**Main File:**

function [output] = steganography()

%% STEP 1: Determine Whether User is Encoding or Decoding a Message

enc\_dec = input('Welcome to the Steganography Program \nEnter 1 for Encoding, 2 for Decoding:\n');

if enc\_dec == 1

%% STEP A: ENCODING VERSION

%% STEP 2A: Select "Canvas Image" and "Message File".

% First Get Image.

[FileName,PathName] = uigetfile({'\*.jpg';'\*.png';'\*.gif';'\*.bmp'},'Select "Canvas Image" to Hide Message.');

img = imread( strcat(PathName,FileName) );

% Next get Message File

msg\_type = input('Enter 1 for TEXT Message: \n');

if msg\_type == 1

[FileName,PathName] = uigetfile('\*.txt','Select TEXT MESSAGE.');

testmsg = fopen( strcat(PathName,FileName) );

[msg] = fscanf(testmsg,'%c');

else

error('Invalid Message Type Selection');

end

%% STEP 3A: Prompt User for Encryption Key

enc\_key = input('Please Enter an Encryption Key Between 0 - 255:\n');

if enc\_key < 0 || enc\_key > 255

error('Invalid Key Selection');

end

enc\_key = uint8(enc\_key);

%% STEP 4A: Allow User to Select Encoding

encode = input('Enter 1 for Encoding\n');

if encode == 1

output = stegancoder(img,msg,enc\_key);

else

error('Invalid Encoding Selection');

end

%% STEP 5A: Write Canvas Image to .BMP File

% BMP was chosen because it DOES NOT use

% compression. JPEG compression destroys the message.

secfn = input('Enter File Name for Image + Message:\n','s');

nametest = ischar(secfn);

if nametest == 1

imwrite(output,strcat(secfn,'.bmp'));

else

error('Invalid File Name');

end

elseif enc\_dec == 2

%% STEP B: DECODING VERSION

%% STEP 2B: Import "Canvas Image" With Hidden Message.

[FileName,PathName] = uigetfile('\*.bmp','Select "Canvas Image" With Hidden Message.');

img = imread( strcat(PathName,FileName) );

%% STEP 3B: Prompt User for Encryption Key

enc\_key = input('Please Enter an Encryption Key Between 0 - 255:\n');

if enc\_key < 0 || enc\_key > 255

error('Invalid Key Selection');

end

enc\_key = uint8(enc\_key);

%% STEP 4B: Allow User to Select Decoding

decode = input('Enter 1 Decoding\n');

if decode == 1

%DECODING: This only needs an Encryption Key Input.

output = stegandecoder(img,enc\_key);

else

error('Invalid Encoding Selection');

end

%% STEP 5B: Writing Message to .TXT or .JPG File

secfn = input('Enter File Name for Image + Message:\n','s');

nametest = ischar(secfn);

if nametest == 1

msgtest = ischar(output);

if msgtest == 1

% TEXT Message CASE

fid = fopen(strcat(secfn,'.txt'),'w');

fwrite(fid,output,'char');

fclose(fid);

else

% IMAGE Message CASE

imwrite(output,strcat(secfn,'.bmp'));

end

else

error('Invalid File Name');

end

else

error('Invalid Selection');

end

**Decoding Code:**

function [msg] = stegandecoder(img,enc\_key)

%% Step 1a: Recover Header Set

rm = 1; gm = 1; bm = 1; % Initializing Counters

rn = 1; gn = 1; bn = 1;

header = [];

[maxM, maxN, chan] = size(img);

for z = 1:8;

temp = zeros(1,8);

% Red

temp(1,1) = mod(img(rm,rn,1),2);

rm = rm + 1;

if rm > maxM

rn = rn + 1;

rm = 1;

if rn > maxN

break

end

end

% Green

temp(1,2) = mod(img(gm,gn,2),2);

gm = gm + 1;

if gm > maxM

gn = gn + 1;

gm = 1;

if gn > maxN

break

end

end

% Blue

temp(1,3) = mod(img(bm,bn,3),2);

bm = bm + 1;

if bm > maxM

bn = bn + 1;

bm = 1;

end

% Blue

temp(1,4) = mod(img(bm,bn,3),2);

bm = bm + 1;

if bm > maxM

bn = bn + 1;

bm = 1;

end

% Green

temp(1,5) = mod(img(gm,gn,2),2);

gm = gm + 1;

if gm > maxM

gn = gn + 1;

gm = 1;

if gn > maxN

break

end

end

% Red

temp(1,6) = mod(img(rm,rn,1),2);

rm = rm + 1;

if rm > maxM

rn = rn + 1;

rm = 1;

if rn > maxN

break

end

end

% Red

temp(1,7) = mod(img(rm,rn,1),2);

rm = rm + 1;

if rm > maxM

rn = rn + 1;

rm = 1;

if rn > maxN

break

end

end

% Green

temp(1,8) = mod(img(gm,gn,2),2);

gm = gm + 1;

if gm > maxM

gn = gn + 1;

gm = 1;

if gn > maxN

break

end

end

tempstr = num2str(temp);

header = vertcat(header,tempstr);

end

%% Step 1b: Header Analysis - Decrypt and Determine Message Dimensions

% key = 42; % Used for Test Phase

msg\_head\_set = bin2dec(header);

temp\_head = bitxor(uint8(msg\_head\_set),uint8(enc\_key));

if temp\_head(1) == 116

% CASE 1: Text Set

dim1 = char(temp\_head(2:8));

m = str2double(dim1);

n = 1;

else

% CASE 2: Image Set

% Determine Dimensions from Header Values

tempm = char(temp\_head(1:4));

tempn = char(temp\_head(5:8));

m = str2double(tempm');

n = str2double(tempn');

end

%% Step 2: Isolate Potential Message

z = 0;

enc\_msg = [];

stopmax = (m \* n);

for z = 1:stopmax

temp = zeros(1,8);

% Red

temp(1,1) = mod(img(rm,rn,1),2);

rm = rm + 1;

if rm > maxM

rn = rn + 1;

rm = 1;

if rn > maxN

break

end

end

% Green

temp(1,2) = mod(img(gm,gn,2),2);

gm = gm + 1;

if gm > maxM

gn = gn + 1;

gm = 1;

if gn > maxN

break

end

end

% Blue

temp(1,3) = mod(img(bm,bn,3),2);

bm = bm + 1;

if bm > maxM

bn = bn + 1;

bm = 1;

end

% Blue

temp(1,4) = mod(img(bm,bn,3),2);

bm = bm + 1;

if bm > maxM

bn = bn + 1;

bm = 1;

end

% Green

temp(1,5) = mod(img(gm,gn,2),2);

gm = gm + 1;

if gm > maxM

gn = gn + 1;

gm = 1;

if gn > maxN

break

end

end

% Red

temp(1,6) = mod(img(rm,rn,1),2);

rm = rm + 1;

if rm > maxM

rn = rn + 1;

rm = 1;

if rn > maxN

break

end

end

% Red

temp(1,7) = mod(img(rm,rn,1),2);

rm = rm + 1;

if rm > maxM

rn = rn + 1;

rm = 1;

if rn > maxN

break

end

end

% Green

temp(1,8) = mod(img(gm,gn,2),2);

gm = gm + 1;

if gm > maxM

gn = gn + 1;

gm = 1;

if gn > maxN

break

end

end

tempstr = num2str(temp);

enc\_msg = vertcat(enc\_msg,tempstr);

end

%% Step 3: Decryption Step

msg\_dec\_set = bin2dec(enc\_msg);

msg\_dec = bitxor(uint8(msg\_dec\_set),uint8(enc\_key));

% msg\_dec\_set = dec2bin(msg\_dec,8);

%% Step 4: Message Prep

if temp\_head(1) == 116

% CASE 1: Text Set

msg\_set = msg\_dec;

msg\_out = char(msg\_set');

else

% CASE 2: Image Set

% Determine Dimensions from Header Values

tempm = char(temp\_head(1:4));

tempn = char(temp\_head(5:8));

m = str2double(tempm');

n = str2double(tempn');

% Reshape Message Set into an Image Output

msg\_set = msg\_dec;

count = 1;

msg\_out = uint8(zeros(m,n));

for y = 1:m

for x = 1:n

msg\_out(y,x) = msg\_set(count);

count = count + 1;

end

end

msg\_out = im2uint8(msg\_out);

end

%% Step 5: Final Output

msg = msg\_out;

end

**Encoding Code:**

function [J] = stegancoder(img,msg,enc\_key)

msgtype = ischar(msg); % If message is text this will be true;

% false otherwise

if msgtype == 1 % Message = TEXT

msg\_temp = double(msg); % Converts from ASCII to Integer Values.

msg\_dim = num2str(length(msg\_temp));

msg\_length = length(msg\_dim);

z = 0;

if msg\_length < 7

padtext = 7 - msg\_length;

for z = 1:padtext

msg\_dim = horzcat('0',msg\_dim);

end

msg\_head = horzcat('t',msg\_dim);

% Applying Header To Beginning of Message to be Encoded.

msg\_temp\_head = horzcat(msg\_head,msg\_temp);

end

else

% Message = IMAGE

msg = im2uint8(msg); % Convert to Integer Value Representation.

msg\_temp = rgb2gray(msg); % Converts Hidden Message to Grayscale.

% Reduces Amount of Data to Hide.

