida dado cierto tipo de entrada?

    El programa debe funcionar en dos modalidades, para ocultar (opción **h**) y para develar (opción **u**).

    Para **ocultar** debe recibir la opción h seguido del nombre del archivo que contiene el texto a ocultar, el nombre del archivo de imagen y el nombre con el que guardaremos el archivo de imagen resultante con los datos ocultos.
    Regresará el archivo de imagen con el texto oculto.

    Para **develear** debe recibir la opción u seguido de el nombre del archivo con la imagen que contiene los datos ocultos y el nombre del archivo en el que se guardará el texto develado.
    Y regresará el archivo con el texto claro.
  - ¿Qué se debe llevar a cabo?

    Para ocultar debemos descomponer nuestro texto convirtiendolo en bits y escondiendo un bit por cada nivel de color, en el bit menos significativo. Para develar solo debemos hacer el proceso inverso, calculando el valor decimal de cada ocho bits para así, obtener el carácter que le corresponde.
- Requsitos no funcionales:
  - Eficiencia

    El sistema recibe dos entradas diferentes, sin embargo no es necesario que sea capaz de procesar N transacciones por segundo. Por lo que la funcionalidad del sistema y transacción de negocio no requiere responder al usuario en un cierto límite de tiempo, ya que este no interactua con el mismo.
  - Tolerancia a fallas

    El programa lanza un error cuando le dan una entrada inexistente o un archivo que no contiene datos, es decir, el programa se detiene.
  - Amigabilidad

    La interfaz del programa debe ser amigable para que el usuario pueda meter la entrada junto con el archivo de texto a ocultar.
  - Seguridad

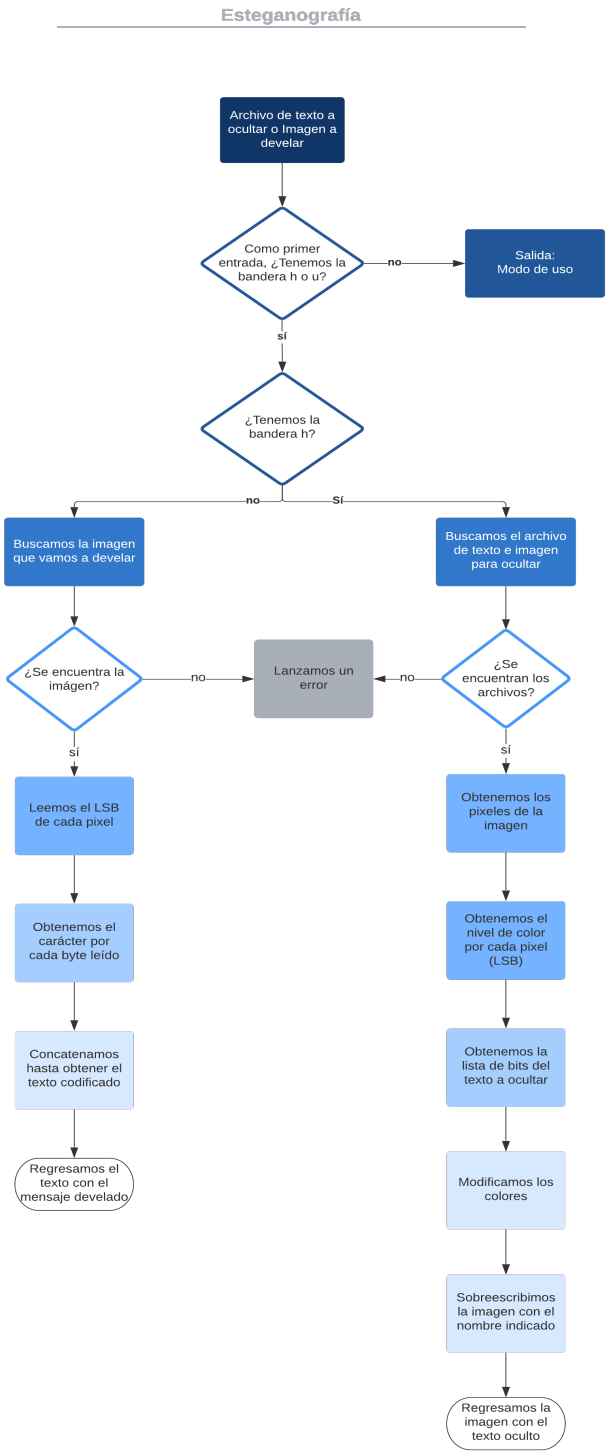
    Se encripta el texto antes de codificarlo, de esta manera, si llega a ser descubierto, no sabrán el mensaje original.
  - Interoperabilidad

