Volume 4, Issue 9, September 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Steganography: The Art of Hiding Text in Image using Matlab

Shikha
M.tech (cse) Deptt of CSE, JCDV Sirsa & GJU, Hissar, India

Vidhu Kiran Dutt
Lect in deptt of CSE, JCDV Sirsa &GJU, Hissar, India

Abstract-Steganography is the art of hiding the fact that communication is taking place, by hiding information in other information. Many different carrier file formats can be used, but digital images are the most popular because of their frequency on the Internet. For hiding secret information in images, there exists a large variety of steganographic techniques some are more complex than others and all of them have respective strong and weak points. Different applications have different requirements of the steganography technique used. For example, some applications may require absolute invisibility of the secret information, while others require a larger secret message to be hidden. This paper intends to give an overview of image steganography, its uses and techniques. This paper intends to give an overview of image steganography, its uses and techniques. It also show how a text document to be hide in an image file.

Key Words- Steganography, Cryptography, Secret information, Cover image, Encryption, Decryption.

I. INTRODUCTION

The internet is a huge collection of networks. It is a super-highway that connects places all over the world. Internet is one of the rapidly growing technologies in the present era. This growth has focused attention on one of the most important aspect of internet viz. information security. Since internet is a public network, securing the information on internet is very important. Various techniques including cryptography, steganography etc. are used to secure information on the internet. Cryptography is the science of converting the messages that are intended to be secret into some other form, such that it is not understandable to anyone other than the intended sender and recipients. Steganography is a technique for securing information by hiding it in some other medium, such that the existence of information is concealed to everyone except for the intended sender and receiver [1].

Steganography refers to the art and science of hiding secret information in some other media. The information to be hided is called the secret message and the medium in which the information is hided is called the cover document. The cover document containing hidden message is called stego-document. The algorithms employed for hiding the message in the cover medium at the sender end and extracting the hidden message from the stego-document at the receiver end is called stego system.

A Types of Steganography: Steganography can be broadly classified into four categories, and these are [2]:

1) Text Steganography
2) Image Steganography
3) Audio Steganography
4) Video Steganography

Figure 1. Types of Steganography

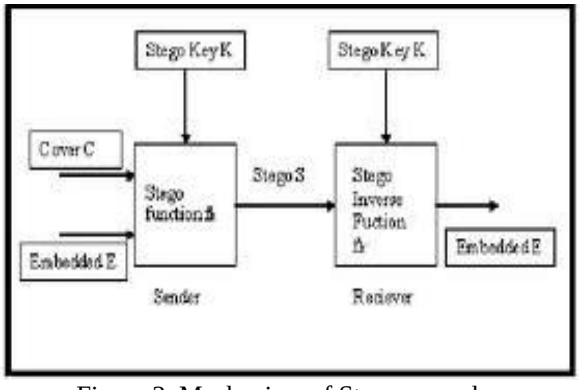

Figure 2. Mechanism of Steganography

1) *Text Steganography*: A steganography technique that uses text as the cover media is called a text steganography. It is one of the most difficult types of the steganography technique. This is because text files have a very small amount of redundant data to hide a secret message.
2) *Audio Steganography*: A steganography technique that uses audio as the cover media is called an audio steganography. It is the most challenging task in steganography. This is because the human auditory system (HAS) has a large dynamic range that it can listen over. Thus, even a minute change in audio quality also can be detected by the human ears.
3) *Video Steganography*:A steganography technique that uses video as the cover media is called image steganography.
4) *Image Steganography:* A steganography technique that uses images as the cover media is called an image steganography. Hiding secret messages in digital images is the most widely used method as it can take advantage of the limited power of the human visual system (HVS) and also because images have a large amount of redundant information that can be used to hide a secret message.

II. RELATED WORK

Hiding data is the process of embedding information into digital content without causing perceptual degradation. In data hiding, three famous techniques can be used. They are watermarking, steganography and cryptography [3]. The main advantage of steganography algorithm is because of its simple security mechanism. There are several steganography techniques used for hiding data such as batch steganography, permutation steganography, least significant bits (LSB), bit-plane complexity segmentation (BPCS) and chaos based spread spectrum image steganography (CSSIS)[4]. Research in hiding data inside image using steganography technique has been done by many researchers. Warkentin et al. proposed an approach to hide data inside the audiovisual files. In their steganography algorithm, to hide data, the secret content has to be hidden in a cover message. El-Emam, on the other hand, proposed a steganography algorithm to hide a large amount of data with high security[5]. His steganography algorithm is based on hiding a large amount of data (image, audio, text) file inside a colour bitmap (bmp) image.

