



BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

Licenciatura en Ciencias de la Computación

Tópicos de Ingeniería en Computación



BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

CRİPTOGRAFIA EN MATLAB |

DR. ARNULFO LARA ELIOSA

EDGAR ESPINOSA ORDOÑES 201119702

PROYECTO: ESTEGANOGRAFIA CON IMÁGENES EN MATLAB

## Definición:

La **esteganografía** (del griego στεγανος *steganos*, "cubierto" u "oculto", y γραφος *graphos*, "escritura") trata el estudio y aplicación de técnicas que permiten ocultar mensajes u objetos, dentro de otros, llamados *portadores*, de modo que no se perciba su existencia. Es decir, procura ocultar mensajes dentro de otros objetos y de esta forma establecer un canal encubierto de comunicación, de modo que el propio acto de la comunicación pase *inadvertido* para observadores que tienen acceso a ese canal.

Una forma de diferenciar la esteganografía con la criptografía común es que la criptografía solo cifra los archivos manteniendo el archivo original visible, pero al abrirlo mostrará una secuencia de caracteres que no permitirá su lectura y para ver su contenido original es necesario conocer la clave. En la esteganografía, puede verse un archivo con un formato diferente, y para conocer su contenido original será necesario conocer la clave y el software con el que se ocultó.

## Funcionamiento y terminología:

- ? Se define como *esquema esteganográfico* al conjunto de componentes que permite llevar a cabo la comunicación esteganográfica.
- ? El *portador* es todo aquel conjunto de datos que es susceptible de ser alterado para incorporarle el mensaje que queremos mantener en secreto. Ejemplos: imagen (en sus distintos formatos), audio (en sus distintos formatos), texto plano, archivos binarios, un mensaje de protocolo de comunicación.
- ? *mensaje-legítimo* para referirse al mensaje transportado por el portador.
- ? Se llama *mensaje esteganográfico* al mensaje que queremos mantener en secreto y queremos esconder dentro del portador.
- ? *Estego-algoritmo* es el algoritmo esteganográfico que indica cómo realizar el procedimiento de incorporación del mensaje que queremos mantener en secreto en el portador.
- ? La acción de ocultar el mensaje dentro del portador se denomina *empotrar*.
- ? Se llama *estego-mensaje* al resultado de empotrar el mensaje esteganográfico dentro del portador.
- ? La acción de la recuperación, a partir del estego-mensaje, del mensaje oculto esteganográfico se denomina *extraer*.

## IMPLEMENTACION DEL PROYECTO:

Haciendo uso de las múltiples funciones que ofrece Matlab para el procesamiento de imágenes se implementara un procedimiento de esteganografía para realizar cifrado de texto llano en formato .txt hacia una imagen elegida.

- 1) Se define una serie de formatos de imagen a utilizar como medio para realizar el procedimiento, definidas por:

.png

.jpg

.bmp

- 2) Se obtendrá el texto de un archivo cualquiera con extensión .txt ya que es texto plano sin formato simplemente caracteres que componen el conjunto ASCII

### LSB (Low Significant Bit)

Para nuestro caso puntual de la técnica LSB es necesario tener presente que todo en nuestro computador funciona muy internamente como una combinación de unos (1) y ceros (0), lo cual es definido como sistema binario y hago especial énfasis a esto, dado que trabajaremos sobre ellos.




- Bit = Es la mínima expresión que representa información, cero (0) o uno (1).
- Byte = Es la unidad estándar de información, consta de 8 bits (00000001).

Una imagen esta conformada por píxeles (pixel: picture element), la combinación de dichos píxeles es lo que le da color y forma a la imagen terminada, como lo podemos observar, cuando le hacemos mucho zoom a una imagen, generalmente vamos a encontrarnos con que todo esta conformado por pequeños cuadritos y son estos a los que nosotros llamamos píxeles, pues son la unidad mínima de representación para una imagen.



