INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES
DE MONTERREY, CAMPUS HIDALGO

PROYECTO FINAL – MANUAL TÉCNICO

SEGURIDAD INFORMÁTICA

ALEJANDRO RESÉNDIZ MARTÍNEZ, MARCOS ZÁRRAGA FLORES
A01273960, A01271642

A. Pantalla Principal.

HTML. Contiene los elementos visuales, manuales y descripción del proyecto. Elementos:

- Título de asignatura y proyecto.
- Descripción.
- Botones Insertar y Extraer.
- Campos de texto Mensaje y Representación Binaria.
- Canvas myCanvas y mySecretCanvas (invisibles).
- Manual Técnico.
- Manual del Usuario.

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/css/materia
lize.min.css">
    <title>Esteganografía LSB</title>
    <meta charset="UTF-8">
</head>
<body background="preview.jpg" >

    <div class="card-panel teal lighten-4">
        <div class='row' >
            <div class="col s1"> </div>
            <div class="col s10 ">
                <h3 >Seguridad Informática</h3>
                <h4 >Esteganografía</h4>
            </div>
        </div>
    </div>
<div class='row' >
        <div class="col s1"> </div>
        <div class="col s10 ">
            <div class="card-panel teal lighten-4">

                <div class="gray-text text-darken-4" id='text'>
                    <h4 class="left-align" >Algoritmo LSB</h4>
```

```
                    <p>
                        En este proyecto se prueba el algoritmo de
esteganografía LSB (Least Significant Bit), también conocido como método de
Sustitución. Consiste en leer cada pixel de una imagen digital y desglosarlo
en bits; una vez teniendo esta información, se modifica el ultimo bit de la
secuencia de colores para poder ocultar el mensaje en la imagen. De esta
manera, el ojo humano no puede diferenciar a simple vista las modificaciones
realizadas a la imagen original.
                    </p>
                    <!--hr/-->
                </div>
            </div>
        </div>
    </div>

        <div class="row">
                <div class="col s1"> </div>
                <div class="col s10">

                <form action="#">
                    <div class="file-field input-field">
                        <div class="btn">
                            <span>Imagen</span>
                            <input type="file" id="myImage">
                        </div>
                        <div class="file-path-wrapper">
                            <input class="file-path validate" type="text">
                        </div>
                    </div>
                </form>

                <canvas id="myCanvas" width="256" height="256"
style='display:none;'></canvas>
                <!--br-->
                <canvas id="mySecretCanvas" width="256" height="256"
style='display:none;'></canvas>
                <a id="mySecretDownload" style='display:none;'>Secret</a>

            </div>
        </div>

        <div class="row">
            <div class="col s1"> </div>
            <div class="col s10 card-panel teal lighten-4">
                <span class="flow-text"> Mensaje:</span>
```

```
                <br>
                    <textarea id="myMessage" rows="7" cols="100">
</textarea>
                    <!--hr/-->
            </div>
            </div>

        <div class="row">
            <div class="col s1"> </div>
            <div class="col s10">
                <button id="hideMessage" class="btn waves-effect waves-
light">  Insertar   </button>
                <button id="showMessage" class="btn waves-effect waves-
light">  Extraer </button>
            </div>
        </div>

        <div class="row">
            <div class="col s1"> </div>
            <div class="col s10 card-panel teal lighten-4">
                <span class="flow-text"> Representación Binaria:</span>
                <br>
                <textarea id="myBinMessage" rows="7" cols="100"> </textarea>
                <hr/>
            </div>
        </div>
</body>
<footer>
    <div class="card-panel teal lighten-4">
        <div class="row valign-wrapper">

            <div class="col s4 left-align">

                    <a href="ManualTecnico.docx">Manual Tecnico</a>
                    <br>

            </div>
            <div class="col s4 left-align">
                    <p>
                        Alejandro Reséndiz Martínez - A01273960
                        <br>
                        Marcos Zárraga Flores - A01271642
                    </p>
            </div>
            <div class="col s1"></div>
```

```
        <div>
    </div>
</footer>
</html>
```

B. Script.

Javascript. Contiene la programación del algoritmo LSB, cuyo código está debidamente comentado. Elementos:

- Variables globales.
- Funciones genéricas.
- Funciones de los botones.
- Funciones LSB.

