Apéndice B: Código en Matlab de la simulación del WEP.

Este apéndice muestra el código elaborado en Matlab para crear la interfase gráfica que ejecuta la simulación del WEP.

```
% este código se crea automáticamente para inicializar los valores de la interfase
% gráfica

function varargout = untitled1(varargin)



gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @untitled1_OpeningFcn, ...
                   'gui_OutputFcn',  @untitled1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end

% finaliza el código de inicialización

% el código siguiente llama a las imágenes JPG que se muestran en la interfase

set(hObject, 'Units', 'pixels');
handles.intruso = imread(['intruso.jpg']); % Read the image file banner.jpg
info = imfinfo(['intruso.jpg']); % Determine the size of the image file
position = get(hObject, 'Position');

axes(handles.axes1);
image(handles.intruso)
set(handles.axes1, ...
   'Visible', 'off', ...
   'Units', 'pixels', ...
   'Position', [560 10 info.Width info.Height]);

set(hObject, 'Units', 'pixels');
handles.compu_izq = imread(['compu_izq.jpg']); % Read the image file banner.jpg
info = imfinfo(['compu_izq.jpg']); % Determine the size of the image file
position = get(hObject, 'Position');

axes(handles.axes2);
image(handles.compu_izq)
set(handles.axes2, ...
   'Visible', 'off', ...
   'Units', 'pixels', ...
   'Position', [50 460 info.Width info.Height]);
```

```
set(hObject, 'Units', 'pixels');
handles.compu_derecha = imread(['compu_derecha.jpg']); % Read the image file banner.jpg
info = imfinfo(['compu_derecha.jpg']); % Determine the size of the image file
position = get(hObject, 'Position');

axes(handles.axes4);
image(handles.compu_derecha)
set(handles.axes4, ...
   'Visible', 'off', ...
   'Units', 'pixels', ...
   'Position', [625 460 info.Width info.Height]);

set(hObject, 'Units', 'pixels');
handles.bluepp = imread(['bluepp.jpg']); % Read the image file banner.jpg
info = imfinfo(['bluepp.jpg']); % Determine the size of the image file
position = get(hObject, 'Position');

axes(handles.axes5);
image(handles.bluepp)
set(handles.axes5, ...
   'Visible', 'off', ...
   'Units', 'pixels', ...
   'Position', [380 180 info.Width info.Height]);

set(hObject, 'Units', 'pixels');
handles.puntos = imread(['puntos.jpg']); % Read the image file banner.jpg
info = imfinfo(['puntos.jpg']); % Determine the size of the image file
position = get(hObject, 'Position');

axes(handles.axes6);
image(handles.puntos)
set(handles.axes6, ...
   'Visible', 'off', ...
   'Units', 'pixels', ...
   'Position', [440 90 info.Width info.Height]);

set(hObject, 'Units', 'pixels');
handles.wep2 = imread(['wep2.jpg']); % Read the image file banner.jpg
info = imfinfo(['wep2.jpg']); % Determine the size of the image file
position = get(hObject, 'Position');

axes(handles.axes9);
image(handles.wep2)
set(handles.axes9, ...
   'Visible', 'off', ...
   'Units', 'pixels', ...
