UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Esteganografía

Modelado y programación

Autores:
Valencia Cruz Jonathan Josué
318280211
Aguirre Muñoz Leonardo
318017686

20 de enero de 2022

1. Definición del problema

A partir de estenografía por método LSB, se nos pide ocultar/develar información dentro de una imagen. Este método consiste en utilizar el bit menos significante que se encuentra en cada uno de los píxeles que conforman la imagen para almacenar los diferentes caracteres de un texto en formato binario (Bit menos significante, por sus siglas en ingles Least Significant Bit, normalmente se encuentra en algunos de los extremos de la cadena de bits, esto varía de acuerdo a la arquitectura de la computadora).

El programa funcionará a partir de dos opciones a petición del usuario:

Ocultar. Se debe proporcionar, en la línea de llamada:

1. El nombre del archivo que contiene el texto a ocultar.
2. El nombre del archivo de imagen.
3. El nombre del archivo de imagen resultante con los datos ocultos.

Develar. Se debe proporcionar, en la línea de llamada:

1. El nombre del archivo con la imagen que contiene los datos ocultos.
2. El nombre del archivo en el que se guardará el texto develado.

2. Análisis del problema

Para la resolución del problema, planteamos realizar diferentes clases. a continuación se presenta cada una de las clases adjunto a su descripción:

- Clase imagen: Esta clase se encargará del procesamiento de imágenes, específicamente tendrá métodos especializados en la lectura/escritura de las diferentes propiedades que conforman a la imagen, tales como la altura, anchura y las matrices de píxeles en cada uno de los canales.
- Clase mensaje: Se especializará en el procesamiento de archivos con formato .txt, específicamente se encargará de obtener/almacenar diferentes aspectos tales como la escritura/lectura del contenido, longitud del texto y obtener caracteres individuales en formato ASCII.
- Clase Ocultar: Servirá como subclase de apoyo para la creación de la opción ocultar, está opción primeramente solicitará un archivo .txt y una imagen .png, posteriormente y con la ayuda de las clases imagen y ocultar, obtendrá cada unos de los caracteres que conforman el archivo .txt en formato binario y los almacenará en el bit menos significativo de cada unos de los píxeles, y por último almacenará la imagen resultado con el nombre solicitado al usuario.
- Clase Develar: De igual manera, será una subclase de apoyo para la creación de la opción develar, su funcionamiento comenzará por hacer la petición de la imagen con el mensaje oculto, este asimismo y con las clases mensaje e imagen, obtendrá cada uno de los bits menos significativos en la imagen, agrupará esos bits en cadenas de 8 bits, codificará los caracteres por medio de código ASCII, reconstruirá el texto y por último almacenará el texto obtenido con nombre a petición del usuario.
- Clase Main: Funcionará como clase principal y servirá para la creación de la interfaz de usuario e implementación de las demás clases. Será el archivo a ejecutar para el uso del programa.

3. Selección de la mejor alternativa

En este caso, trabajaremos con los siguientes parámetros para la realización del proyecto:

- Se utilizará el formato .png para la imagen, dada a la característica "lossless compression", es decir, almacena los datos de la imagen sin pérdida de datos por compresión. Este formato nos ayuda a que nuestro mensaje no se pierda en el proceso.
- Se usará archivos .txt para el guardado de información, por indicaciones del proyecto y por su fácil manejo de importación/exportación de datos.
- Para la codificación de caracteres se utilizará código ASCII, esto por su practicidad y uso generalizado en diferentes medios.

- El formato de arquitectura "little-endian.estará presente dado los formato .png y ASCII, es decir, el bit menos significativo se localizará al extremo derecho de la cadena de bits.
- El lenguaje de programación a utilizar será Python, esto debido a las diferentes bibliotecas para el procesamiento de imágenes digitales y de texto, creación de interfaces gráficas de usuario y el fácil entendimiento de su sintaxis, lo que facilita en gran medida la legibilidad del programa.