    Es la capacidad de los sistemas de información y de los procedimientos a los que éstos dan soporte, de compartir datos y posibilitar el intercambio de información y conocimiento entre ellos, en este caso no hay intercambio de información de un sistema a otro.

1.3. Selección de la mejor alternativa

- Paradigma de programación: Orientado a Objetos.
- Lenguaje: Python.
  Ya que es un lenguaje multiparadigma enfocado en la legibilidad del código, además de ser interpretado, es decir, puede compilar el código fuente original en una forma intermedia más compacta, y después traducir eso al código de máquina; dinámico, las variables pueden tomar valores de distinto tipo, y multiplataforma.
- Herramientas para pre o post procesar datos:
  - Stepic: Es un módulo de Python y una herramienta de línea de comandos para ocultar datos arbitrarios dentro de las imágenes, modificando ligeramente los colores de los píxeles de la imagen para almacenar los datos. En este caso lo utilizaremos únicamente para develar ya que no soporta jpg's y otros formatos de perdida. Solamente vamos a develar cosas que nuestro programa va a codificar los cuales son archivos sin perdida, es decir, válidos para stepic.
  - Pillow: Nos permite procesar con facilidad imágenes con Python ya que soporta una variedad de formatos, incluídos los más utilizados como GIF, JPEG y PNG.
- Frameworks o bibliotecas:
  - ¿Qué tan útiles son? Son muy útiles ya que son amplias y cuentan con gran cantidad de producciones en contenidos, constan de diversos módulos que permiten el acceso de funcionalidades específicas de modo que podemos concentrarnos en lo que realmente nos ocupa.
  - ¿Qué se necesita para usarlas? Únicamente se necesita importarlas con la palabra import seguido del nombre de la biblioteca a utilizar.
  - ¿Están bien documentadas? Si, siguen un estándar oficial de codificación, el cual es PEP-8.
  - ¿Han sido probadas? Si, son programadas y probadas por terceros.
  - ¿Vale la pena usarlas? Si, ya que nosotros como programadores nos queremos evitar la necesidad de repetir en nuestros programas una y otra vez el mismo código, lo que también hace a nuestro programa poco eficiente.
  - ¿Son seguras? Con python podemos cifrar la información, se puede cifrar desde archivos hasta carpetas, con esto se puede mantener una integridad en la información. Existen muchas bibliotecas que nos permiten elegir el algoritmo de cifrado que deseemos utilizar.

1.4. Diagrama de flujo



1.5. Mantenimiento

- ¿Cuánto cobrarías por el proyecto y futuro mantenimiento? Para poder poner un precio al proyecto podemos desglosar una lista de tareas a traves de las cuales se estimara el tiempo que se le dedicara a cada una en horas, hay que tener en cuenta que a cada una de las tareas se le deben agregar los costos que se requerirán para que el sistema funcione, como lo sería:

  - Equipo específico para el proyecto.
  - Servidores en la nube.
  - Dominios
  - Licencias
  - Bases de datos

- Una buena forma de determinar el precio por hora del proyecto es tomando en cuenta la experiencia que se tiene y en todo caso lo que se desea ganar mensualmente. Tomando esto en cuenta, de acuerdo a nuestra experiencia y horas trabajadas, el precio por hora sería de 168.75 MXN lo que nos da un total de 27,000 MXN.

- Mantenimiento: El tipo de mantenimiento que se le puede dar a este proyecto es adaptativo, el cual consiste en la modificación del programa debido a cambios en el entorno en el que se ejecuta, tambien se podría contar con un manejo perfectivo o preventivo en los cuales se mejoran o añaden nuevas funcionalidades así como la mejora de la eficiencia de ejecución, la estructuración de los programas para aumentar su legibilidad o incluir nuevos comentarios.