```
<script>
        // VARIABLES GLOBALES
        var myMsgBox = document.getElementById('myMessage');
        var myBinMsgBox = document.getElementById('myBinMessage');
        var myImg = document.getElementById('myImage');
        var mySecretLink = document.getElementById('mySecretDownload');
        var myCtx = myCanvas.getContext('2d');
        var mySecretCtx = mySecretCanvas.getContext('2d');
        var myImgArray;
        var myMsgArray;
        var myTxtMessage = new String();
        var index = 0;

        // FUNCIONES GENERICAS
        // UInt8; arreglo que almacena hasta 256 valores (caracteres)
        function stringToUint(string) {
            ui =new Uint8Array(string.length);
            for(var i=0,j=string.length;i<j;++i){
                ui[i]=string.charCodeAt(i);
            }
            return ui;
        }

        // DecodeUtf8; obtener caracteres UTF-8 a partir de ArrayBuffer
        function decodeUtf8(arrayBuffer) {
            var result = "";
            var i = 0;
```

```
            var c = 0;
            var c1 = 0;
            var c2 = 0;

            var data = new Uint8Array(arrayBuffer);

            // If we have a BOM skip it
            if (data.length >= 3 && data[0] === 0xef && data[1] === 0xBB &&
data[2] === 0xBF) {
                i = 3;
            }

            while (i < data.length) {
                c = data[i];

                if (c < 128) {
                    result += String.fromCharCode(c);
                    i++;
                } else if (c > 191 && c < 224) {
                    if (i + 1 >= data.length) {
                        throw "UTF-8 Decode failed. Two byte character was
truncated.";
                    }
                    c2 = data[i + 1];
                    result += String.fromCharCode(((c & 31) << 6) | (c2 &
63));
                    i += 2;
                } else {
                    if (i + 2 >= data.length) {
                        throw "UTF-8 Decode failed. Multi byte character was
truncated.";
                    }
                    c2 = data[i + 1];
                    c3 = data[i + 2];
                    result += String.fromCharCode(((c & 15) << 12) | ((c2 &
63) << 6) | (c3 & 63));
                    i += 3;
                }
            }
            return result;
        }

        // Canvas; descarga de imagen
        function downloadCanvas(link, canvasId, filename) {
            link.href = document.getElementById(canvasId).toDataURL();
```

```
            link.download = filename;
        }

        document.getElementById('mySecretDownload').addEventListener('click'
, function() {
            downloadCanvas(this, 'mySecretCanvas', 'Secret.png');
        }, false);

        // Clear; reiniciar variables globales
        myImg.addEventListener('click', function (e) {
            myMsgBox.value = "";
            myBinMsgBox.value = "";
            index=0;
            myTxtMessage = 0;
            myCanvas.width = myCanvas.width;
            mySecretCanvas.width = mySecretCanvas.width;
        });

        // FUNCIONES DE LOS BOTONES
        // TextBox de Mensaje
        myMessage.addEventListener('input', function(e) {
            if (this.value.length   < 256 ) {
                myTxtMessage = this.value;
            } else {
                alert("¡Mensaje demasiado largo!");
                this.value  = "";
            }
        });

        // Boton Insertar Mensaje
        hideMessage.addEventListener('click', function(e) {
            var file = myImg.files[0];
            var fr = new FileReader();


            if(file != null) {
                if (file.width > 256 || file.length > 256) {
                    alert("¡La imagen no cumple con las especificaciones de
tamaño!");
                } else {
                    if (myTxtMessage != "") {
                        console.log("Longitud del MSg: " +
myTxtMessage.length)
                        if (myTxtMessage.length < 256) {
                        fr.addEventListener( "load", loadEvent );
```

```
                        fr.addEventListener( "loadend", loadEndEvent );

                        function loadEvent ( e ) {
                            console.info( 'Cargando... esperar un momento.'
);
                        }

                        function loadEndEvent ( e ) {
                            console.info('Carga lista.');
                            var img = new Image();
                            img.src = e.target.result;
                            // Cargar imagen
                            img.onload = function() {
                                // Dibujando imagen en Canvas (invisible)
                                myCtx.drawImage(img, 0, 0);
                                // Obteniendo los valores RGB
                                myImgArray = myCtx.getImageData(0, 0,
myCanvas.height, myCanvas.width);
                                console.log(myImgArray);
                                // Leyendo y reemplazando bits para ocultar
el Mensaje
                                myMsgArray = stringToUint(myTxtMessage);

                                for(var i=0; i < myMsgArray.length; i++) {
                                myBinMsgBox.value +=
(myMsgArray[i].toString(2) + " ");
                                }

                                console.log(myMsgArray);
                                readByte(myMsgArray);
                                // Dibujando imagen usando la matriz de
bytes
                                mySecretCtx.putImageData( myImgArray, 0, 0
);
                                // Descargando Imagen
                                mySecretLink.click();

                                console.log('Done.');

                            }
                        }
                        fr.readAsDataURL(file);
                        } else {
                            alert("¡Texto demasiado largo!");
```

```
                            //console.log("Longitud del MSg: " +
myTxtMessage.length)
                            myTxtMessage="";
                            myMsgBox.value="";
                        }
                    }
                    else {
                        alert("¡Insertar texto primero!");
                    }
                }
            } else {
                alert("¡Insertar imagen primero!");
            }
        });

        // Boton extraer Mensaje
        showMessage.addEventListener('click', function(e){
            var file = myImg.files[0];
            var fr = new FileReader();

            if (file != null) {
                fr.addEventListener('loadend', loadEndEvent);

                function loadEndEvent(e) {
                    // Carga de imagen
                    var img = new Image();
                    img.src = e.target.result;
                    img.onload = function() {
                        // Dibujar imagen en el canvas (invisible)
                        mySecretCtx.drawImage(img, 0, 0);
                        // Objeto ArrayBuffer obtenido a partir de la Imagen

                        var loadView = mySecretCtx.getImageData(0, 0,
mySecretCanvas.height, mySecretCanvas.width);
                        console.log(loadView);

                        var totalLength = 0;
                        var lastIndex;
                        // Recorrer todos los bits de los pixeles
                        // Obtener la longitud del mensaje secreto
                        for(var b=0, viewLength = loadView.data.length; b <
viewLength; b++) {
                            if (loadView.data[ b ] == 255) {
                                totalLength += loadView.data[ b ];
                                if (loadView.data[ b + 1 ] < 255) {
```

```
                                totalLength += loadView.data[ b + 1 ];
                                lastIndex = b + 1;
                                break;
                            }
                        } else {
                            totalLength += loadView.data[ b ];
                            lastIndex = b;
                            break;
                        }
                    }
                    console.info( 'Total length :' + totalLength + ',
Last Index : ' + lastIndex )
                    // Obtener la longitud del mensaje secreto
                    var secretLength = totalLength;
                    // Generar un arreglo Uint8, dividido entre cuatro
ya que un caracter equivale a 8 bits
                    var newUint8Array = new Uint8Array( totalLength / 4
);
                    var j = 0;
                    // Extrayendo los bits del pixel
                    for ( var i = ( lastIndex + 1 ); i < secretLength; i
= i + 4 ) {
                        // Solo se necesitan 2 bits de casa byte, los
que contienen el mensaje secreto
                        // Primero, se eliminan los bits sobrantes con
la mask(3) == 0000 0011
                        // Después, corrimiento a la izquierda por cada
bit
                        var aShift = ( loadView.data[ i ] & 3 );
                        var bShift = ( loadView.data[ i + 1 ] & 3 ) <<
2;
                        var cShift = ( loadView.data[ i + 2 ] & 3 ) <<
4;
                        var dShift = ( loadView.data[ i + 3 ] & 3 ) <<
6;
                        // finalmente, combinar los bits modificados
para formar un byte (8 bits)
                        var result = ( ( ( aShift | bShift) | cShift ) |
dShift );
                        // almacenar el byte en un arreglo Uint (sin
signo)
                        newUint8Array[ j ] = result;
                        j++;
                    }
                    console.log(newUint8Array);
```

```
                        // Decodificar el arreglo Uint8 para obtener el
texto representado en ASCII
                        var result = decodeUtf8( newUint8Array );
                        myBinMsgBox.value="";

                        for(var i=0; i< newUint8Array.length; i++) {
                            myBinMsgBox.value +=
(newUint8Array[i].toString(2) + " ");
                        }
                        //myBinMsgBox.value = newUint8Array;

                        console.log( result )
                        // Imprimir el texto en la caja de Texto
                        myMsgBox.value="";
                        myMsgBox.value=result;

                    }
                }
                fr.readAsDataURL(file);
            } else {
                alert("¡Inserta una imagen!");
            }

        });

        // FUNCIONES LSB
        // ReadByte; Insertando la informacion del mensaje en la Matriz de
Img
        function readByte( secret ) {
            for ( var i = 0, length = secret.length; i < length; i++ ) {
                if ( i == 0 ) {
                    // En el primer bit se almacena la longitud del mensaje,
el cual debe ser multiplo de 4
                    // El codigo de un caracter contiene 8 bits, los cuales
se dividen en grupos de 2
                    // Cada grupo de 2 bits reemplaza el LSB(Least
significant bit) del byte del pixel
                    var secretLength = length * 4;
                    console.info( 'Secret Length(' + length + 'x4) : ' +
secretLength )
                    // La informacion de la imagen se almacena en un arreglo
(Uint8 myImgArray)
                    // Solo puede almacenar un valor maximo de 256
caracteres (8bit or 1byte)
                    if ( secretLength > 255 ) {
```

```
                        // Calculando el numero de ciclos (en funcion de los
caracteres)
                        // Almacenar la longitud del mensaje
                        var division = secretLength / 255;
                        // Entero
                        if ( division % 1 === 0 ) {
                            for ( var k = 0; k < division; k++ ) {
                            myImgArray.data[ k ] = 255;
                            index++;
                            }
                        }
                        // Flotante
                         else {
                            var firstPortion =
division.toString().split(".")[ 0 ];
                            var secondPortion =
division.toString().split(".")[ 1 ];

                            for ( var k = 0; k < firstPortion; k++ ) {
                                myImgArray.data[ k ] = 255;
                                index++;
                            }

                            var numberLeft = Math.round( ( division -
firstPortion ) * 255 );
                            console.info( 'numberLeft : ' + numberLeft )
                            myImgArray.data[ k ] = numberLeft;
                            index++;
                        }

                    } else {
                        myImgArray.data[ 0 ] = secretLength;
                        index++;
                    }

                   console.log( 'sss : ' + myImgArray.data[ 0 ] )

                }

                var asciiCode = secret[ i ];
                // Usando masking, para limpiar los bits, tomando solo los
que se necesitan
                // Tomando los primeros 2 bits, ej. : 0111 0011 => 0000 0011
                var first2bit = ( asciiCode & 0x03 ); // 0x03 = 3
```

```
                // Tomando solo los primeros  4 bits (2bits al final), e. :
0111 0011 => 0000 0000
                var first4bitMiddle = ( asciiCode & 0x0C ) >> 2; // 0x0C =
12, cambiar a la derecha 2 bit o dividir entre 2^2, tomando primero los 2
bits al final
                // Tomando solo los primeros 6 bits (2bits al final), ej. :
0111 0011 => 0011 0000
                var first6bitMiddle = ( asciiCode & 0x30 ) >> 4; // 0x30 =
48, cambiar a la derecha 4 bit o dividir entre 2^4, tomando primero los 2
bits al final
                // Tomando los primeros  8 bits (2bit al final), ej. : 0111
0011 => 0100 0000
                var first8bitMiddle = ( asciiCode & 0xC0 ) >> 6; // 0xC0 =
192, cambiar a la derecha 6 bit o dividir entre 2^6, tomando primero los 2
bits al final
                //console.log(i + ' : ' + first2bit);
                //console.log(i + ' : ' + first4bitMiddle);
                //console.log(i + ' : ' + first6bitMiddle);
                //console.log(i + ' : ' + first8bitMiddle);
                // Reemplazando el bit del mensaje secreto en el LSB
                replaceByte( first2bit );
                replaceByte( first4bitMiddle );
                replaceByte( first6bitMiddle );
                replaceByte( first8bitMiddle );
            }
        }

        // Replacebyte, sustituye los bits en la matriz de la imagen
        function replaceByte ( bits ) {
            // Eliminar los 2 primeros bits y sustituyendolos por los del
mensaje
            myImgArray.data[ index ] = ( myImgArray.data[ index ] & 0xFC ) |
bits;
            index++;
        }

    </script>
```

C. Referencias del código:

El proyecto de clase estaba realizado en MatLab. Debido a la imposibilidad de incluirlo en una página web de una manera sencilla, se investigó la implementación LSB en Javascript.

- https://jsfiddle.net/norlihazmeyGhazali/quo42a2n/
- http://ciaranj.blogspot.my/2007/11/utf8-characters-encoding-in-javascript.html
- https://stackoverflow.com/questions/6965107/converting-between-strings-and-arraybuffers
- https://jsperf.com/string-to-uint8array