4. Diagrama de flujo o pseudocódigo

```
#Clase que procesa las imágenes
class Imagen:

    #Función que se encarga de cargar la imagen
    cargarimagen():

        #Le pregunta al usuario la localización de la imagen
        nombrefoto = preguntarNombreArchivo{Seleccionar_archivo .png}

        #Abre la imagen
        imagen = abrirImagen(nombrefoto)

        regresar imagen

    #función que obtiene las dimensiones de una imagen
    obtenerdimensiones(datosImagen):

        tamaño = datosImagen.tamaño

        regresar tamaño

    #Función que obtiene el ancho dada una imagen
    obtenerancho(dimensiones):

        ancho=dimensiones[0]

        regresar ancho

    #Función que obtiene el ancho dada una imagen
    obtenerlargo(dimensiones):

        largo=dimensiones[1]

        regresar largo

    #Función que obtiene los arreglos de píxeles de la imagen
    obtenerpixeles(datosImagen):

        pixeles=datosImagen.cargar()

        regresar pixeles

    #Función que obtiene los píxeles requeridos para almacenar la información
    PixelesNecesarios(cantidad,pixelesdisponibles,largo):

        espacio=0

        mientras cantidad > largo:

            cantidad=cantidad-largo
```

```
        espacio=espacio+1

    PixelesReservados=[]

    veces=0

    mientras veces<espacio:

        desde i hasta largo:

            PixelesReservados.encolar(pixelesdisponibles[veces,i])

        veces=veces+1

    desde i hasta cantidad:

        PixelesReservados.encolar(pixelesdisponibles[espacio,i])

    regresar PixelesReservados

#Función que obtiene la lista de valores de píxeles en formato binario
PixelEnBinario(ListaPixeles):

    tamañoLista=len(ListaPixeles)

    ListaBinaria=[]

    desde i hasta tamañoLista:

        #Ciclo para pasar por cada canal RGB
        desde j hasta 3:

            #Obtiene y transforma los valores a binario
            NumeroBinario=bin(ListaPixeles[i][j])

            #Añade los valores a una lista
            ListaBinaria.agregar(NumeroBinario)

    regresar ListaBinaria

#Obtiene el ancho requerido para almacenar la información
 AnchoRequerido(cantidad,largo):

    ancho=0

    mientras cantidad > largo:

        cantidad=cantidad-largo

        ancho=ancho+1

        regresar ancho

    regresar ancho

#Transforma una lista binaria a valores decimales
PixelDecimal(Lista):

    longitud=len(Lista)

    RGBModificado=[]
```

```
        desde i hasta longitud:

            valorRGB=decimal(Lista[i],2)

            RGBModificado.agregar(valorRGB)

        regresar RGBModificado


#Se realiza la clase Mensaje
clase Mensaje:

    #Obtiene el contenido de un archivo .txt
     abrirarchivo(ruta):

        #Se abre el archivo .txt
        texto = abrir(ruta)

        #Se lee el contenido de texto
        mensaje = texto.leer()

        #Se cierra el archivo
        texto.cerrar()

        #Regresa el contenido del archivo .txt
        regresar mensaje

    #Regresa la lista de las letras obtenidas en el archivo .txt
    en formato binario a partir de su código ASCII
     obtenerletraporletra(mensaje):

        #Se crea una matriz para almacenar la información en binario
        ListaBinaria=[]

        #Se realiza una lista de las letras contenidas en mensaje
        desde char hasta mensaje:
            listaletras= enlistar(mensaje)
            tamaño= len(listaletras)

        #Se obtiene el valor binario de cada letra contenida en
        listaletras a partir de su código ASCII
        desde x hasta tamaño:
            letraASCII=ASCII(listaletras[x])
            LetraBinaria=binario(letraASCII)

        #Registra la longitud de ListaBinaria en tamaño
        tamaño=len(ListaBinaria)

        #Se le agrega '0' al final de cada elemento contenido en ListaBinaria
        mientras len(ListaBinaria) % 3 !=0:
            ListaBinaria.agrega('0')

        regresar ListaBinaria