Un píxel está conformado por una tripleta de datos que seguramente muchos han escuchado, algo muy conocido como RGB (Red, Green, Blue), esta tripleta de datos es la que conforman el color final del píxel, por ejemplo, los tres colores se mezclan en diferentes proporciones generando distintos colores:

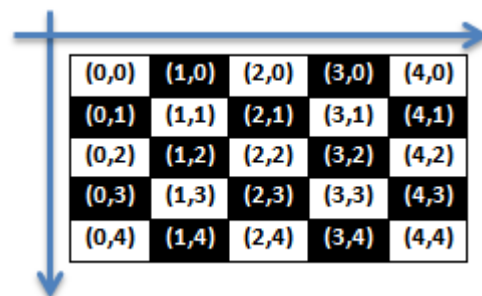
Como podemos observar, distintas combinaciones en RGB generan distintos tipos de colores, unos muy diferentes y otros muy parecidos. El rango de cambio en los valores RGB es de 0 a 255 para cada valor, entonces podemos ver a un píxel como un vaso donde

		
<b>R</b> → 255	<b>R</b> → 255	<b>R</b> → 16
<b>G</b> → 0	<b>G</b> → 100	<b>G</b> → 155
<b>B</b> → 0	<b>B</b> → 0	<b>B</b> → 51

mezclamos pintura, pero donde solo tenemos 3 tipos de pinturas primarias, la roja, la verde y la azul, al mezclar en el vaso distintas proporciones de esos colores vamos a obtener toda una paleta de colores, por ejemplo, el color blanco como todos sabemos (efecto del prisma) está compuesto de todos los colores, por ende en RGB se consigue con la tripleta (255, 255, 255)

y en contraposición, el color negro es la ausencia de color o de luz, en este caso se forma con la mezcla de (0,0,0).

Como podemos observar, cada cuadro blanco y negro representa un píxel y con ello está sujeto a una coordenada con la cual podemos acceder a los valores RGB de dicho píxel. Es importante notar que la coordenada (0,0) es decir, la coordenada inicial, está en la esquina superior izquierda, pues allí es donde comienza toda la imagen.



No solo podemos ver la imagen como píxeles formados por RGB, sino que la podemos empezar a ver como una matriz, en la cual podemos decir, por ejemplo, de la imagen quiero ver el pixel de la posición (2,2), entonces como nuestra imagen es una matriz, vamos a buscar la posición (2,2), lo cual nos daría el centro de la imagen (en el ejemplo), un píxel blanco formado por (255, 255, 255), solo es dar una coordenada y recibimos la tripleta RGB asociada a dicho píxel.

Al pasar por cada pixel obtenemos una tripleta RGB compuesta por números enteros desde el cero (0) hasta el (255), y como cada número tiene su propia representación en binario, lo que haremos será convertir dicha tripleta en su equivalente en binario, por ejemplo, el pixel conformado por:

(148, 28, 202) equivale en binario a (10010100, 00011100, 11001010).

	Binario	LSB	
R -> 148	10010100	10010101	R -> 149
G -> 28	00011100	00011101	G -> 29
B -> 202	11001010	11001011	B -> 203

Ahora lo que hacemos es editar el bit menos significativo, aquel que se encuentre de último a la derecha, como pueden observar en la imagen los hemos resaltado en color rojo, también verán la explicación de por qué es el bit menos significativo,

debajo en la columna LSB hemos alterado los bits (en rojo) pero los demás siguen intactos y el resultado de la tripleta RGB sufre algunos cambios, pero son mínimos. Es muy improbable que hallen algún tipo de diferencia visual pero en realidad hubo un cambio, después de alterar el bit menos significativo, la tripleta RGB es distinta de la que teníamos al inicio, pero el color aparentemente es el mismo.

si deseamos esconder la palabra "Hacking" tenemos que recordar que cada letra (carácter) se puede representar por un Byte siendo así la "H"= 01001000 entonces si tenemos 3 pixeles podemos esconder esa secuencia mediante LSB.

