UNIVERSIDAD AERONÁUTICA EN QUERÉTARO

INGENIERÍA EN ELECTRÓNICA Y CONTROL DE SISTEMAS DE AERONAVES

FINAL PROJECT

Encryption and decryption algorithm using LSB

SECURE SOFTWARE DESIGN

Teacher: Cházaro Zaharias Adriana Concepción
Oros Díaz Yair Elihú
Developers: Contreras Gutiérrez Cynthia 8387
Garnica Cortes Román 8170

June 19, 2024

Índice general

Índice de figuras

Capítulo 1

Introducción

El mundo digital actual nos pide que protejamos la información que se transmite, esto debido a la sensibilidad de la información, así como para la tranquilidad del usuario, es por ello que se han creado múltiples algoritmos que nos permiten proteger nuestra información.

La técnica del bit menos significativo Least Significant Bit (LBS) se usa para el cifrado en imágenes digitales, el cual permite ocultar la información en el bit de un pixel. La técnica a resultado útil pues al ojo humano es imperceptible el cambio que se realiza, por lo que cada vez más organizaciones lo usan.

Como proyecto final de la materia de Diseño de software Seguro, se implementará el cifrado y descifrado del LSB en ADA. ADA es un lenguaje de programación diseñado específicamente para aplicaciones de sistemas embebidos y críticos, lo cual lo hace la herramienta ideal para el desarrollo de este código. Además, se hará uso de otras herramientas externas que permitan la transformación de una imagen a matriz, lo cual es poco eficiente en ADA por la inexistencia de librerías que permitan el poder transformar las imágenes.

Este proyecto no solo proporcionará una aplicación práctica de los conceptos de seguridad informática y programación en ADA, sino que también contribuirá al desarrollo de técnicas de encriptación de imágenes más avanzadas y robustas.

Capítulo 2

Objetivos

- Implementar un algoritmo de encriptación LSB en ADA que permita ocultar datos confidenciales dentro de los bits menos significativos de los píxeles de una imagen de manera eficiente y segura.
- Implementar un algoritmo de encriptación LSB en ADA que permita desencriptar un mensaje oculto en la matriz de la imagen.
- Desarrollar un sistema de manejo de matrices en Matlab
- Integrar medidas de seguridad adicionales, como la generación de claves de encriptación.

Capítulo 3

Definición del problema

El programa tiene como objetivo principal encriptar y desencriptar un mensaje dentro de una imagen representada por una matriz numérica. Para lograr esto, se requiere un conjunto de funciones que conviertan caracteres a su representación binaria, manejen la entrada de una llave de encriptación y apliquen la encriptación sobre la imagen según los parámetros definidos por la llave. Además, se busca guardar el mensaje encriptado en un formato adecuado para su posterior uso en el algoritmo de descifrado.

Requerimientos para el código de Cifrado:

1.1 **Lectura de Datos:**

1.1.1 Leer una matriz de tamaño 128x128 desde un archivo de texto que contiene los valores de píxeles de una imagen.

1.1.2 Requerimientos funcionales del texto a cifrar

1.1.2.1 El mensaje debe tener una longitud máxima de 20 caracteres.

1.1.2.2 Todos los caracteres del mensaje deben ser letras mayúsculas del alfabeto inglés (A-Z). Se permiten espacios en el mensaje.

1.1.3 Requerimientos No Funcionales del texto a cifrar

1.1.3.1 El sistema debe validar que el mensaje ingresado cumpla con las restricciones especificadas antes de ser aceptado.

1.1.3.2 En caso de que el mensaje ingresado no cumpla con las restricciones, se debe proporcionar un mensaje de error adecuado indicando el motivo del rechazo.

1.1.3.3 El código debe permitir al usuario ingresar el mensaje y recibir retroalimentación inmediata sobre su validez.

1.1.3.4 El sistema debe garantizar que no se permitan caracteres especiales en el mensaje para evitar posibles problemas de seguridad o errores en el procesamiento posterior.