    #Regresa un arreglo que contiene los bits que conforman a ListaRGB en cadenas de 8 bits
     extraerBit(ListaRGB):

        #Se inicializa una cadena
        byte=""
```

```
    #Se declara un arreglo para almacenar los bits en ListaBinaria
    ListaBitsExtraidos=[]

    #Se obtiene la longitud de ListaRGB
    longitud=len(ListaRGB)

    #Agrupa los bits en cadenas de 8
    desde i hasta longitud:

        bitExtraido=ListaRGB[i][7]
        byte=byte+str(bitExtraido)

        si len(byte)%8==0:

            ListaBitsExtraidos.agregar(byte)
            byte=""

    regresar ListaBitsExtraidos

#Convierte los valores binarios a decimales dada una lista
 BinarioAdecimal(ListaBinaria):

    #Obtiene la longitud de la lista
    longitud=len(ListaBinaria)

    #Traduce cada cadena de números binarios a números decimales
    desde i hasta longitud:

        NumeroASCII=decimal(ListaBinaria[i],2)
        ListaBinaria[i]=NumeroASCII

    #Regresa la lista de números en formato decimal
    regresar ListaBinaria

#Traduce los números decimales a caracteres a partir del código ASCII
 ConvertirEnChar(ListaDecimal):

    #Obtiene la longitud de ListaDecimal
    longitud=len(ListaDecimal)

    #Traduce cada número decimal a caracteres ASCII
    desde i hasta longitud:

        LetraASCII=char(ListaDecimal[i])
        ListaDecimal[i]=LetraASCII

    regresar ListaDecimal

#Imprime en pantalla el texto develado adjunto a un texto,
además de almacenarlo en formato .txt a petición del usuario
 EscribirMensaje(ListaChar):

    #Se almacena la longitud de la lista de caracteres del mensaje develado
    longitud=len(ListaChar)

    #Se declara una cadena vacía
    texto=""

    #Adjunta los caracteres de la lista en una cadena de texto
    desde i hasta longitud:
```

```
            letra=ListaChar[i]
            texto=texto+letra

        #Imprime un mensaje previo al mensaje develado
        imprimir("El texto descifrado es:\n\n\n"+strtexto))

        #Imprime un mensaje para preguntar la forma al que se le
        nombrará al archivo .txt que almacenará el mensaje develado
        nombreArchivo= entrada("\n\n¿Cómo se llamará el archivo en
        el que se guardara el mensaje?\n\n")

        #Se crea el archivo .txt con el nombre dado por el usuario
        y en el se almacena el texto develado
        Archivo= abrir(nombreArchivo+str(".txt"),"a")
        Archivo.escribir(texto)
        Archivo.cerrar()

        regresar texto


#Definimos la clase ocultamiento que nos servirá para obtener la información oculta en una imagen
class Develar:

    #Definimos el método develar
     develar(im):

        #Se almacena la dirección de la imagen en datosImagen
        datosImagen=imagen.cargarimagen(im)

        #Obtenemos el tamaño de la imagen
        dimensiones=imagen.obtenerdimensiones(datosImagen)

        #Guardamos el ancho de la imagen
        #ancho=imagen.obteneranchо(dimensiones)

        #Guardamos el largo de la imagen
        largo=imagen.obtenerlargo(dimensiones)

        #Obtiene los valores de los pixeles encontrados en datosImagen
        Pixeles=imagen.obtenerpixeles(datosImagen)

        #Llamamos a PixelesNecesarios y recoje el resultado en Lista
        Lista=imagen.PixelesNecesarios(960,Pixeles,largo)

        #Transforma la información contenida en Lista a binario
        ListaRGBBinaria=imagen.PixelEnBinario(Lista)

        #Obtenemos el mensaje oculto en la imagen por cada canal
        en formato binario
        BytesExtraidos=mensaje.extraerBit(ListaRGBBinaria)

        #Convertimos la información a decimal
        BytesExtraidos=mensaje.BinarioAdecimal(BytesExtraidos)

        #Traducimos los números decimales a código ASCII
        BytesExtraidos=mensaje.ConvertirEnChar(BytesExtraidos)

        #Se imprime el mensaje en pantalla y se almacenará a
        petición del usuario con formato .txt
        #print(BytesExtraidos)