% Determine Message Image's Size for Encoding in Header

[hideM1,hideN1] = size(msg\_temp);

hideM = num2str(hideM1);

hideN = num2str(hideN1);

dimM = length(hideM);

dimN = length(hideN);

padM = 0; padN = 0;

z = 0;

if dimM < 4

padM = 4 - dimM;

for z = 1:padM

% Zero Padding Dimension if less than 4 Sig Figs.

hideM = horzcat('0',hideM);

end

end

z = 0;

if dimN < 4

padN = 4 - dimN;

for z = 1:padN

% Zero Padding Dimension if less than 4 Sig Figs.

hideN = horzcat('0',hideN);

end

end

msg\_head = horzcat(hideM,hideN);

msg\_temp\_head = msg\_head;

y = 0; k = hideM1;

for y = 1:k

% Applying Header To Beginning of Message to be Encoded.

msg\_temp\_head = horzcat(msg\_temp\_head,msg\_temp(y,:));

end

end

%% Step 3: Encrypting Using XOR Key

% key = 42; % Used for Test Phase

msg\_enc = bitxor(uint8(msg\_temp\_head),uint8(enc\_key));

msg\_enc\_set = dec2bin(msg\_enc, 8);

%% Step 4: Preparing Hiding Canvas

img\_prep = im2uint8(img);

%% Step 5: Hiding Data

rm = 1; gm = 1; bm = 1; % Initializing Counters

rn = 1; gn = 1; bn = 1;

[maxM,maxN] = size(img\_prep);

z = 0;

% RUN\_TIME Variable indicates the number of Message "Words" that need to be

% encoded in the IMG\_PREP "Canvas" Image.

run\_time = length(msg\_enc\_set);

for z = 1:run\_time;

temp\_code = msg\_enc\_set(z,:);

% Bit 1: Red

if str2double(temp\_code(1)) == 0

img\_prep(rm,rn,1) = bitand(img\_prep(rm,rn,1),uint8(254));

else

img\_prep(rm,rn,1) = bitor(img\_prep(rm,rn,1),uint8(1));

end

rm = rm + 1;

if rm > maxM

rn = rn + 1;

rm = 1;

end

% Bit 2: Green

if str2double(temp\_code(2)) == 0

img\_prep(gm,gn,2) = bitand(img\_prep(gm,gn,2),uint8(254));

else

img\_prep(gm,gn,2) = bitor(img\_prep(gm,gn,2),uint8(1));

end

gm = gm + 1;

if gm > maxM

gn = gn + 1;

gm = 1;

end

% Bit 3: Blue

if str2double(temp\_code(3)) == 0

img\_prep(bm,bn,3) = bitand(img\_prep(bm,bn,3),uint8(254));

else

img\_prep(bm,bn,3) = bitor(img\_prep(bm,bn,3),uint8(1));

end

bm = bm + 1;

if bm > maxM

bn = bn + 1;

bm = 1;

end

% Bit 4: Blue

if str2double(temp\_code(4)) == 0

img\_prep(bm,bn,3) = bitand(img\_prep(bm,bn,3),uint8(254));

else

img\_prep(bm,bn,3) = bitor(img\_prep(bm,bn,3),uint8(1));

end

bm = bm + 1;

if bm > maxM

bn = bn + 1;

bm = 1;

end

% Bit 5: Green

if str2double(temp\_code(5)) == 0

img\_prep(gm,gn,2) = bitand(img\_prep(gm,gn,2),uint8(254));

else

img\_prep(gm,gn,2) = bitor(img\_prep(gm,gn,2),uint8(1));

end

gm = gm + 1;

if gm > maxM

gn = gn + 1;

gm = 1;

end

% Bit 6: Red

if str2double(temp\_code(6)) == 0

img\_prep(rm,rn,1) = bitand(img\_prep(rm,rn,1),uint8(254));

else

img\_prep(rm,rn,1) = bitor(img\_prep(rm,rn,1),uint8(1));

end

rm = rm + 1;

if rm > maxM

rn = rn + 1;

rm = 1;

end

% Bit 7: Red

if str2double(temp\_code(7)) == 0

img\_prep(rm,rn,1) = bitand(img\_prep(rm,rn,1),uint8(254));

else

img\_prep(rm,rn,1) = bitor(img\_prep(rm,rn,1),uint8(1));

end

rm = rm + 1;

if rm > maxM

rn = rn + 1;

rm = 1;

end

% Bit 8: Green

if str2double(temp\_code(8)) == 0

img\_prep(gm,gn,2) = bitand(img\_prep(gm,gn,2),uint8(254));

else

img\_prep(gm,gn,2) = bitor(img\_prep(gm,gn,2),uint8(1));

end

gm = gm + 1;

if gm > maxM

gn = gn + 1;

gm = 1;

end

end

%% Step 6: Final Output

J = img\_prep; % Final Encoding Output

% J = msg\_enc\_set; % ENCRYPTION STEP TEST OUTPUT

end