	Binario	"H"	LSB	
R -> 148	10010100	0	10010100	R -> 148
G -> 28	00011100	1	00011101	G -> 29
B -> 202	11001010	0	11001010	B -> 202
R -> 178	10110010	0	10110010	R -> 178
G -> 71	01000111	1	01000111	G -> 71
B -> 22	00010110	0	00010110	B -> 22
R -> 87	01010111	0	01010110	R -> 86
G -> 219	11011011	0	11011010	G -> 218
B -> 100	01100100	-	01100100	B -> 100

Como cada pixel tiene tres valores que lo componen y en cada uno solo podemos alterar un bit, entonces son necesarios 3 pixeles para poder esconder la letra "H", pues su representación en binario corresponde a 8 bits. La

tabla antes presentada es muy intuitiva, pues lo que hacemos es obtener 3 pixeles de la imagen original sacamos sus respectivos RGB y como deseamos esconder la letra "H" en binario, simplemente reemplazamos los bits menos significativos en el orden de la "H" y volvemos a reconstruir nuevamente los 3 pixeles, solo que ahora como escondimos una letra en ellos, sus valores han cambiado, pero como lo habíamos dicho antes, ningún cambio perceptible al ojo humano.

## Ejecución del programa

Debemos listar en una carpeta nuestros archivos a utilizar como son:

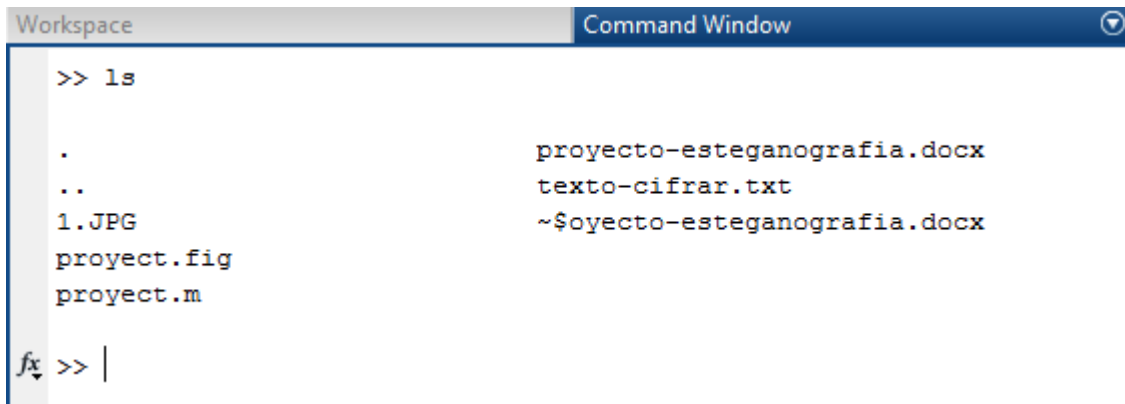
- Código fuente: "proyect.m"
- Interfaz gráfica: "proyect.fig"
- Imagen contenedora: "1.JPEG"
- Texto a cifrar: "texto-cifrar.txt"

Nota: Este programa corre sobre versiones 2017<sup>a</sup> o posteriores de Matlab debido a las librerías utilizadas para la implementación del algoritmo LSB.