        textoDescifrado=mensaje.EscribirMensaje(BytesExtraidos)
```

```
        regresar textoDescifrado

#Definimos la clase ocultamiento que nos servirá para almacenar información en una imagen
class Ocultamiento:

    #Definimos la clase ocultar
     ocultar(nombrefoto):

        #Se le pide al usuario abrir el archivo .txt
        ruta = askopenfilename(defaultextension='.txt',filetypes = [
        ("Formato txt", ".txt")],title="Selecciona el archivo .txt")

        #Almacenamos la dirección del archivo en donde se almacena el mensaje
        texto=mensaje.abrirarchivo(ruta)

        #Guardamos la información contenida en texto
        lista=mensaje.obtenerletraporletra(texto)

        #Obtenemos la longitud de lista
        tamaño= decimal(len(lista)/3)

        #Se almacena la dirección de la imagen en datosImagen
        datosImagen=imagen.cargarimagen(nombrefoto)

        #Obtenemos el tamaño de la imagen
        dimensiones=imagen.obtenerdimensiones(datosImagen)

        #Guardamos el ancho de la imagen
        ancho=imagen.obteneranchо(dimensiones)

        #Guardamos el largo de la imagen
        largo=imagen.obtenerlargo(dimensiones)

        #Obtiene los valores de los pixeles encontrados en datosImagen
        Pixeles=imagen.obtenerpixeles(datosImagen)

        #Llamamos a PixelesNecesarios y recoje el resultado en Lista
        Lista= imagen.PixelesNecesarios(tamaño,Pixeles,largo)

        #Transforma la información contenida en Lista a binario
        ListaRGBBinaria=imagen.PixelEnBinario(Lista)

        #Calcula el ancho requerido para almacenar la información
        AnchoNecesitado=imagen.AnchoRequerido(tamaño,largo)

        #Se registra la longitud del mensaje convertido en binario
        longitud=len(ListaRGBBinaria)

        #Se crea una matriz
        Nuevo=[]

        #A partir de un ciclo for se almacena el mensaje en binario para cada canal RGB
        desde i hasta range(longitud):
            bytemodificado=ListaRGBBinaria[i][:-1]+str(lista[i])
            Nuevo.agregar(bytemodificado)

        #Se almacena la matriz de pixeles asociada a la imagen en valores decimales
        ListaModificadaRGB=imagen.PixelDecimal(Nuevo)

        #Se inicializan y declaran las variables contador y limite
```

```
        contador=0
        limite=0

        #Se guarda la longitud de Lista
        LongitudRGB=len(Lista)

        #Se crea una matriz asociada para cada canal RGB
        ListaRoja=[]
        ListaVerde=[]
        ListaAzul=[]

        #Con un ciclo while se guarda el mensaje en formato
        binario en los canales RGB de la imagen para al final
        guardar la imagen modificada en formato png
        mientras contador < AnchoNecesitado+1:

            desde i hasta range(0,longitud,3):
                ListaRoja.agregar(ListaModificadaRGB[i])
            desde i hasta range(1,longitud,3):
                ListaVerde.agregar(ListaModificadaRGB[i])
            desde i hasta range(2,longitud,3):
                ListaAzul.agregar(ListaModificadaRGB[i])

            pixel=Pixeles[contador,tamaño]

            desde i hasta tamaño:
                Pixeles[contador,i]=(ListaRoja[i],ListaVerde[i],ListaAzul[i])

            contador=contador+1

        regresar datosImagen

#Se declara la clase Main, servirá para interactuar con el usuario
main():

    #Se llama a ocultar y develar
    oculta=Ocultamiento()
    devela=Develar()

    #Despliega un "menú" para la selección de opciones
    opcion=entrada("Bienvenido a RevealShowApp\n\n¿Qué te gustaría hacer?\n\n\n1.
    -Ocultar texto en una imagen\n\n\n2.- Develar alguna imagen
    que tenga un mensaje oculto\n\n\n3.-Salir\n\n"))
    si opcion==1:
        oculta.ocultar()
    si opcion==2:
        devela.develar()
    si opcion==3:
        imprimir("Gracias por usar nuestra aplicación, vuelve pronto")
    en otro caso:
        imprimir("Opción invalida, inténtalo de nuevo")

main()
```

5. Mantenimiento y proyecto a largo plazo

A lo largo de la realización de este proyecto se presentaron diversas complicaciones y se idealizaron diferentes cuestiones que podrían hacerse en un futuro como parte de dar mantenimiento al programa, algunos de estos planteamientos son:

a) Una de las cosas a mejorar a corto plazo es la parte de implementación de una interfaz gráfica o de algún medio de visualización más agradable para el usuario, esta interfaz ayudaría para hacer del programa más amigable con el usuario.

b) Otra cuestión a remarcar es la implementación de diferentes formatos de imágenes que cumplan para la correcta aplicación de la estenografía por método LSB, como lo serían JPEG, FILF, BFG y demás formatos imagen "lossless compress". Esto ayudará a que el programa sea más variado y completo en funciones.

c) El tercer posible mantenimiento que podría requerir el programa sería la parte de verificar actualizaciones, ya sean por parte del lenguaje de programación, de los diferentes formatos de imagen/texto o incluso del mismo sistema operativo pues en el caso de los formatos podría suponer un problema mayor para el programa pues supondría un cambio total en el funcionamiento interno del programa, lo que haría de nuestro programa obsoleto y es por eso que sería un importante punto a considerar

d) El cuarto es la actualización del código para atender complicaciones que puedan surgir por parte de los usuarios pues suele suceder que cuando se crea un programa, no se tiene en consideración algunos aspectos que pueden hacer de errores para algunos usuarios, algunos de estos errores pueden ser bugs, situaciones no planeadas por parte del programa, errores de compatibilidad con algún equipo por parte de algún usuario, entre otros.

e) El quinto y último punto a mencionar pero no menos importante es el mejoramiento y optimización de código de manera general pues al ser un proyecto creado a partir de programadores con poca experiencia y en formación da lugar a que el programa sea poco eficiente en algunos aspectos y que se podrían mejorar con la implementación de algunas metodologías.

   Una vez dicho lo anterior, sobre la cuestión de cuánto se esperaría cobrar por la realización de este programa y por futuros mantenimientos, una situación a plantearse consiste en la labor de un programador, que presentan varios retos, siendo principalmente el planteamiento y desarrollo en la solución de un problema un aspecto clave a considerar al momento de estimar algún honorario además de quien solicita el servicio del programador, horario de trabajo, fecha de entrega, factores externos como lo es consultar el servicio de demás lugares, el tipo de problema a resolver; esto formula la "pregunta definitiva"de responder pero diremos que, en este caso, se cobraría por el programa un monto de alrededor de 1000 a 4000 pesos aproximadamente pues se trata de un proyecto menor y que no requiere de una gran planificación de por medio.

6. Cambios realizados recientemente

Se realizaron los siguientes cambios:

a) Se hicieron cambios en la organización de las clases y subclases que componen el programa, esto con el fin de contribuir al mantenimiento e implementación de código.

b) Se optó por agregar una interfaz gráfica de usuario (GUI) al programa con el fin de facilitar la visualización de los diferentes resultados.

c) Se reestructuró la organización de los archivos contenidos en Estenografía/, se crearon las carpeta de nombre .Estenografía", RevealShowApp", Ïmágenesz "Texto", además se movieron los diferentes archivos a estas carpeta.

d) Se hizo un pequeño cambio de nombre al archivo "Main.py", se renombró a "Main.pyw", esto con el fin de hacer del programa un ejecutable

e) Se agregó la documentación al código del programa.

f) Se agregó el archivo "Proyecto_Aguirre_Leonardo_Valencia_Jonathan.ª .Estenografía/", además de agregar contenido a README.md", en él se muestra cómo ejecutar el programa además de información adicional de cada archivo ubicado en el proyecto.

Puedes ver los cambios más a detalle en el apartado Commits en el GIT del proyecto:

https://github.com/WraithLion/Esteganografia/commits/main