   'Position', [250 400 info.Width info.Height]);

% finaliza el código que llama a las imágenes



function edit1_Callback(hObject, eventdata, handles)
% esta función se crea automáticamente al agregar un campo de texto en la
% interfase, este campo es en el cual se ingresa el texto a encriptar


% en la variable plain_text se guarda el texto a encriptar
handles.plain_text = get(hObject,'String')
```

```
guidata(hObject, handles); % Save the updated structure

% esta función se ejecuta durante la creación del objeto después de establecer sus
% propiedades
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



% la siguiente función incluye el código que se ejecutara al presionar el botón
% encriptar
function pushbutton1_Callback(hObject, eventdata, handles)


handles.plainASCII = double(handles.plain_text)  %convierte la cadena ingresada por el usuario en
caracteres ASCII


%inicialización de la llave S, RC4

% long_K es el número de bytes con el que se forma el vector K
% mismo que se repetirá tantas veces como sea necesario
% para llenar el vector temporal T

for i = 0:(handles.long_K-1)
   K(i+1) = i;            %generación del vector K
end

S = [0:255];

for j=0:255                % se inicializa S con valores de 0 a 255
   y = mod(j,handles.long_K);    % se repite el vector K tantas veces como sea
   T(j+1) = K(y+1);          % necesario para genera el vector T
end

% se inicializa el vector S
k = 0;
for m=0:255
   z = (k + S(m+1) + T(m+1));
   k = mod(z,256);
   temp = S(m+1);
   S(m+1) = S(k+1);
   S(k+1) = temp;
end


size_plain = size(handles.plainASCII);
handles.size_plain = size_plain(2);  % estas instrucciones obtienen la longitud del texto ingresado


% el siguiente algoritmo genera la secuencia de llaves de encriptación

i=0;
k=0;
```

```
for j = 1:handles.size_plain

   q = i+1;
   i = mod(q,256);
   p = k + S(q);
   k = mod(p,256);
   temp = S(i+1);
   S(i+1) = S(k+1);
   S(k+1) = temp;
   r = (S(i+1) + S(k+1));
   t = mod(r,256);
   handles.Keys(j) = S(t+1);
   handles.cipher(j) = bitxor(handles.Keys(j), handles.plainASCII(j)); %esta sentencia
%realiza la operación XOR entre las llaves generadas y el texto a encriptar

end

% las siguientes instrucciones envían las cadenas de texto original, llaves y cifrado
% a cuadros de texto en la interfase visual
set(handles.ASCIIplain,'String',num2str(handles.plainASCII))
set(handles.Llaves_enc,'String',num2str(handles.Keys))
set(handles.texto_cifrado,'String',num2str(handles.cipher))

% esta función genera el cuadro de texto donde se mostrará el texto cifrado
function texto_cifrado_Callback(hObject, eventdata, handles)




% esta función se ejecuta durante la creación del cuadro de texto

function texto_cifrado_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% función creada para recibir la longitud de la llave K
function bytesK_Callback(hObject, eventdata, handles)

handles.long_K = str2double(get(hObject,'String')); % la longitud de la llave K
% se almacena  en la variable long_K

% función creada durante la creación del objeto
function bytesK_CreateFcn(hObject, eventdata, handles)

if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end




% funciones creadas para mostrar el texto ASCII de la cadena ingresada en
% interfase
function ASCIIplain_Callback(hObject, eventdata, handles)
```

```
function ASCIIplain_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



%estas funciones crean el cuadro de texto que muestra la secuencia de llaves
%en la interfase
function Llaves_enc_Callback(hObject, eventdata, handles)



function Llaves_enc_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



% esta función ejecuta el código siguiente al presionar el botón Decriptar
function pushbutton2_Callback(hObject, eventdata, handles)


% este ciclo realize la operación XOR entre las llaves y el texto cifrado para
% recuperar
% el texto original
for j = 1:handles.size_plain

   plain_text_recuperado(j) = bitxor(handles.Keys(j), handles.cipher(j));

end

% las tres instrucciones siguientes envían el texto recuperado, las llaves
% y el texto cifrado a los cuadros de texto de la interfase
set(handles.Texto_recuperado,'String',num2str(plain_text_recuperado))
set(handles.Llaves_enc_dec,'String',num2str(handles.Keys))
set(handles.texto_cifrado_dec,'String',num2str(handles.cipher))

cadena_texto_recuperado = char(plain_text_recuperado);

set(handles.cadena_rec,'String',cadena_texto_recuperado)

function Llaves_enc_dec_Callback(hObject, eventdata, handles)


% estas funciones se ejecutan al crear el cuadro de texto que mostrará las
% llaves de encriptacion
function Llaves_enc_dec_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
```

```
end

function texto_cifrado_dec_Callback(hObject, eventdata, handles)

% estas funciones se ejecutan al crear el cuadro de texto que mostrará el
% texto cifrado

function texto_cifrado_dec_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% las siguientes funciones se ejecutan al crear el cuadro de texto que
% mostrará el texto ASCII recuperado por el receptor
function Texto_recuperado_Callback(hObject, eventdata, handles)


function Texto_recuperado_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% las siguientes funciones se ejecutan al crear el cuadro de texto que
% mostrará la cadena de texto recuperado por el receptor


function cadena_rec_Callback(hObject, eventdata, handles)



function cadena_rec_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```