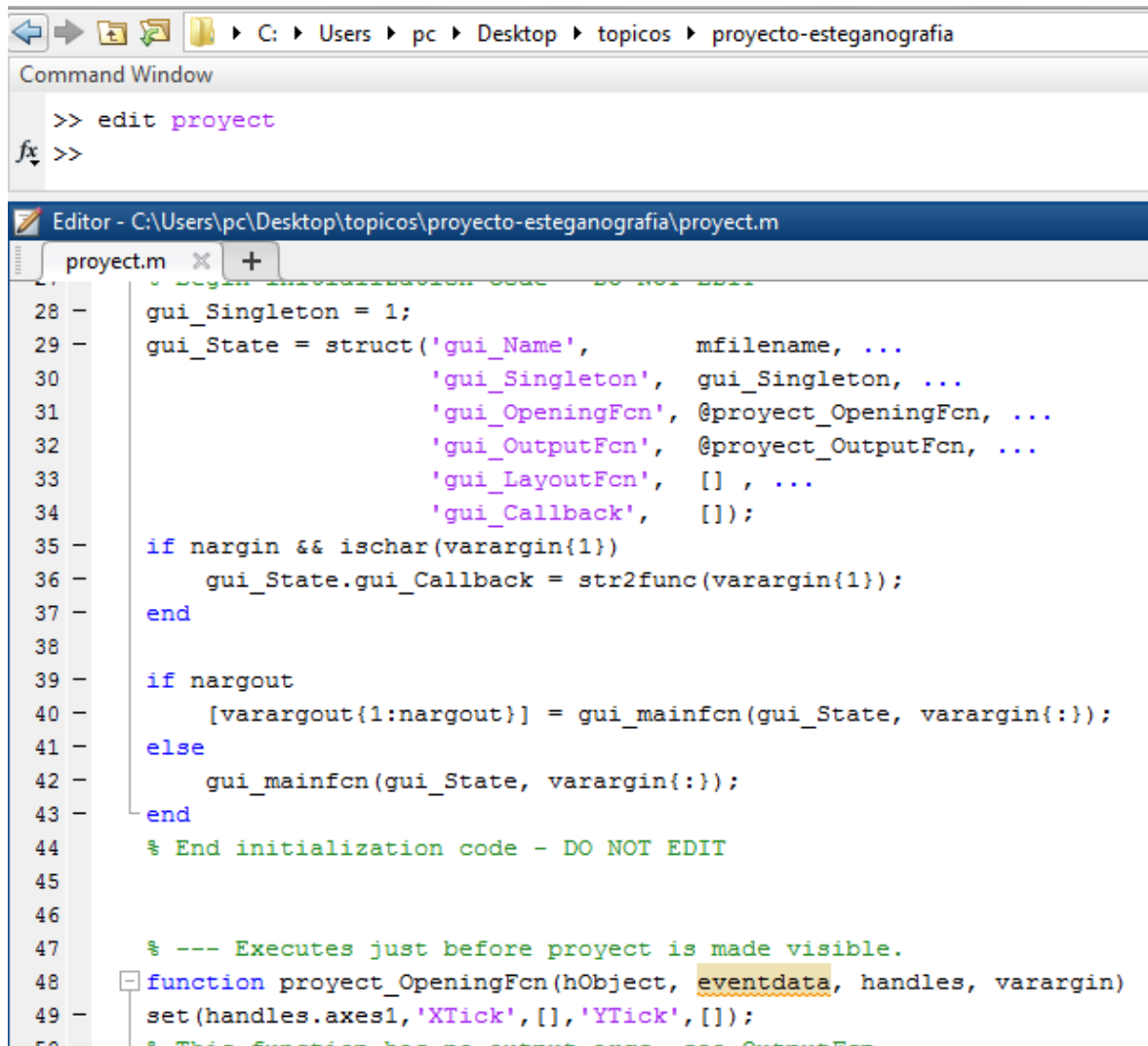
Una vez abierto Matlab procedemos a ubicarnos en el directorio donde se encuentra la carpeta con todos los elementos:



Una vez ubicados en el directorio listaremos el contenido en la ventana de comandos para verificar que se encuentran todos los archivos:



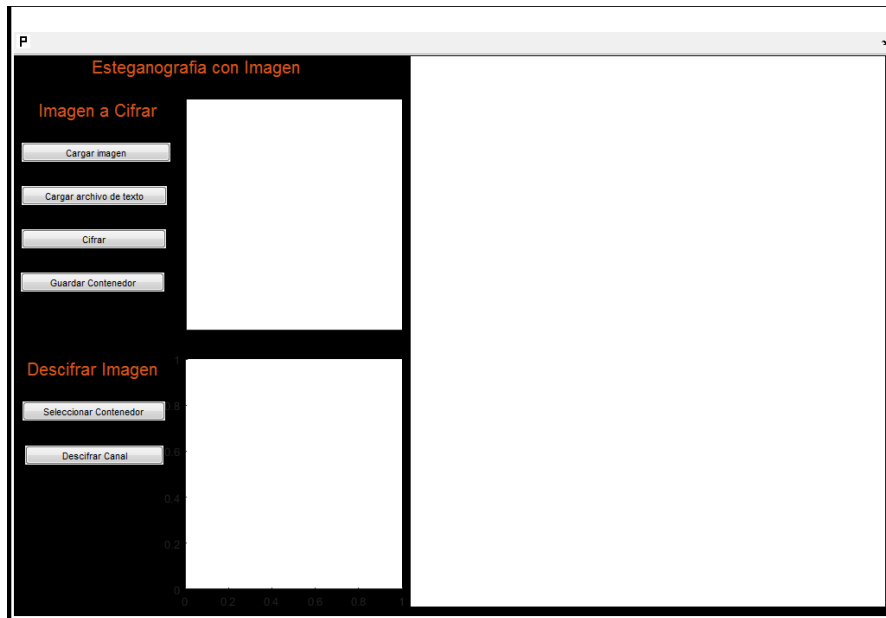
Realizado esto procedemos a abrir nuestro programa tecleando edit project



