Universidad Nacional Autónoma de México.

Facultad de Ciencias

Modelado y Programación

---

# Proyecto II

---

Santiago Díaz Ponton-no.Cuenta: 317164402,
Mauricio Guerrero Palomares-no.Cuenta :51716402

17 de diciembre de 2021

1)*Definición del problema*
Lo que se pide en el problema es un tipo de encriptación de datos.
Tenemos que recibir un mensaje y una imagen, y por el método de LSB encriptar el mensaje dentro de la imagen.
Cómo funciona esta forma de encriptación, es tomar el bit menos significante de cada pixel de la imagen, y cambiarlo con cada bit del mensaje que se quiere ocultar.

2)*Análisis del problema*
Para ocultar un mensaje, lo primero que tenemos que realizar es recibir un mensaje a partir de un archivo txt, y guardar esta cadena. Después pasamos a la imagen. Para poder resolver el problema, necesitamos ver la imagen como alguna estructura en la cual se pueda cambiar el valor de cada uno de sus píxeles.
Para develar, es el proceso pero a la inversa. Es decir, ahora en lugar de cambiar los valores los vamos a tomar y a guardar para descifrarlos, finalmente sólo deolvemos este mensaje en un archivo .txt

3)*Selección de la mejor alternativa*

La mejor alternativa para resolver este problema fue usar Python, ya que cuenta con librerias como **NumPy**, que se enfoca más en el análisis numérico, pero cuenta con métodos que justo nos sirven para este problema, como por ejemplo *np.array(list(imagen.getData))* (donde imagen es la imagen que queremos volver en un arreglo). Este método vuelve la imagen en un array de píxeles, donde cada pixel tiene forma RGB ó RGBA dependiendo del formato, y de nuevo, python tiene una librería llamada **PIL**, donde se pueden acceder a las especificaciones de imágenes en su clase *Image*.

Es por esto que Python fue la mejor opción para resolver el problema, aunque volver una cadena a binario fuera extraño (con métodos como *".join([format(ord(i), "08b") for i in mensaje])*, que fue algo complicado de hacer.

4)*Pseudocódigo*

**Clase Esteganografía:**

*Oculta(imagen, mensaje,imagenOculta)*

Este método se encarga de ocultar un mensaje '*mensaje*' dado por el usario. Como funciona es volver una imagen en un array de pixeles (esto gracias a la libreria **numPy**), y luego checamos si la iamgen es tipo RGB o RGBA. Esto es importante, ya que si es tipo RGB sólo recibe 3 paramétros, es decir su composición en tipo Red Green Blue; en cambio si es RGBA su composición es de tipo Red,Green,Blue Alpha.

Después, para tratar de evitar problemas, calculamos el número de pixeles totales en la imagen, luego convertimos nuestro mensaje a binario, (esto con el método format) y luego calculamos cuantos pixeles necesita el mensaje para ser ocultado en la imagen, y si este es mayor que el número de pixeles totales en la imagen manda una excepción.

También pondremos un delimitador. Esto para saber hasta dónde llega el mensaje. En este caso el delimitador es *"$3ncripTAd0"*.

Luego Iteramos sobre el número de pixeles y vamos modificando su bit menos significante con el bit de el mensaje a ocultar.

Finalmente creamos una nueva imagen con el array creado ya con la imagen con sus bits cambiados, y devolvemos esta imagen.

*Devela(imagen)*

Este método funciona de forma inversa a *Oculta*. Este método recibe *imagen*, que es la imagen que contiene un mensaje oculto. Convertimos esta imagen en un array de pixeles con la librería NumPy, y revisamos si la imagen es de tipo RGB o RGBA.

Luego, lo que haremos es ir recorriendo cada pixel de la imagen, esto para tomar el bit menos significante de cada pixel y lo guardaremos.

Finalmente, volvemos esta lista de bits menos significantes de binario a ASCII, y si sí se encuentra un el delimitador *"$3ncripTAd0"*. devolvemos la cadena que se obtuvo.

Luego guardamos esta cadena en una archivo .txt, y lo devolvemos.

*ocultaDevela(self)*

Este método solo realiza preguntas al usuario para recibir los parámetros que los métodos necesitan, y depediendo de lo que necesite el usuario se lleva a cabo.

**Clase manejaArchivo:**

*leeArchivo(self,nombreArchivo)*

Este método se encarga de leer un archivo y regresar lo que tiene adentro.

Primero abre un archivo con el nombre *nombreArchivo* usando el método **open()** de la librería de python. Este método recibe el nombre de un archivo, seguido de una letra representando lo que se quiere hacer, es decir; "w"de write para escribir, r"de read para leer, .ª"de append para adjuntar una cadena a un archivo, etc.

Después, leemos el archivo y guardamos lo que contiene en una cadena. Finalmente regresamos esta cadena.

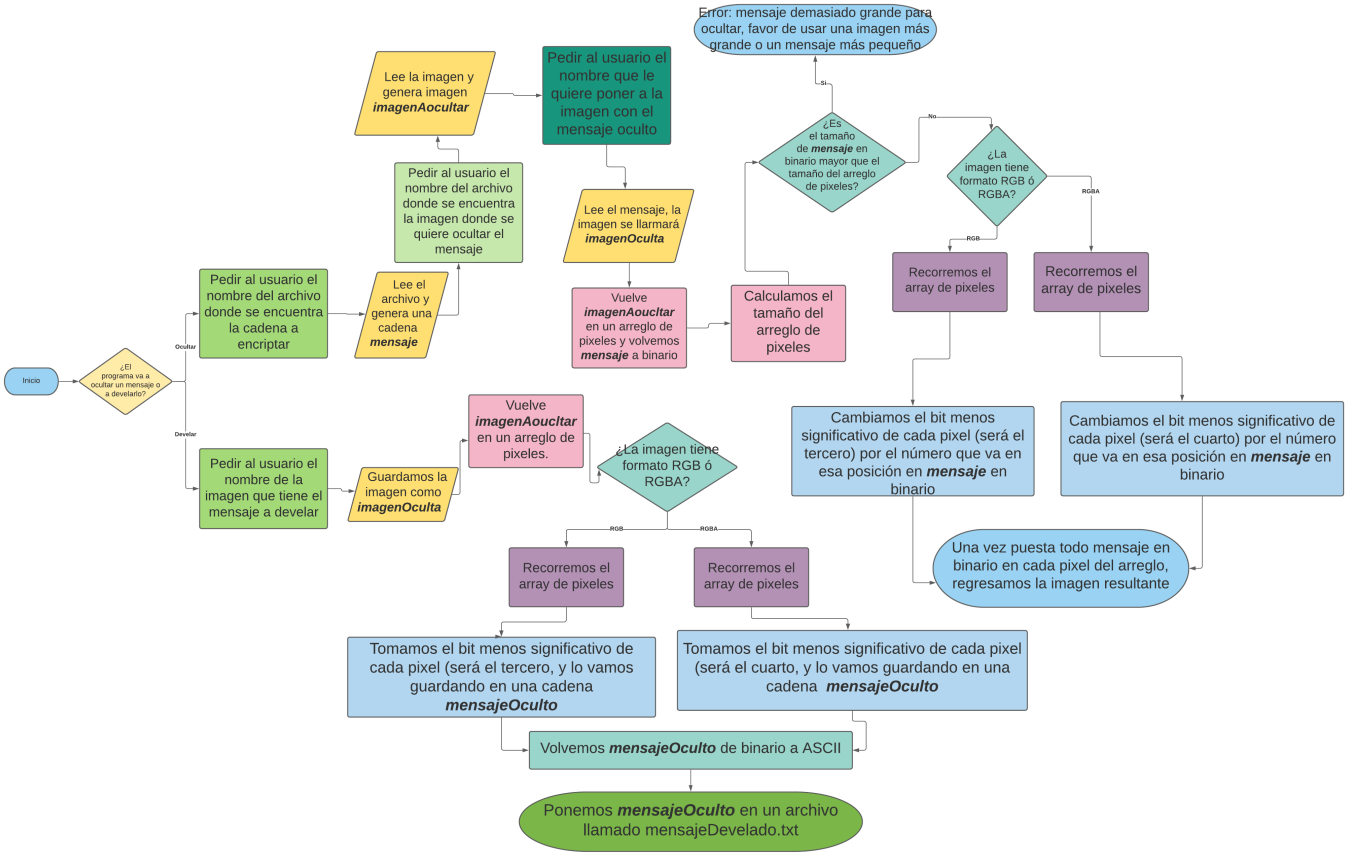
*escribeArchivo(self,nombreArchivo,mensaje)*

Este método funciona de igual forma que *leeArchivo*. Usamos el método *open* pero con la letra "w", para escribir.

Si un archivo con el nombre *nombreArchivo* no existe, entonces lo creamos.

Finalmente usamos el método *write()* para escribir el mensaje en el archivo.

## Diagrama de flujo




*El diagrama en pantalla completa se encuentra en la imagen DIAGRAMADEFLUJO*

*5)Mantenimiento a futuro*

Creemos que el mayor mantenimiento a futuro sería minimizar el tiempo de ejecución del programa, ya que es tardado.