1.1.4 Convertir cada caracter del mensaje a una representación binaria de 8 bits y almacenarlos en una matriz de tamaño 20x8.

1.2 **Procesamiento de la Llave de Encriptación:**

1.1 Verificar que la llave proporcionada no tenga un valor sin sentido y solicitar una nueva llave si es inválida.

1.2 Los bits de la clave se traducen en valores específicos según su posición:

1.2.1 El primer bit indica si la acción es hacia arriba o hacia abajo.

1.2.2 El segundo bit indica si la acción es hacia la izquierda o hacia la derecha.

1.2.3 El tercer bit indica la dirección (positiva o negativa) de la acción arriba/abajo.

1.2.4 El cuarto bit indica la dirección (positiva o negativa) de la acción izquierda/derecha.

1.2.5 El quinto bit indica si se está moviendo en filas o columnas.

1.2.6 El sexto bit indica si se está moviendo 1 bit o 2 bits.

1.2.7 Los bits siete y ocho no se utilizan en la traducción.

1.3 Traducir los 8 dígitos de la llave en valores numéricos que representen la dirección de desplazamiento y el tipo de encriptación.

1.3 **Encriptación de la Imagen:**

1.3.1 Iterar sobre la matriz de la imagen y aplicar la encriptación según los parámetros definidos por la llave.

1.3.2 En el caso de encriptación de 1 bit, modificar el bit menos significativo del píxel según el mensaje binario.

1.3.3 En el caso de encriptación de 2 bits, modificar dos bits consecutivos del píxel según el mensaje binario.

1.3.4 Guardar la imagen encriptada en un archivo CSV para su posterior visualización o procesamiento.

Requerimientos para el código de Descifrado:

1.1 **Lectura de Datos:**

1.1.1 Leer una matriz de tamaño 128x128 desde un archivo de texto que contiene los valores de píxeles de una imagen.

1.2 **Procesamiento de la Llave de Encriptación:**

1.2.1 El primer bit indica si la acción es hacia arriba o hacia abajo.

1.2.2 El segundo bit indica si la acción es hacia la izquierda o hacia la derecha.

1.2.3 El tercer bit indica la dirección (positiva o negativa) de la acción arriba/abajo.

1.2.4 El cuarto bit indica la dirección (positiva o negativa) de la acción izquierda/derecha.

1.2.5 El quinto bit indica si se está moviendo en filas o columnas.

1.2.6 El sexto bit indica si se está moviendo 1 bit o 2 bits.

1.2.7 Los bits siete y ocho no se utilizan en la traducción.

1.3 **Procesamiento del Mensaje:**

1.3.1 Decodificar un mensaje codificado previamente dentro de la matriz de la imagen.

1.3.2 Requerimientos Funcionales del Mensaje a Decodificar:

1.3.2.1 El mensaje debe tener una longitud máxima de 20 caracteres.

1.3.2.2 Todos los caracteres del mensaje deben ser letras mayúsculas del alfabeto inglés (A-Z). Se permiten espacios en el mensaje.

Capítulo 4

Definición de la solución

Algoritmo de Solución para el cifrado:

4.0.1. Datos de Entrada:

- Matriz de la imagen de tamaño 128x128.
- Mensaje de texto a encriptar (`chars`).
- Llave de encriptación.

4.0.2. Datos de Salida:

- Matriz de la Imagen encriptada

4.0.3. Algoritmo

1. Leer la matriz de la imagen desde un archivo de texto.
   - 1.1 Abrir el archivo de texto que contiene la matriz de la imagen.
   - 1.2 Leer cada valor de píxel del archivo y almacenarlo en la matriz de la imagen.
   - 1.3 Cerrar el archivo.
2. Convertir cada carácter del mensaje a una representación binaria de 8 bits y almacenarlo en una matriz de tamaño 20x8.
   - 2.1 Para cada carácter en el mensaje:
     - 2.1.1 Obtener el valor numérico del carácter.
     - 2.1.2 Convertir el valor numérico a binario y almacenar cada bit en la matriz binaria.
3. Verificar la validez de la llave de encriptación y traducirla en parámetros de encriptación.
   - 3.1 Comprobar si la llave es válida y no tiene un patrón incorrecto.
   - 3.2 Traducir los 8 dígitos de la llave en valores numéricos que representen la dirección de desplazamiento y el tipo de encriptación.

4. Encriptar la imagen utilizando la llave de encriptación y el mensaje binario.

    4.1 Para cada posición en la matriz binaria:

        4.1.1 Obtener los parámetros de encriptación de la llave.

        4.1.2 Según el tipo de encriptación (1-bit o 2-bits):

            4.1.1.1 Para cada píxel en la matriz de la imagen:

            4.1.1.2 Convertir el píxel a binario.

            4.1.1.3 Modificar el bit menos significativo del píxel con el bit encriptado actual.

            4.1.1.4 Convertir el píxel de nuevo a decimal y actualizar la matriz de la imagen.

            4.1.1.5 Desplazar la posición según los parámetros de la llave.

5. Guardar la imagen encriptada en un archivo CSV para su posterior visualización o procesamiento.

    5.1.1 Crear un archivo CSV para la imagen encriptada.

    5.1.2 Escribir cada valor de píxel de la imagen en el archivo CSV.

    5.1.3 Cerrar el archivo.

Algoritmo de Solución para el descifrado:

4.0.4. Datos de Entrada:

- Matriz de la imagen (`img`) con el mensaje cifrado de tamaño 128×128.
- Llave de desencriptación (`key`).

4.0.5. Datos de Salida:

- Mensaje desencriptado

4.0.6. Algoritmo

1. Leer la matriz de la imagen desde un archivo de texto.

    1.1 Abrir el archivo de texto que contiene la matriz de la imagen.

    1.2 Leer cada valor de píxel del archivo y almacenarlo en la matriz de la imagen.

    1.3 Cerrar el archivo.

2. Leer y validar la llave de desencriptación.

    2.1 Verificar la validez de la llave.

    2.2 Traducir la llave a parámetros de desencriptación.

3. Desencriptar la imagen utilizando la llave y obtener el mensaje decodificado.

    3.1 Para cada posición en la matriz binaria del mensaje:

        3.1.1 Obtener los parámetros de desencriptación de la llave.

3.1.2 Según el tipo de desencriptación (1-bit o 2-bits):

3.1.2.1 Para cada píxel en la matriz de la imagen:

3.1.2.2 Obtener el bit menos significativo del píxel.

3.1.2.3 Almacenar el bit en la matriz binaria del mensaje.

3.1.2.4 Desplazar la posición según los parámetros de la llave.

4. Guardar el mensaje decodificado en un archivo de texto.

4.1 Crear un archivo de texto para el mensaje decodificado.

4.2 Escribir el mensaje decodificado en el archivo.

4.3 Cerrar el archivo.

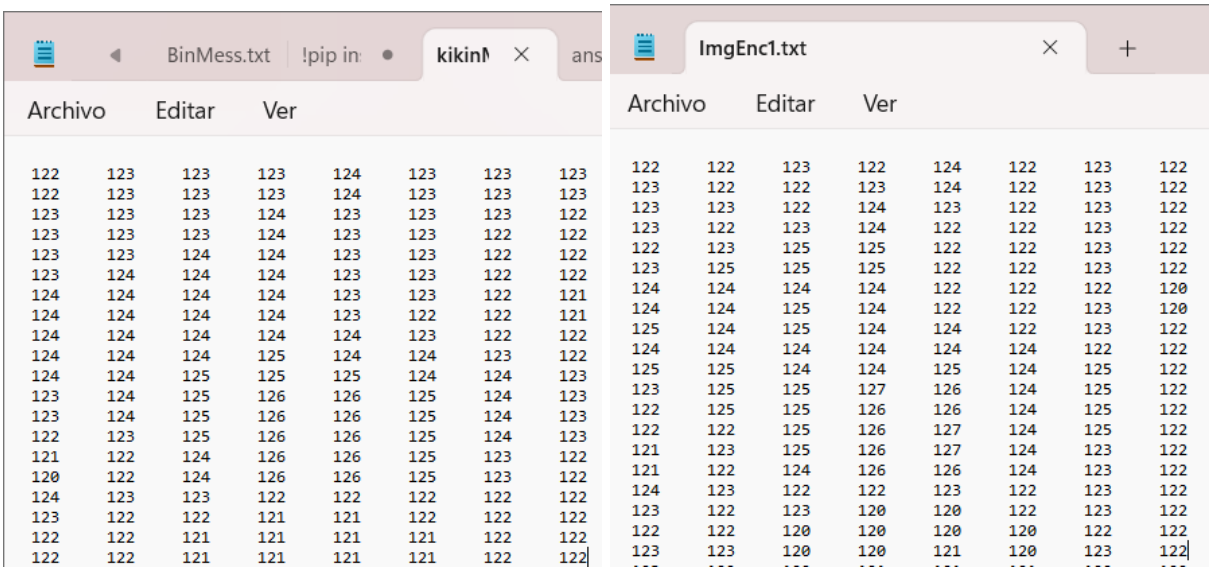
Capítulo 5

Evaluación de la funcionalidad del software

Primer ejemplo

```
146
147     key:string:="11001000"; --IMPORTANTE!!! AQUI SE CAMBIA LA LLAVE
148     chars:string:="DISENO DE SOFTWARE S"; --AQUI SE CAMBIA EL MENSAJE
149     keyTranslate:arr8;
150
```

Figura 5.1: Código en ADA con la llave y el mensaje a encriptar

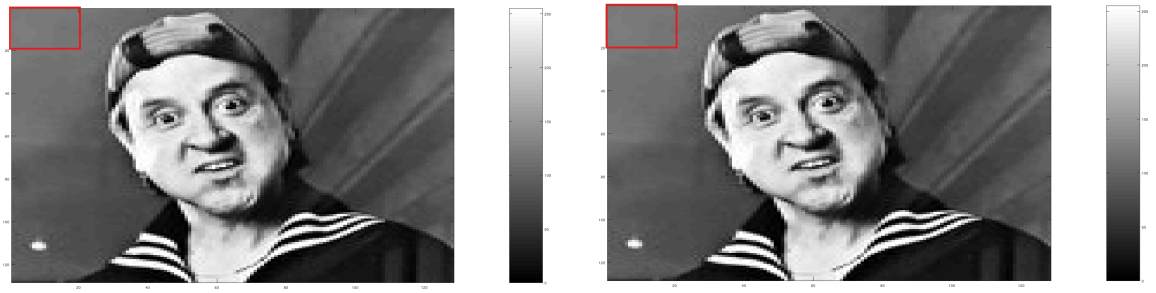


(a) Matriz original

(b) Matriz con mensaje

Figura 5.2: Comparación de matrices

La llave inicia arriba a la izquierda, se empieza su recorrido de abajo hacia la derecha con preferencia al renglon y es de 1 bit.



(a) Imagen original (b) Imagen con mensaje

Figura 5.3: Comparación de imágenes

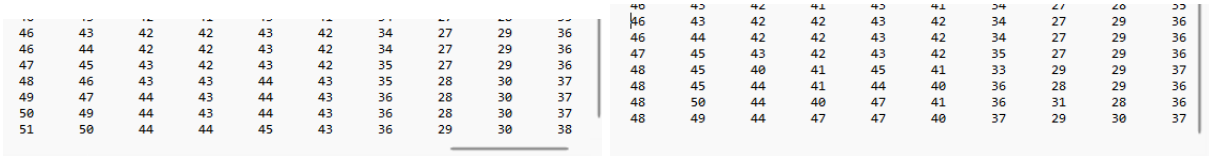
```
C:\Users\tuf\Desktop\UNAQ\DSS\DescifradoImage\obj\main.exe
DISENO DE SOFTWARE S
MENSAJE GUARDADO EN BinMess.txt
[2024-04-18 21:06:56] process terminated successfully, elapsed time: 00.21s
```

Figura 5.4: Mensaje desencriptado

Segundo ejemplo

```
key:string:="00110100"; --IMPORTANTE!!! AQUI SE CAMBIA LA LLAVE
chars:string:="ARMADOS Y PREPARADOS"; --AQUI SE CAMBIA EL MENSAJE
keyTranslate:arr8;
```

Figura 5.5: Código en ADA con la llave y el mensaje a encriptar



46	43	42	42	43	42	34	27	29	36
46	44	42	42	43	42	34	27	29	36
47	45	43	42	43	42	35	27	29	36
48	46	43	43	44	43	35	28	30	37
49	47	44	43	44	43	36	28	30	37
50	49	44	43	44	43	36	28	30	37
51	50	44	44	45	43	36	29	30	38

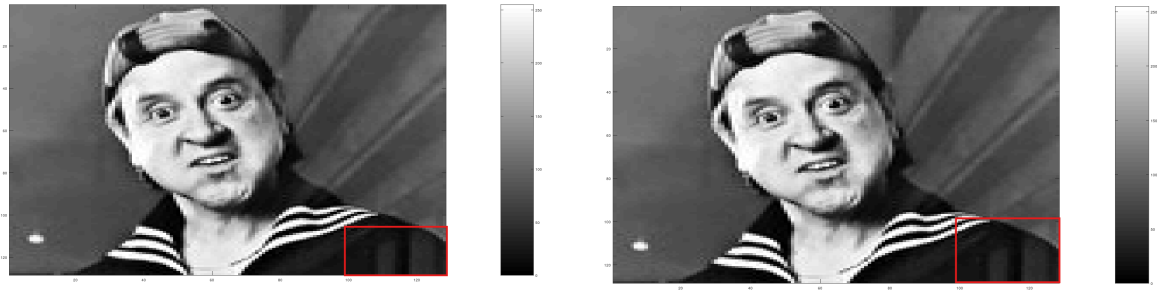
(a) Matriz original

46	43	42	42	43	42	34	27	29	36
46	44	42	42	43	42	34	27	29	36
47	45	43	42	43	42	35	27	29	36
48	45	40	41	45	41	33	29	29	37
48	45	44	41	44	40	36	28	29	36
48	50	44	40	47	41	36	31	28	36
48	49	44	47	47	40	37	29	30	37

(b) Matriz con mensaje

Figura 5.6: Comparación de matrices

La llave inicia abajo a la derecha, se empieza su recorrido de arriba hacia la izquierda con preferencia a la columna y es de 2 bits.



(a) Imagen original (b) Imagen con mensaje

Figura 5.7: Comparación de imágenes

BinMess.txt | !pip install roboflow | kikinMatrix.txt | ans.txt | BinaryMess.

Archivo Editar Ver

```
1 0 0 0 0 0 1 0
0 1 0 0 1 0 1 0
1 0 1 1 0 0 1 0
1 0 0 0 0 0 1 0
0 0 1 0 0 0 1 0
1 1 1 1 0 0 1 0
1 1 0 0 1 0 1 0
0 0 0 0 0 0 0 0
1 0 0 1 1 0 1 0
0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 0
0 1 0 0 1 0 1 0
1 0 1 0 0 0 1 0
0 0 0 0 1 0 1 0
1 0 0 0 0 0 1 0
0 1 0 0 1 0 1 0
1 0 0 0 0 0 1 0
0 0 1 0 0 0 1 0
1 1 1 1 0 0 1 0
1 1 0 0 1 0 1 0
ARMADOS Y PREPARADOS
```

Figura 5.8: Mensaje desencriptado

Capítulo 6

Modo de ejecución

Algorithm 1 Matriz en Matlab

1: Añadir la imagen en blanco y negro en la misma carpeta que el script y el código de ADA
2: Escribir los siguientes comandos en un script en Matlab u Octave:

```
        % Cargar la imagen
imagen = imread('nombre_de_la_imagen.jpg');
% Aseg rate de proporcionar la ruta y el nombre correctos de la imagen

% Convertir la imagen a una matriz
matriz_de_imagen = double(imagen);

% Guardar la matriz en un archivo de texto
dlmwrite('matriz_imagen.txt', matriz_de_imagen, 'delimiter', '\t');

disp('Matriz de imagen guardada en "matriz_imagen.txt"');
```

3: Ya tiene su matriz elaborada correctamente

Algorithm 2 Encriptación

1: Asegurarnos que el archivo .txt se añadió correctamente con la matriz
2: Ingresar manualmente la llave **en binario (0,1)** en procedure Encr dentro del codigo de ADA
3: Ingresar manualmente el mensaje a encriptar en **el caracter original (A,B,C,...)** en procedure Encr dentro del código de ADA
4: Correr el código
5: En caso de que la llave no sea válida, se le enviará un mensaje correspondiente para que vuelva a ingresarla
6: Obtendrá el .txt con la matriz del mensaje cifrado en binario.
7: Obtendrá un .csv con la matriz correspondiente al mensaje cifrado

Algorithm 3 Desencriptación

1: Asegurarnos que el archivo .txt se añadió correctamente con la matriz de la imagen con el mensaje encriptado
2: Ingresar manualmente la llave **en binario (0,1)** en procedure Decr dentro del codigo de ADA
3: Correr el código
4: En caso de que la llave no sea válida, se le enviará un mensaje correspondiente para que vuelva a ingresarla
5: Obtendrá el .txt con la matriz del mensaje descifrado en binario.

Capítulo 7

Resultados y Conclusiones

Los resultados presentados muestran que los programas funcionan de manera correcta, como se mostró en el ejemplo de corrida presentada. Se logró implementar con éxito un algoritmo de encriptación y desencriptación usando la técnica LSB en ADA que permite ocultar datos confidenciales dentro de los bits menos significativos de los píxeles de una imagen. La integración de matrices generadas en Matlab facilitó la manipulación eficiente de los datos de la imagen durante el proceso de encriptación y desencriptación en ADA.

Así mismo, el uso de una llave, permite que el usuario tenga más confianza en el proceso que se lleva a cabo. Pues al ser generada en el código, le da la opción de solo compartir esta información con las personas que necesiten tenerla, sin ser de llave publica como se usa en otros cifrados, de igual manera, nos da la certeza que solo el programador conocerá la llave en caso de ser necesario, cómo en muchos software se ocupa la "llave maestra".

Aunque durante el desarrollo del código han surgido inconvenientes, hemos sabido como resolverlos. Sin embargo, uno de ellos es que no podemos regresar a ver la imagen, por lo que el proyecto plantea un cimiento para que futuros proyectos puedan retomar el proyecto y puedan finalizarlo de manera correcta, haciendo uso completo del programa ADA.

Finalmente, podemos decir que el uso de este tipo de técnicas nos permite que podamos tener más alternativas al momento de cifrar, sabiendo que la información está llevando un proceso estructurado y que la protección de información se llevará de manera segura y eficiente. El ejemplo fue muy claro en este caso, donde la imagen cifrada no muestra cambios significativos en la encriptación, haciendolo invisible al ojo humano.

Bibliografía

Rgb2gray. (s/f). Mathworks.com. Recuperado el 17 de abril de 2024, de https://www.mathworks.com/help/matlab/ref/rgb2gray.html