The screenshot displays the MATLAB environment. At the top, the Command Window shows the command `>> edit project` and a cursor at the next prompt. Below it, the Editor window is open, showing the file `project.m` located at `C:\Users\pc\Desktop\temicos\proyecto-esteganografia\project.m`. The code in the editor includes initialization for a GUI singleton, setting up callback functions for opening, output, layout, and callbacks, and defining the `project_OpeningFcn` function.

```
27 - % Begin initialization code - DO NOT EDIT
28 - gui_Singleton = 1;
29 - gui_State = struct('gui_Name',       mfilename, ...
30 -                   'gui_Singleton',   gui_Singleton, ...
31 -                   'gui_OpeningFcn',   @project_OpeningFcn, ...
32 -                   'gui_OutputFcn',    @project_OutputFcn, ...
33 -                   'gui_LayoutFcn',    [], ...
34 -                   'gui_Callback',     []);
35 - if nargin && ischar(varargin{1})
36 -     gui_State.gui_Callback = str2func(varargin{1});
37 - end
38 -
39 - if nargin
40 -     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41 - else
42 -     gui_mainfcn(gui_State, varargin{:});
43 - end
44 - % End initialization code - DO NOT EDIT
45 -
46 -
47 - % --- Executes just before project is made visible.
48 - function project_OpeningFcn(hObject, eventdata, handles, varargin)
49 -     set(handles.axes1, 'XTick', [], 'YTick', []);
50 - % This function has no output args - see OutputFcn
```

El comando nos mostrara nuestro código correspondiente al archivo proyect.m, ya realizado esto ejecutaremos la interfaz gráfica tecleando proyect que a la vez hace conexión con el código y permite ejecutar el cifrado LSB con imágenes.



#### **Imagen a cifrar realizaremos la carga de los parámetros correspondientes**

1. "Cargar Imagen" nos abrirá un panel para elegir una imagen en formato JPEG, PGN, JPG la cual servirá como nuestro Estego para el cifrado
2. "Cargar Archivo" Con este botón podremos elegir un archivo de texto a ocultar dentro de la imagen, el formato del archivo de texto debe ser ".txt"
3. "Cifrar" Con este botón ejecutamos el código contenido en el archivo proyect.m el cual se encargará de ejecutar los pasos de cifrado con el algoritmo LSB
4. "Guardar Contenedor" Con este botón el programa realizara la escritura de una nueva imagen con el texto ya cifrado, dando como salida una imagen llamada "Cifrada.PNG"

#### **Descifrar Imagen realizaremos la carga de los parámetros correspondientes**






1. "Seleccionar Contenedor" Nos permitirá elegir una imagen que previamente ha sido cifrado con este programa utilizando el método de cifrando con LSB
2. "Descifrar Canal" Al hacer clic en este botón y previamente cargada la imagen nos mostrara en el recuadro derecho el texto que ha sido extraído resultado de realizar el proceso inverso del algoritmo LSB.