In his research, the image will be filtered and segmented where bits replacement is used on the appropriate pixels. These pixels are selected randomly rather than sequentially. Chen et al. modified a method used in using the side match method [6]. They concentrated on hiding the data in the edge portions of the image. Wu et al., on the other hand, used pixel-value differencing by partitioning the original image into non-overlapping blocks of two consecutive pixels. This research uses a similar concept introduced by El-Emam . A bitmap (bmp) image will be used to hide the data. Data will be embedded inside the image using the pixels. Then the pixels of stego image can then be accessed back in order to retrieve back the hidden data inside the image[7]. Two stages are involved. The first stage is to come up with a new steganography algorithm in order to hide the data inside the image and the second stage is to come up with a decryption algorithm using data retrieving method in order to retrieve the hidden data that is hided within the stego image [8].

III. PROPOSED WORK

In this paper we are going to propose the following four parts.

1) Encrypt the text using play fair method.
2) Encode the encrpted text in to the image file using matlab code.
3) Decode the image file,and extract the encrypted text message using matlab code.
4) Decrypt the text message using play fair method.

Code for encoding and decoding the text message in matlab is as the following:

```
function []=txthide(arg)
% TXTHIDE Hides text message (MSG) within an image, using a key (KEY).
% After typing txthide at the command line, a GUI appears that will guide
% the user through the process of encoding the text. The maximum length of
% text that can be encoded is 1000 characters. This could be
% modified, but represents a good comprimise between speed and
% functionality on the author's computer.
% Text can be hidden in a bitmap (*.bmp), a tif (*.tif) or a jpg (*.jpg ).
% However, the encoded image can only be saved as a bitmap or
% tif; the jpg format uses compression and the decoder will not return the
% correct MSG if the user tries to save the encoded image in this
% format.
% Any character from a common keyboard can be encoded, including the
% shifted characters and numbers, such as !@#$%^&*()_+123 etc.
%---------USER NOTES--------%
% 1. User may hit the 'Return' key while typing the MSG, but this is NOT
% recommended, as the formatting will not be preserved on decoding,
% and the string length counter will not display the correct string
% length. Similarly for the 'Tab' key, use spaces instead.
% 2. User should NOT hit the 'Return' key as part of the decoding KEY.
if nargin==0
```



```
arg = 'initialize';
else
% Retrieve/redefine (See definitions below) Mainly for easy reading.
hands = get(gcf,'userdata');
pos = get(gcbf,'position');
checkd = hands(1); % For user to choose to decode a MSG.
checke = hands(2); % For user to choose to encode a MSG.
edit1 = hands(3); % Space for user to enter MSG.
edit2 = hands(4); % Space for user to enter key.
frame1 = hands(5);
frame2 = hands(6);
push1 = hands(7); % Indicates user is done entering MSG.
push2 = hands(8); % Indicates user is done entering KEY.
static1 = hands(9);
static2 = hands(10);
static3 = hands(11);
static4 = hands(12);
static5 = hands(13);
end
switch arg
case 'initialize' % Creation of GUI and storage of handles.
counter=0; % This is where the length of MSG is stored.
h_fig = figure('name','Text Hider','position',[460 760 300 80],...
'menubar','none','resize','off');
col = get(h_fig,'color');
checkd=uicontrol(h_fig,'style','checkbox','position',[140 20 60 20],...
'string','Decode','backgroundcolor',col,'callback',...
'txthide(''decode'')','fontweight','bold');
checke=uicontrol(h_fig,'style','checkbox','position',[140 50 60 20],...
'string','Encode','fontweight','bold',...
'backgroundcolor',col,'callback',...
'txthide(''encode'')');
edit1=uicontrol(h_fig,'style','edit','horizontalalignment','left',...
'max',9,'min',1,'visible','off','KeyPressFcn',...
'txthide(''count'')','userdata',counter);
edit2=uicontrol(h_fig,'style','edit','horizontalalignment','left',...
'max',9,'min',1,'visible','off');
frame1=uicontrol(h_fig,'style','frame','visible','off');
frame2=uicontrol(h_fig,'style','frame','visible','off');
push1=uicontrol(h_fig,'style','pushbutton','string','done',...
'callback','txthide(''done1'')','visible','off');
push2=uicontrol(h_fig,'style','pushbutton',...
'string','done','callback','txthide(''done2'')',...
'visible','off');
static1=uicontrol(h_fig,'style','text','visible','off',...'string','Enter the message you wish to hide.',...
'fontweight','bold','backgroundcolor',col);
static2=uicontrol(h_fig,'style','text','position',[30 33 75 20],...
'strin','Choose one:','fontweight','bold',...
'backgroundcolor',col);
static3=uicontrol(h_fig,'style','text','visible','off',...
'string','Enter the key for decoding.',...
'fontweight','bold','backgroundcolor',col);
static4=uicontrol(h_fig,'style','text','visible','off',...'string','Message Length:',...
'fontweight','bold','backgroundcolor',col);
static5=uicontrol(h_fig,'style','text','visible','off',...'string','0','foregroundcolor','blue',...
'fontweight','bold','backgroundcolor',col);
hands=[checkd checke edit1 edit2 frame1 frame2 push1 push2 static1...
static2 static3 static4 static5];
set(h_fig,'userdata',hands); % Store handles in figures userdata.
case 'encode' % The user has chosen to encode a MSG.
set(gcbf,'position',[pos(1) pos(2)-300 300 380]);
set(checkd,'enable','off','position',[140 320 60 20]);
set(checke,'position',[140 350 60 20],'enable','off');
```



```
set(edit1,'visible','on','position',[30 50 245 200],'fontsize',...
10,'string',[' Maximum Number of Characters ',...
' is 1000. See Counter Below'],'foregroundcolor',...
'red'); % Issue a reminder about the max MSG length.
set(frame1,'visible','on','position',[30 295 245 3],...
'backgroundcolor','black');
set(push1,'visible','on','position',[230 10 45 28]);
set(static1,'visible','on','position',[30 260 245 20]);
set(static2,'position',[30 333 75 20]);
set(static4,'visible','on','position',[28 4 100 28]);
set(static5,'visible','on','position',[130 4 50 28]);
pause(1.75); % Displays reminder only a short time.
set(edit1,'foregroundcolor','black','string','','fontsize',9)
case 'decode' % The user has decided to decode a MSG.
set(gcbf,'position',[pos(1) pos(2)-110 300 190]);
set(checkd,'enable','off','position',[140 130 60 20]);
set(checke,'enable','off','position',[140 160 60 20]);
set(edit2,'visible','on','position',[30 40 245 30]);
set(frame1,'position',[30 110 245 3],'visible','on',...
'backgroundcolor','black');
set(push2,'visible','on','position',[130 7 45 28]);
set(static2,'position',[30 143 75 20]);
set(static3,'visible','on','position',[30 75 245 20]);
case 'done1' % The user is done entering the MSG to encode.
set(gcbf,'position',[pos(1) pos(2)-140 300 520]);
set(checkd,'position',[140 460 60 20]);
set(checke,'position',[140 490 60 20]);
set(edit1,'position',[30 190 245 200],'enable','off');
set(edit2,'visible','on','position',[30 60 245 30]);
set(frame1,'position',[30 435 245 3]);
set(frame2,'position',[30 133 245 3],'visible','on',...
'backgroundcolor','black');
set(push1,'position',[230 150 45 28],'enable','off');
set(push2,'visible','on','position',[130 20 45 28]);
set(static1,'position',[30 400 245 20]);
set(static2,'position',[30 473 75 20]);
set(static3,'visible','on','position',[30 95 245 20]);
set(static4,'position',[28 144 100 28]);
set(static5,'position',[130 144 50 28])
if isempty(get(edit1,'string')) % User tried to encode nothing!
fprintf('\n\t\t No message entered, try again.\n\n')
close(gcbf)
return
end
case 'done2' % User has entered the key for use in encoding/decoding.
en_or_de = get(checke,'value'); % Is user encoding or decoding?
if en_or_de==1 % Encoding.
key = get(edit2,'string');
if isempty(key) % Check if user tried to encode with no KEY.
fprintf('\n\t\t No key entered, try again.\n\n');
close(gcbf);
return
end
msg = get(edit1,'string');
[i j] = size(msg); % Useful if user hit 'return' in MSG.
if i >1
msg = reshape(msg',1,i*j); % MSG should be 1 by n.
end
close(gcbf);
wrkd = encode(msg,key); % Pass to encode function.
if isempty(wrkd) % User hit cancel when choosing a file.
close(gcbf);
fprintf('\n\t\t Operation aborted.\n\n');
```



```
return
end
fprintf('\t\t\n Your message was encoded.\n\n');
else % Decoding.
key = get(edit2,'string');
if isempty(key) % Check if user tried to dencode with no KEY.
close(gcbf)
fprintf('\n\t\t No key entered, try again.\n\n')
return
end
msg=decode(key); % Pass to decode function.
if isempty(msg) % User hit cancel when choosing a file.
close(gcbf)
fprintf('\n \t\t Operation aborted.\n\n')
return
end
close(gcbf); % New figure is created next to display MSG.
h_fig = figure('name','The Message','position',...
[460 460 300 380],'menubar','none');
col=get(h_fig,'color');
stat = uicontrol(h_fig,'style','edit','position',...
[30 40 245 300],'string',msg,'fontsize',9,...
'horizontalalignment','left','max',9,'min',1);%#ok
stat2 = uicontrol(h_fig,'style','text','position',...
[30 355 245 20],'string','Your Message:',...
'fontweight','bold','backgroundcolor',col,...
'horizontalalignment','left','fontsize',12);%#ok
end
case 'count' % This is where the length of MSG is counted.
ch=get(gcbf,'currentcharacter');
if isempty(ch) || ch==13
return
elseif ch==8
counter = get(edit1,'userdata')-1;
else
counter = get(edit1,'userdata')+1; % Retrieve current length.
end
set(edit1,'userdata',counter);
set(static5,'string',num2str(counter));
if counter > 990 % Issue a warning when getting close to limit.
set(static5,'foregroundcolor','red');
end
end % End switch
%----------New Function Starts Below----------
function success = encode(msg,key)
% ENCODE(msg,key) Encodes a text message using a key.
% This works by using the ascii number representation of the chars in
% KEY (and certain permutations) to create coordinate pairs that represent
% the places in matrix A (which is a submatrix of the pic matrix) where the
% unit alterations to the pic matrix will be made. I tried to make the
% code easy to follow, but I will respond to questions about how this
% works.
num2add = 1000-length(msg); % Number of spaces to add to end of MSG.
if num2add < 0, error('This message is too long to encode.'), end
newmsg = [msg,repmat(' ',1,num2add)]; % 1000 chars always encoded.
msgmat = dec2bin(newmsg)-48; % Each row is a bin. rep. of an ascii char.
[filen pth]=uigetfile({'*.bmp';'*.tif';'*.jpg'},'Choose Image To Encode.');
if isequal(filen,0) || isequal(pth,0)
success = []; return % User cancelled.
end
pic1 = imread([pth filen]);
B = pic1(:,:,1); [piclngth pichght] = size(B); % Choose the first page.
dim1 = piclngth-2; dim2 = pichght-3; keyb = key(end:-1:1);
```



```
rows = cumsum(double(key));
columns = cumsum(double(keyb)); % Coord pairs for KEY (rows,columns)
A = zeros(dim1,dim2); % This matrix will house the hiding points.
A = crtmtrx(A,rows,columns,dim1,dim2,key);
idx = find(A==1); % This same index will be used for pic matrix.
for vv = 1:1000 % This is the encoder.
for uu = 1:7
if msgmat(vv,uu)==1;
if rem(B(idx(uu+7*(vv-1))),2)==0
B(idx(uu+7*(vv-1))) = B(idx(uu+7*(vv-1)))+1;
end
elseif rem(B(idx(uu+7*(vv-1))),2)==1
B(idx(uu+7*(vv-1))) = B(idx(uu+7*(vv-1)))-1;
end
end
end
newpic = pic1; newpic(:,:,1) = B;
[filen pth] = uiputfile({'*.bmp';'*.tif'},'Save Encoded File As');
if isequal(filen,0) || isequal(pth,0)
success = []; return
end % User cancelled.
imwrite(newpic,[pth filen])
success = 1;
%----------New Function Starts Below----------
function msg = decode(key)
% DECODE(key) Decodes the message hidden by encode in pic, using key.
[filen pth] = uigetfile({'*.bmp';'*.tif'},'Choose Image To Decode.');
if isequal(filen,0) || isequal(pth,0)
msg = []; return
end % User cancelled.
pic2 = imread([pth filen]);
B = pic2(:,:,1); [piclngth pichght] = size(B); % Choose the top page.
dim1 = piclngth-2; dim2 = pichght-3; keyb = key(end:-1:1);
rows = cumsum(double(key)); columns = cumsum(double(keyb));
A = zeros(dim1,dim2); % This matrix houses the hiding points.
A = crtmtrx(A,rows,columns,dim1,dim2,key);
idx = find(A==1); msgmat = zeros(1000,7);
for vv = 1:1000 % This is the decoder.
for uu = 1:7
if rem(B(idx(uu+7*(vv-1))),2)==1
msgmat(vv,uu) = 1;
end
end
end
msg = char(bin2dec(num2str(msgmat)))';
%----------New Function Starts Below----------
function A = crtmtrx(A,rows,columns,dim1,dim2,key)
% Creates the matrix used to find the points to hide the message.
jj = 1; idx = 1;
while 7000 > length(idx) % Need 7000 points to hide 1000 characters.
for ii = 1:length(rows)
if rows(ii) < dim1
rows(ii) = rem(dim1,rows(ii))+1;
else
rows(ii) = rem(rows(ii),dim1)+1;
end
if columns(ii) < dim2
columns(ii) = rem(dim2,columns(ii))+1;
else
columns(ii) = rem(columns(ii),dim2)+1;
end
A(rows(ii),columns(ii)) = 1;
end
```



```
rows = jj*cumsum(double(columns))+round(dim2/2); % Each pass is diff.
columns = jj*cumsum(double(rows))+round(dim1/2);
if jj > ceil(7000/length(key))+2 % Estimate how many iters. needed.
idx = find(A==1);
end
jj = jj+1;
end
```

IV. CONCLUSION

Steganography is powerful and effective for communication of secret data. For the image steganography various methods have been proposed[4]. In this paper, we propose a method that hide the secret messages in the image using matlab. Matlab in not only a programming language, but a programming environment as well. It is provide more security for secret communications. Thus the capacity of the hiding process to hide secret messages is also high in the proposed technique.

REFERENCES

[1] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "*Techniques for data hiding*", IBM Systems , vol. 35, Issues 3&4 1996 Journal, pp. 313-336.

[2] T. Moerland, "*Steganography and Steganalysis*", www.liacs.nl/home/tmoerlan/privtech.pdf, May 15, 2003.

[3] Jamil, T., "*Steganography: The art of hiding information is plain sight*", 1999 IEEE Potentials, 18:01.

[4] Moerland, T., "*Steganography and Steganalysis*", www.liacs.nl/home/ tmoerl/privtech.pdf. Leiden Institute of Advanced Computing Science.

[5] Nasir Memon R. Chandramouli. *Analysis of lsb based image steganography* techniques. In , 2001 Proceedings of IEEE ICIP.

[6] R. B¨ohme and A. Westfeld. *Breaking cauchy model-based JPEG steganography with first order statistics*. , 2004 9th European Symposium on Research Computer Security 3193:125–140.

[7] Silman, J., "*Steganography and Steganalysis: An Overview*", 2001 SANS Institute.

[8] Souvik Bhattacharyya. and Gautam Sanyal. *An image based steganography model for promoting global cyber security*. In, 2009 Proceedings of International Conference on Systemics, Cybernetics and Informatics, Hyderabad, India.