**NOTA: Es importante Seguir el orden de los pasos listados anteriormente para que la ejecución del programa sea exitosa.**

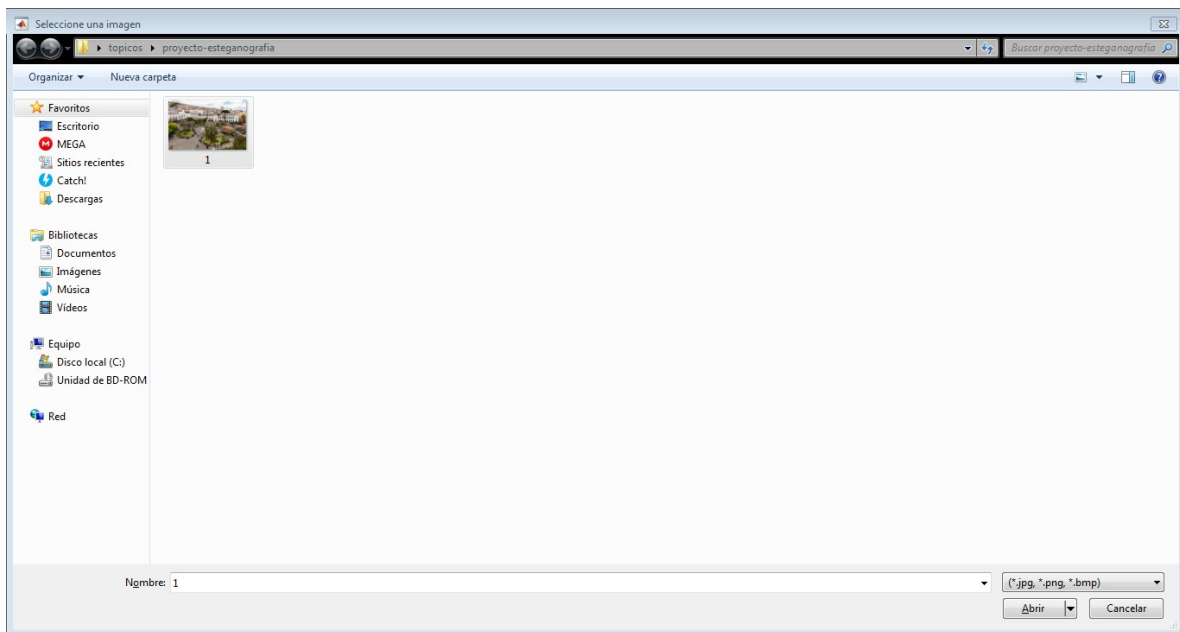


## Capturas de Ejecución del programa CIFRADO

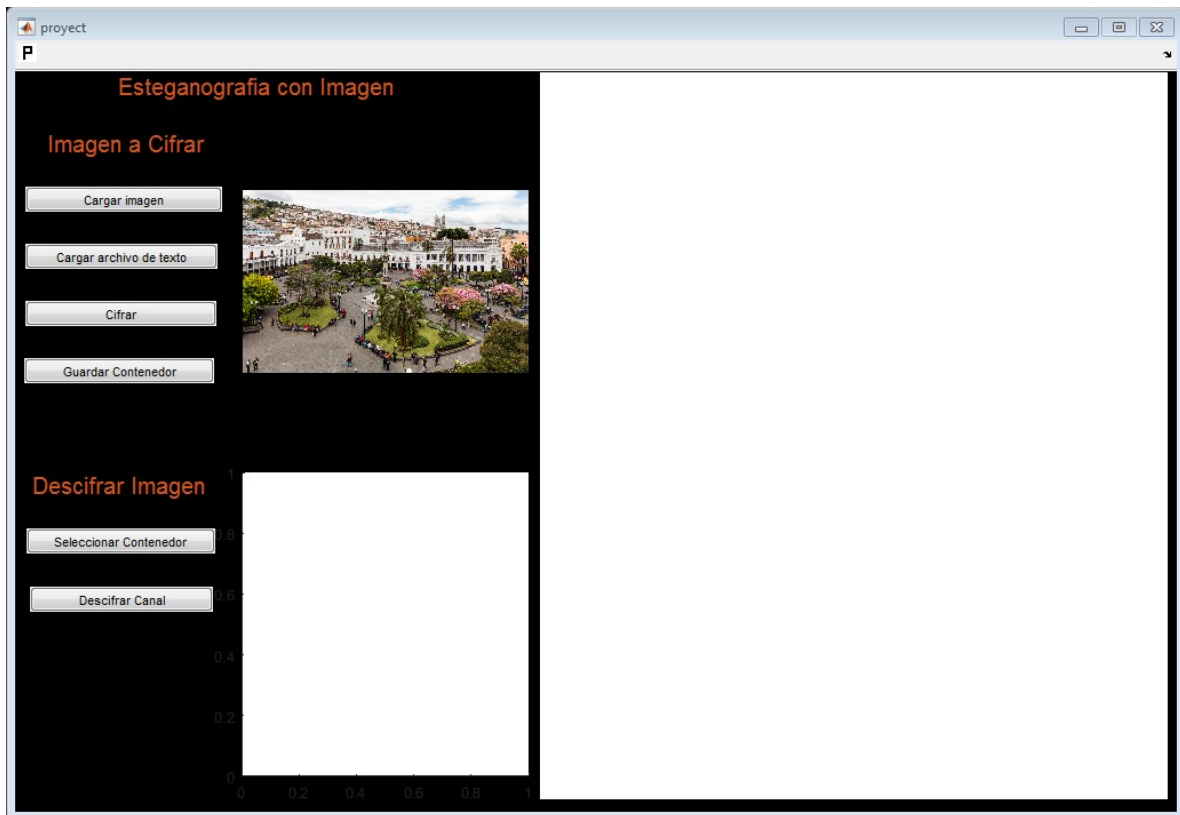
Archivos necesarios para su ejecución en una misma carpeta:

Nombre	Fecha de modifica...	Tipo	Tamaño
 1	16/11/2018 08:35 a...	Imagen JPEG	7,751 KB
 texto-cifrar	16/11/2018 09:00 a...	Documento de tex...	2 KB
 proyect.fig	16/11/2018 09:15 a...	Archivo FIG	38 KB
 proyect.m	17/11/2018 04:38 ...	Archivo M	17 KB
 proyecto-estegan...	17/11/2018 04:54 ...	Documento de Mi...	308 KB

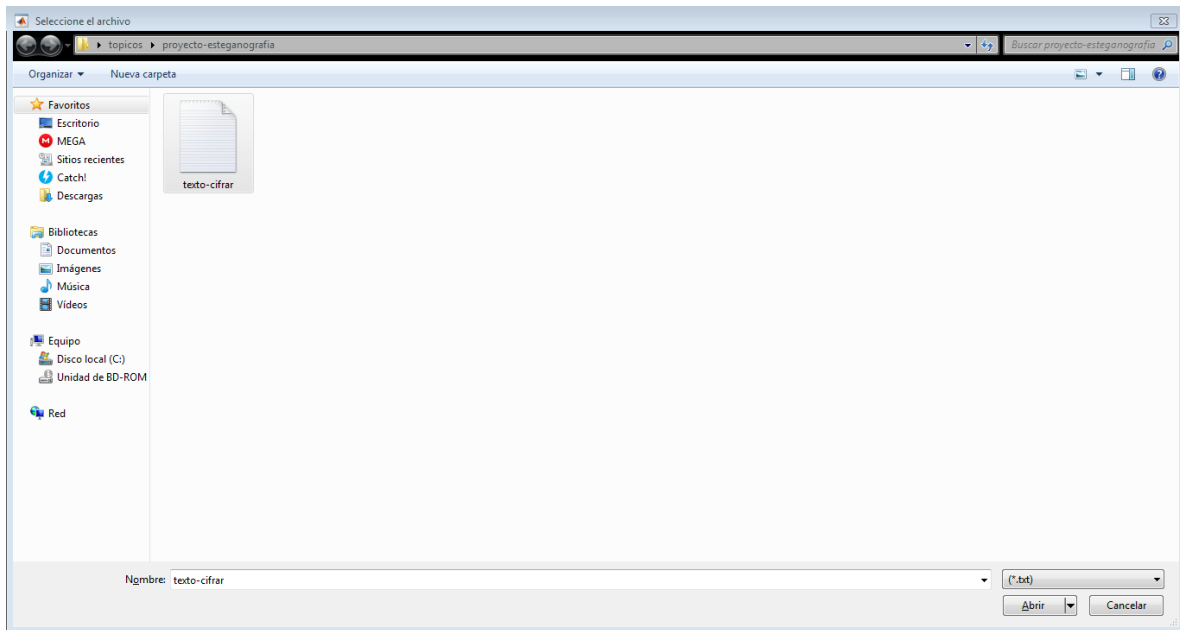
1) Seleccionando imagen a cifrar:



1.1) Imagen cargada en interfaz del programa:



## 2) Carga del texto a cifrar:









## 3) Cifrar

Al hacer clic en este botón tendremos como salida en nuestro command window los parámetros recibidos de la imagen correspondientes a ancho y alto en pixeles.

```
>> project
Warning: MATLAB has disabled some adv...
5162
8042
3
Cifrado terminado .....
```

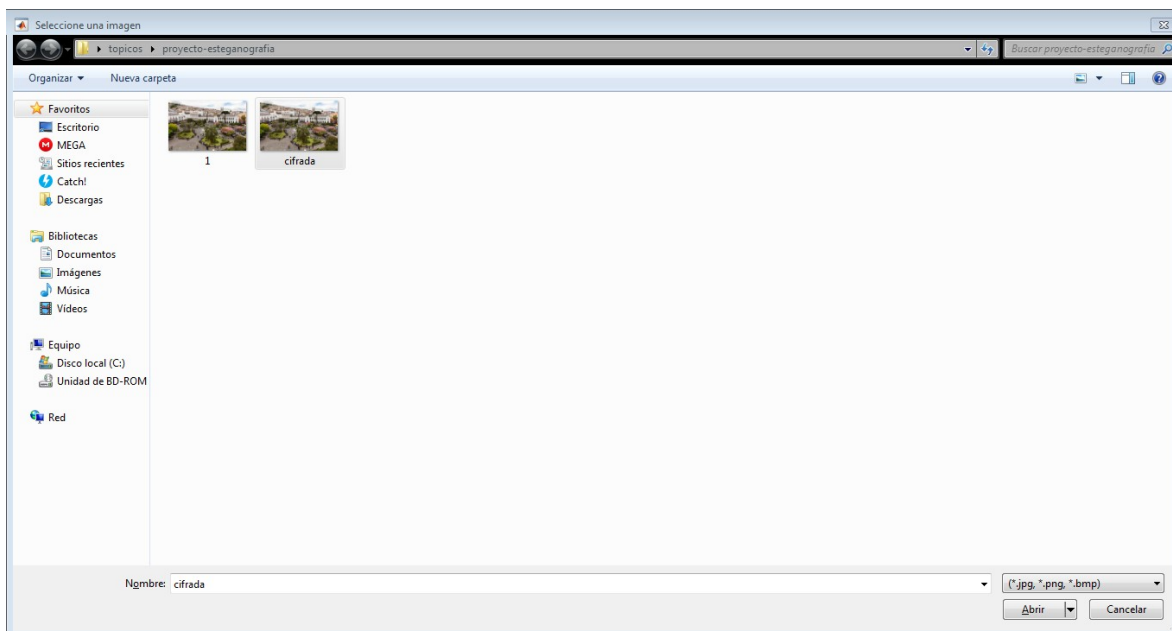
## 4) Guardar imagen

Podemos observar que se ha escrito un nuevo archivo correspondiente a la imagen cifrada.

	texto-cifrar	16/11/2018 09:00 a...	Documento de tex...	2 KB
	project.m	17/11/2018 04:38 ...	Archivo M	17 KB
	project.fig	16/11/2018 09:15 a...	Archivo FIG	38 KB
	proyecto-estegan...	17/11/2018 05:00 ...	Documento de Mi...	561 KB
	1	16/11/2018 08:35 a...	Imagen JPEG	7,751 KB
	cifrada	17/11/2018 05:03 ...	Imagen PNG	53,565 KB

## Capturas de Ejecución del programa DESCIFRADO

1) Seleccionar contenedor para elegir la imagen, tendrá el nombre de “cifrada.PNG”



2) Descifrar Canal:

