DIGITAL IMAGE WATERMARKING

PROJECT REPORT

Malik Aneeb Ul Islam (16D070035)

Ajit Zote (16D070002)

24.11.2019

EE610-DIGITAL IMAGE PROCESSING

INTRODUCTION

A few years back Unique Identification Authority of India (UIDAI) experienced a breach in user data as the information about millions of registered personnel was leaked to a third party for processing and unauthorised use. The main issue that was found at that time was the level of security that was used to safeguard the details of the registered users was not apt. This lack in the level of security was then rectified by UIDAI. Keeping the above issue in mind, what if there is some tampering with the user data, that UIDAI has, like fingerprints, retina scans or basic personal information? The current project deals with the safeguarding of these particular details with the help of the method of Digital Watermarking. We will hide the image of fingerprint inside the photo of Aadhar card. In short, we will use fingerprint as a watermark. So if something anomaly happens inside the database of UIDAI, the system administrators can check whether the data matches with the watermark or not, and based on that take decisions.

ABSTRACT

Digital watermarking is a method used to embed information in the form of bit-stream into a file. The information to be embedded is called the 'message' and the host file in which this bit-stream going to be embedded is called 'asset'. The asset and the message files can be an image, audio or video, or even text files. In our case, both the target files are images. Hence, all our work is focused on the domain of image on image watermarking. Our target is to watermark the individual's fingerprint on his/her 'Aadhar Card' for owner identification and proof of ownership.

Digital Watermarked images are immune to attacks like compression (property of robustness). Other notable specifications are capacity and imperceptibility.

PROCEDURES

1. Bit-plane manipulation in the spatial domain
2. Transform superpositions in the frequency domain

BIT-PLANE MANIPULATION IN SPATIAL DOMAIN

As we know an image is consists of pixels. Assuming for simplicity that each pixel has 8-bit depth, then the matrix of only 8th bits of each pixel is called the 8th bit-plane. So, in an 8-bit deep image, there are 8 bit-planes (starting from first bit-plane to 8th bit-plane). The 8th bit-plane is also known as the most significant bit-plane as it contains the most important details of the image and the first bit-plane is the least significant bit-plane as its importance in image details is not profound. So, in case of image watermarking in the spatial domain, we replace the least significant bit-planes of the asset by the most significant bit-planes of the message (that have the important message details). At the time of extraction and reconstruction of the message, we only need to work with the most significant bit-planes.

Both the message and host files, in our case, therefore, in the beginning, are bought to an 8-bit deep Grayscale level. The bit-planes of both of the images are extracted (Figures 2 and 4). The first few most significant bits contain the most information of the images and the trick is to utilize the first few most significant bit-planes of both the host and message images (Figures 3 and 5). In our case, we've removed the bottom three bit-planes of the host and replaced the first three most significant bit-planes of the message in their place (Figures 3 and 5).

To separate the watermark from the image, we begin by generating the bit-planes of the given image (Figures 6 and 7). We then extract the least three significant bits of the image and using logical left shifts, we bring those three least significant bit-planes to the most significant positions. Then by superposition, we get the fingerprint separated from the host (Figure 8). In a similar way, we superimpose the five bit-planes of the host to recover the asset (Figure 9).

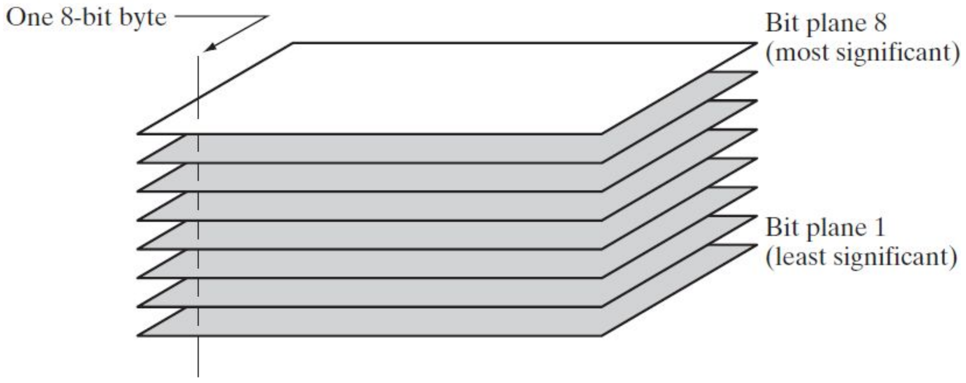
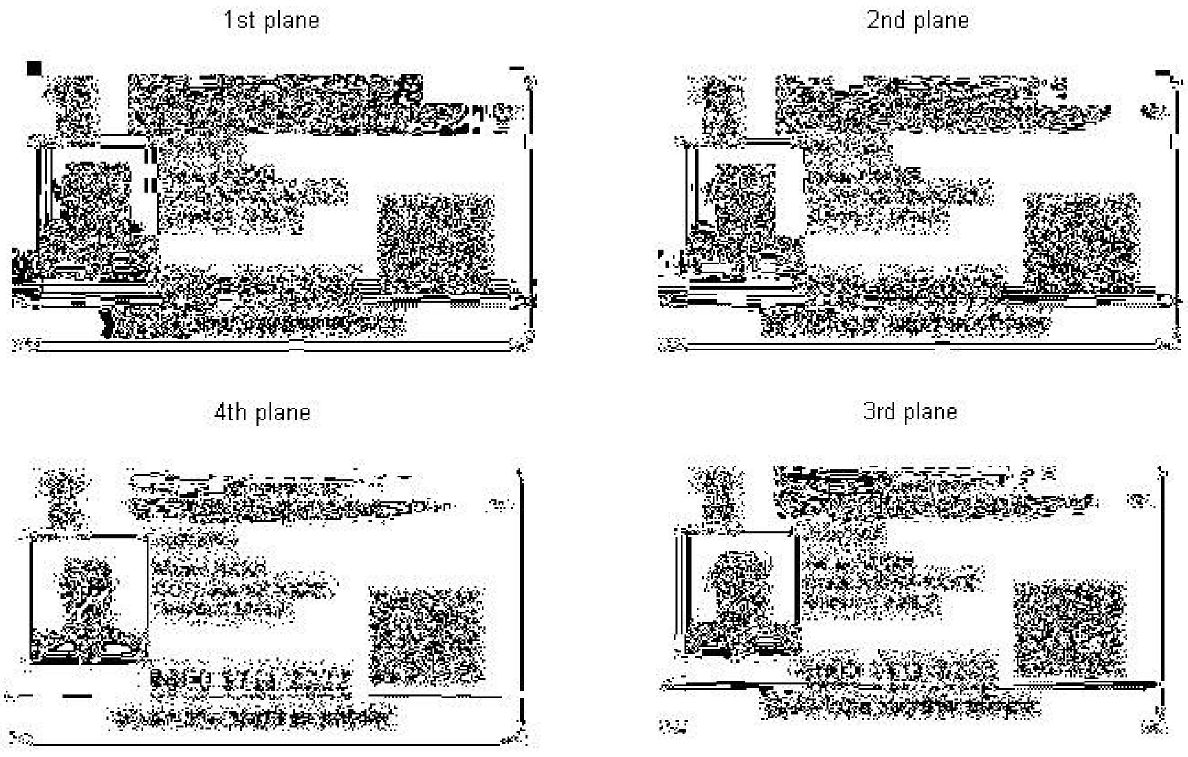


Figure 1: Bit-planes of an 8-bit deep image [1]

RESULTS FOR BIT-PLANE MANIPULATION



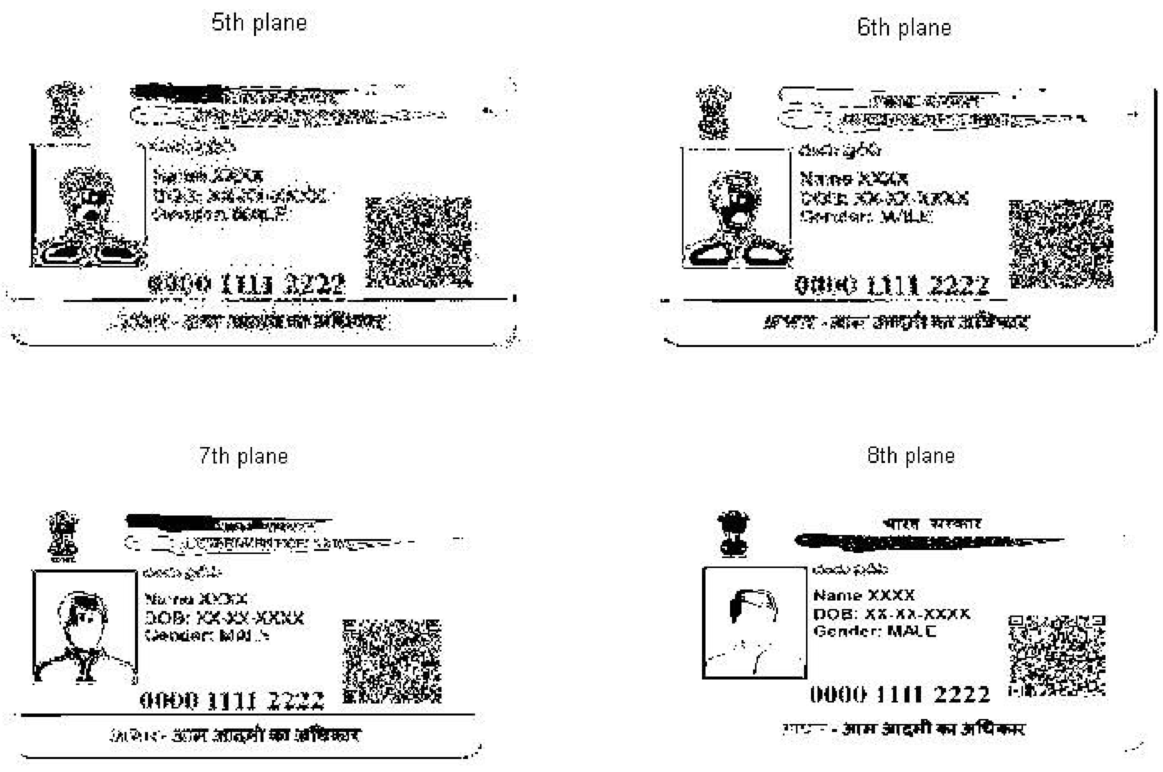


Figure 2: Extracted bit-planes of the host image.

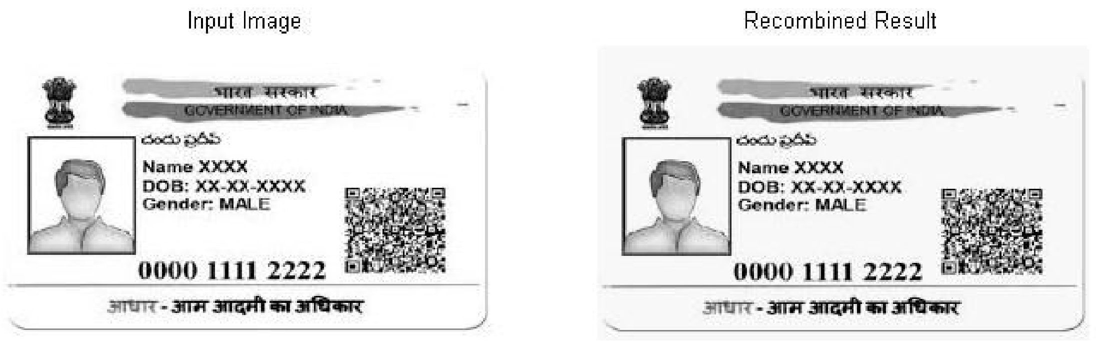


Figure 3: Comparison between the host and the compressed host generated after removal of the three least significant bit-planes.

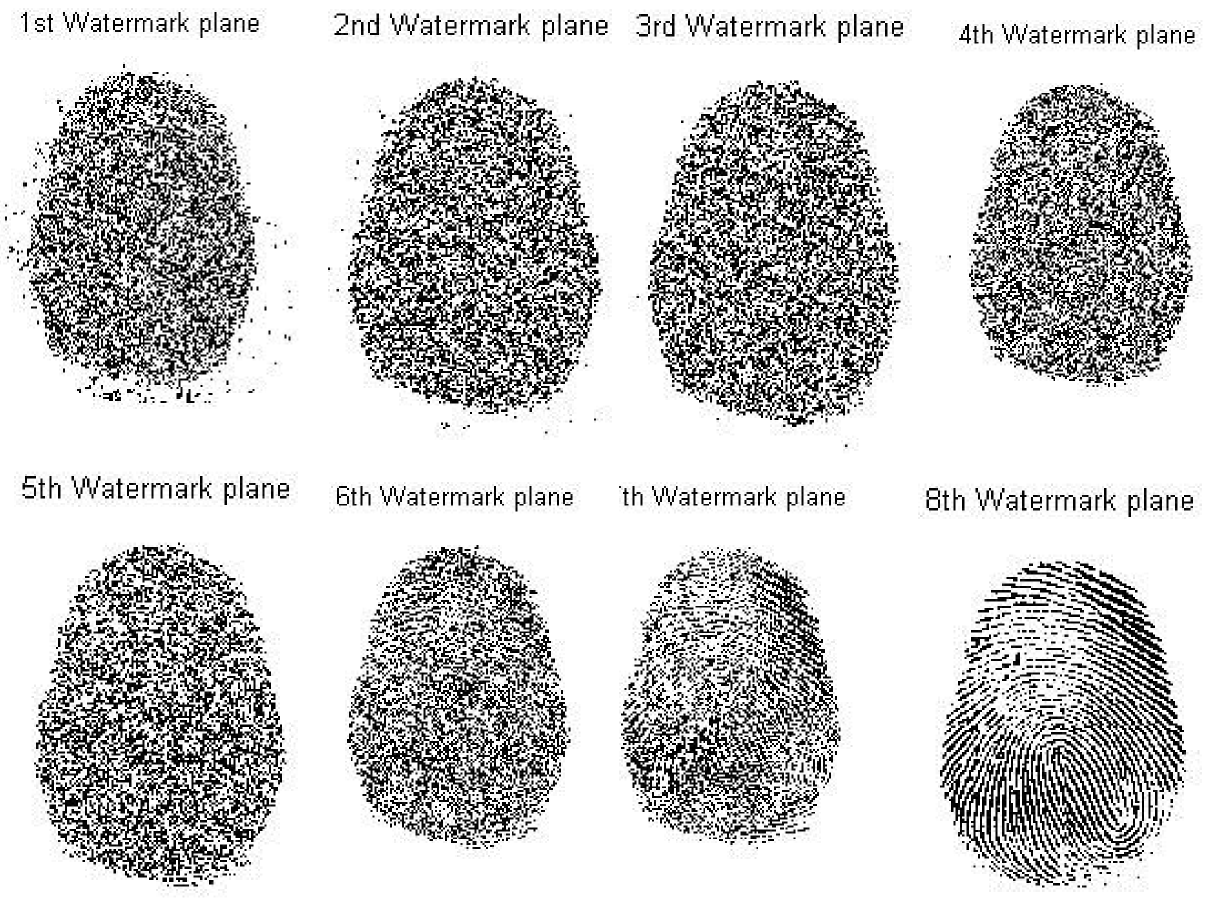


Figure 4: Extracted bit-planes of the watermark.

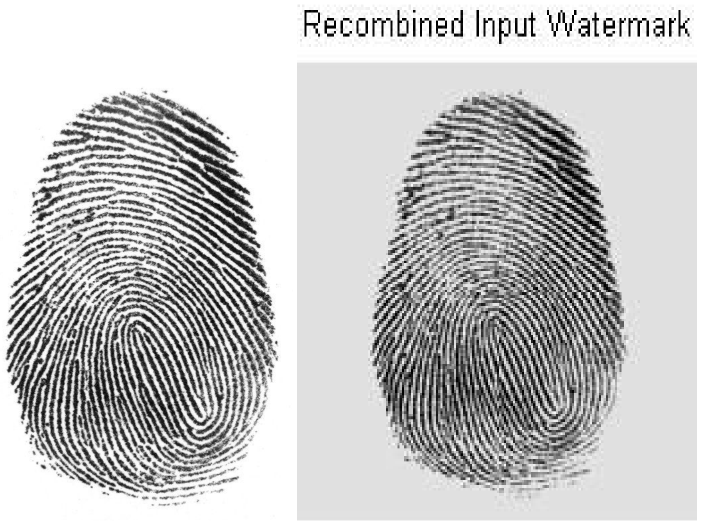


Figure 5: Comparison between the original watermark and the compressed watermark generated after removal of the five least significant bit-planes.

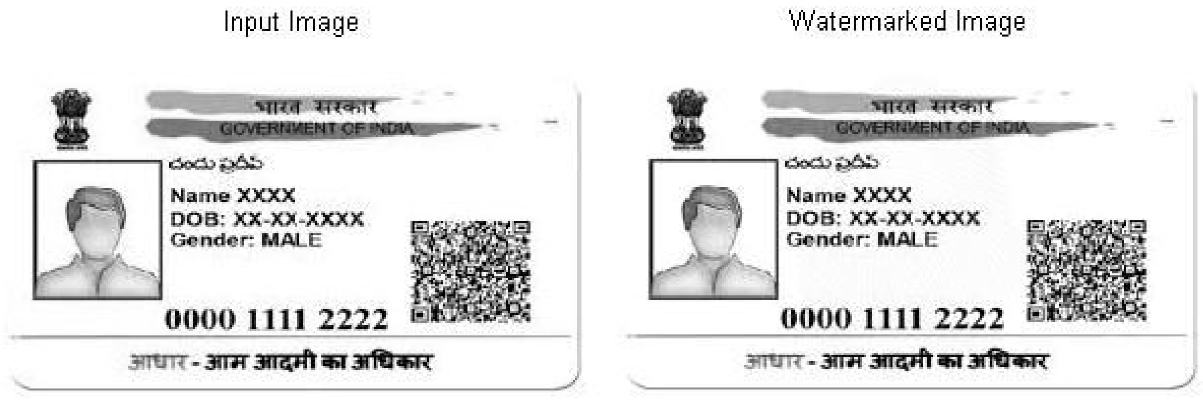


Figure 5: Comparison between the original host and the watermarked image after bit-plane manipulation.

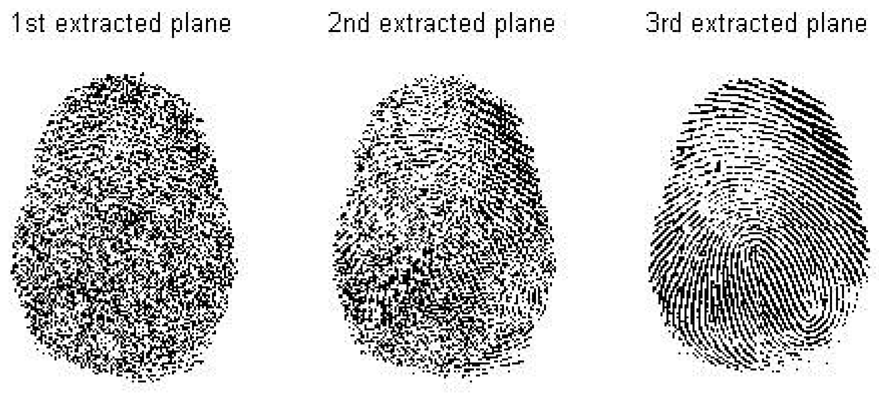
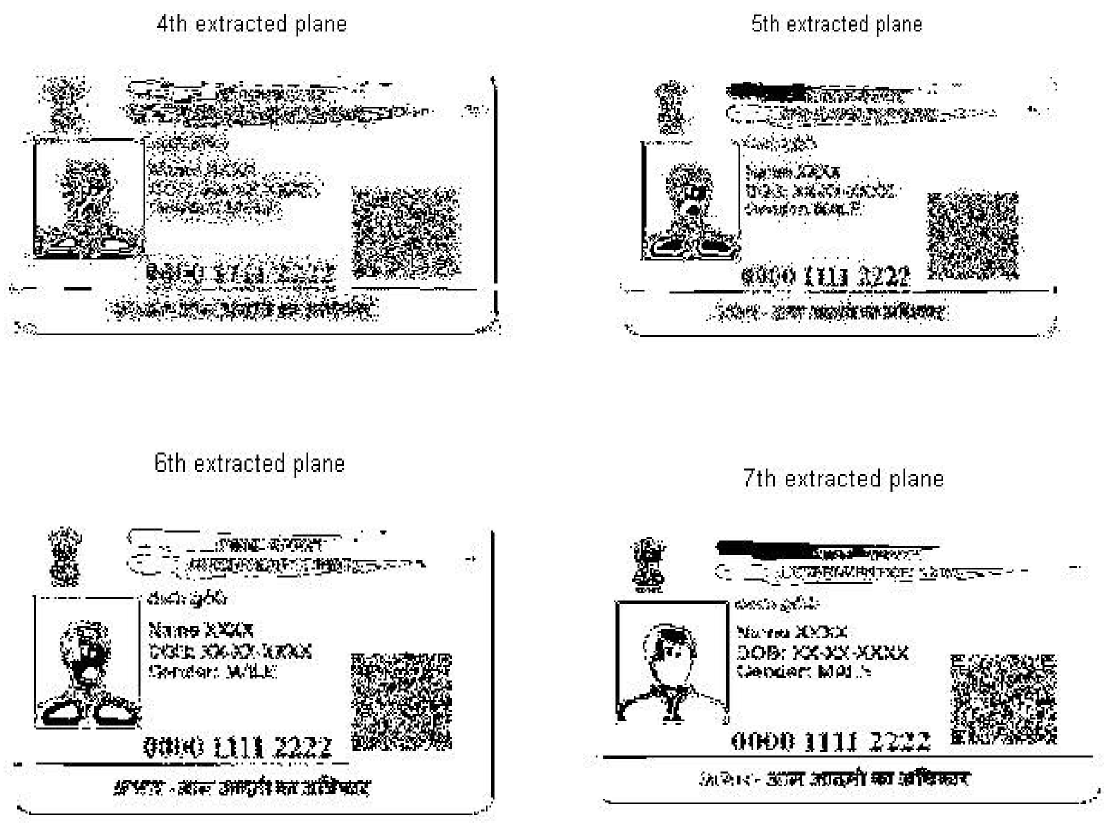
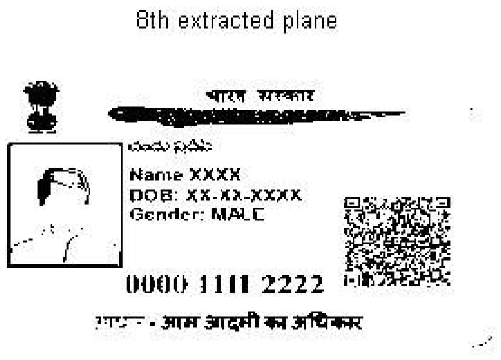


Figure 6: Extracted fingerprint bit-planes from the watermarked image in Figure 5.




8th extracted plane

भारत सरकार

[redacted]

Name XXXX

DOB: XX-XX-XXXX

Gender: MALE

0000 1111 2222

- आम आदमी का अधिकार

Figure 7: Extracted Aadhar bit-planes from the watermarked image in Figure 5.

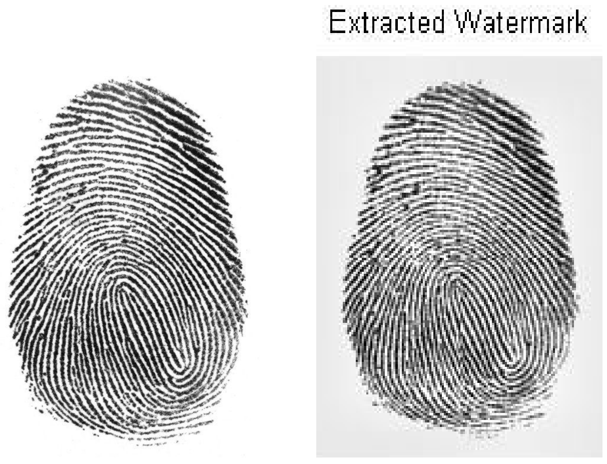


Figure 8: Comparison between extracted fingerprint from the watermarked image in Figure 5 and the original fingerprint.

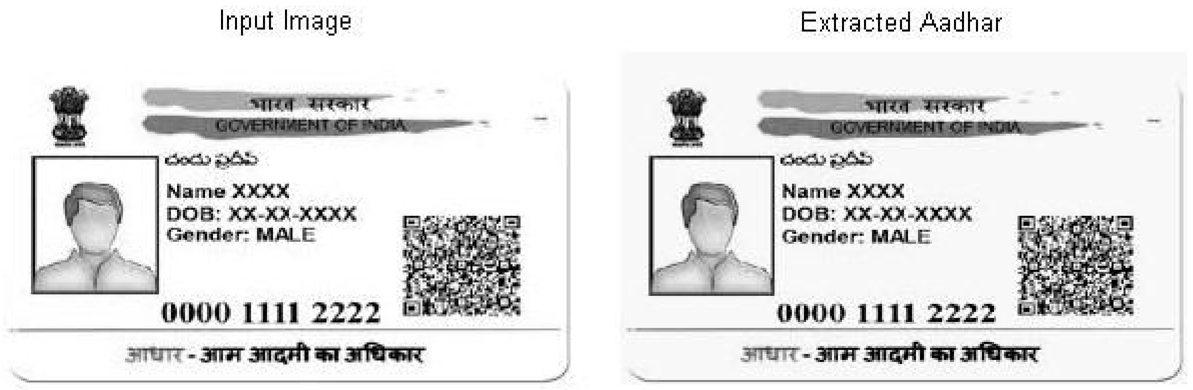


Figure 9: Comparison between extracted Aadhar from the watermarked image in Figure 5 and the original host Aadhar image.

CODE SNIPPET (SPATIAL DOMAIN)

Bit-plane Slicing Function

```
%% Function of Bit-plane Slicing

function [plane_1 plane_2 plane_3 plane_4 plane_5 plane_6 plane_7 plane_8]

                                                  = bitplane_matrix(input_image)

[row col]=size(input_image);

b=zeros(row,col,8);

for k=1:8

      for i=1:row

      for j=1:col

            b(i,j,k)=bitget(input_image(i,j),k);

      end

      end

end

plane_1=b(:,:,1);

plane_2=b(:,:,2);

plane_3=b(:,:,3);

plane_4=b(:,:,4);

plane_5=b(:,:,5);

plane_6=b(:,:,6);

plane_7=b(:,:,7);

plane_8=b(:,:,8);

end
```

Image Watermarking

```
%% INPUT IMAGE

input_image = imread('aadhar.jpg');

[r_i, c_i p_i]=size(input_image);

if (p_i==3)

      input_image = rgb2gray(input_image);

else

      input_image = input_image


end


% Extracting and Displaying Bit-planes

[plane_1,plane_2,plane_3,plane_4,plane_5,plane_6,plane_7,plane_8]

                                    = bitplane_matrix(input_image);


      figure, imshow(plane_1);

      title('1st plane');

      figure, imshow(plane_2);

      title('2nd plane');

      figure,imshow(plane_3);

      title('3rd plane');

      figure, imshow(plane_4);

      title('4th plane');

      figure, imshow(plane_5);

      title('5th plane');

      figure, imshow(plane_6);
      title('6th plane');
      figure, imshow(plane_7);
      title('7th plane');
      figure,imshow(plane_8);
      title('8th plane');

   % Displaying result
      resultant = (plane_4*8 + plane_5*16 + plane_6*32 + plane_7*64
                   + plane_8*128)/256;
      figure,imshow(resultant);
      title('Recombined Result');
      figure,imshow(input_image);
      title('Input Image');

%% WATERMARK IMAGE
input_watermark = imread('fingerprint.jpg');

 % Scaling for proper superposition

 [image_rows, image_columns] = size(input_image);

 scaled_watermark = imresize(input_watermark,[image_rows image_columns]);
 figure, imshow(scaled_watermark);
 title('Scaled Watermark');

 % Extracting and Displaying Bit-planes
[plane_1w,plane_2w,plane_3w,plane_4w,plane_5w,plane_6w,plane_7w,plane_8w]
                                     = bitplane_matrix(scaled_watermark);
```

```
    figure, imshow(plane_1w);

    title('1st Watermark plane');

    figure, imshow(plane_2w);

    title('2nd Watermark plane');

    figure,imshow(plane_3w);

    title('3rd Watermark plane');

    figure, imshow(plane_4w);

    title('4th Watermark plane');

    figure, imshow(plane_5w);

    title('5th Watermark plane');

    figure, imshow(plane_6w);

    title('6th Watermark plane');

    figure, imshow(plane_7w);

    title('7th Watermark plane');

    figure,imshow(plane_8w);

    title('8th Watermark plane');


% Displaying result

    resultant_w = (plane_6w*32 + plane_7w*64 + plane_8w*128)/256;

    figure,imshow(resultant_w);

    title('Recombined Input Watermark');

    figure,imshow('fingerprint.jpg');

    title('Input Watermark');
```

```
%% WATERMARKING
   % Displaying result
      watermarked_image = (plane_6w*1 + plane_7w*2 + plane_8w*4 + plane_4*8
                        + plane_5*16 + plane_6*32 + plane_7*64 + plane_8*128);

watermarked_image_for_display = (plane_6w*1 + plane_7w*2 + plane_8w*4
      + plane_4*8 + plane_5*16 + plane_6*32 + plane_7*64 + plane_8*128)/256;
    figure,imshow(watermarked_image_for_display);
      title('Watermarked Image');
```

Image-Watermark Separation

```
%% WATERMARK AND IMAGE SEPARATION AND BIT-PLANE DISPLAY
[plane_1e,plane_2e,plane_3e,plane_4e,plane_5e,plane_6e,plane_7e,plane_8e]
                                    = bitplane_matrix(watermarked_image);

      figure, imshow(plane_1e);
      title('1st extracted plane');
      figure, imshow(plane_2e);
      title('2nd extracted plane');
      figure,imshow(plane_3e);
      title('3rd extracted plane');
      figure, imshow(plane_4e);
      title('4th extracted plane');
      figure, imshow(plane_5e);
      title('5th extracted plane');
      figure, imshow(plane_6e);
      title('6th extracted plane');
      figure, imshow(plane_7e);
```

```
     title('7th extracted plane');

     figure,imshow(plane_8e);

     title('8th extracted plane');


extracted_watermark_1 = (plane_1e*32 + plane_2e*64 + plane_3e*128)/256;

 extracted_watermark = adapthisteq(extracted_watermark_1,'NumTiles',

                       [8 6],'Cliplimit',0.125);

                 % This toolbox function performs so-called contrast-
           % limited

                 % adaptive histogram equalization (CLAHE). 'NumTiles'

                 % divides the given image into 8x6 tile(section) image.

                 % 'Cliplimit' keeps the contrast limited from 0 to 1.



figure, imshow(extracted_watermark);

title('Extracted Watermark');


extracted_aadhar = (plane_4e*8 + plane_5e*16 + plane_6e*32 + plane_7e*64 +

                     plane_8e*128)/256;

figure, imshow(extracted_aadhar);

title('Extracted Aadhar');
```

Mean Square Error function:

```
%% MEAN SQUARE ERROR FUNCTION


function err = MSE (pic1, pic2)

e=0;

[m,n]=size(pic1);

for i=1:m

for j=1:n

e = e + double((pic1(i,j)-pic2(i,j))^2);

end

end

err = e / (m*n);

end

%% ERROR ANALYSIS

 outside_watermark = imread('fp3.jpg');

 [ew_rows, ew_columns] = size(extracted_watermark);

  external_watermark = imresize(outside_watermark,[ew_rows ew_columns]);

error = MSE (extracted_watermark, external_watermark)
```

Mean square error analysis:

$$MSE=\frac{1}{n}\sum_{i=1}^{n}\left(X_i-X_i^{*}\right)^2$$

In the above last piece of code, we’ve done error analysis between extracted watermark (say of ‘fp1.jpg’) and an external fingerprint (say ‘fp2.jpg’).

The error results:

1) The error between the extracted watermark of ‘fp1.jpg’ and external fingerprint ‘fp1.jpg’ = ZERO.

2) The error between the extracted watermark of ‘fp1.jpg’ and external fingerprint ‘fp2.jpg’ = 1.9797e-05.

3) The error between the extracted watermark of ‘fp1.jpg’ and external fingerprint ‘fp3.jpg’ = 120.00e-05.

TRANSFORM SUPERPOSITIONS IN FREQUENCY DOMAIN

There exist many transforms that bring an image into the frequency domain. Among them, we are going to use Discrete Cosines Transform (DCT). Using this approach we are going to modify the coefficients in the frequency domain. This will make some minute unnoticeable changes to the whole image.

One of the most popular approaches in this category is the one proposed by Cox et al., 1997. In this method, the Discrete Cosines Transform (DCT) is applied to the asset image as shown in Figure 10.

For our convenience, MATLAB provides a preexisting command for obtaining DCT coefficients of images:

B = dct2(A);

Note that the output is a matrix of the same size. As illustrated in Figure 10, the absolute values of the coefficients corresponding to the low frequencies are higher and appear in the up-left corner of the square, while high-frequency coefficients appear in down-right with lower absolute values.

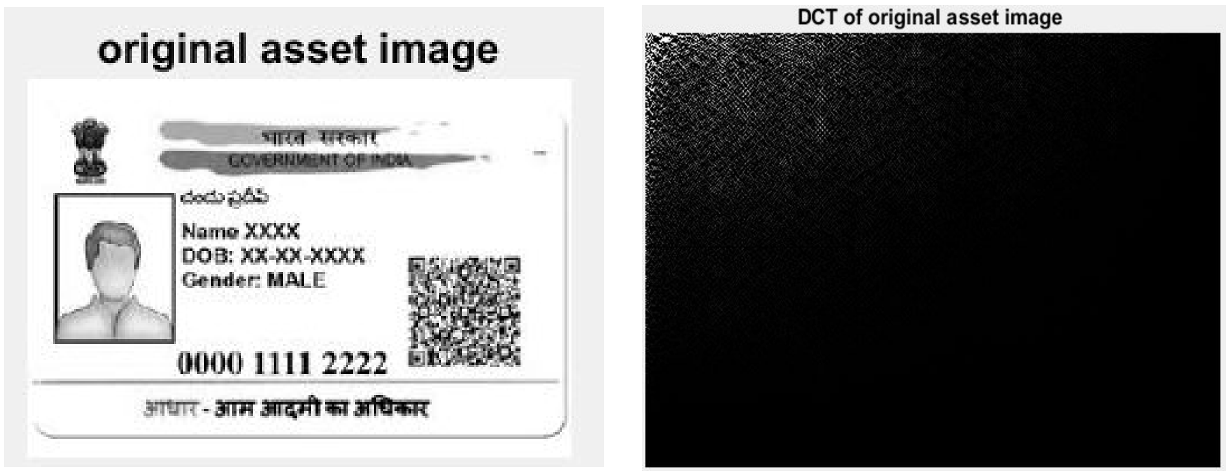


Figure 10: Asset image and its DCT.

The largest coefficient value is in the [0,0] part of the DCT coefficient matrix. The message is also coded into a spread spectrum sequence. Now how the message is added to the asset? Fig. 8 describes the process.

Cox et. al. use R1*C1 largest coefficients to embed a watermark sequence of length R1*C1 where R1 and C1 are rows and columns of watermark image respectively. The DC term, located in (0,0) of the DCT matrix, is not changed due to its perceptible change in the brightness of the picture. the author suggests not to change some coefficients near to DC term due to their noticeable change. So we are going to leave the (0,0) term of the DCT matrix and embed the remaining R1*C1 terms in decreasing order of their magnitude of coefficients.

Coefficients are modified according to the stream bits of the message using to equation 1 (Cox et al., 1997):

$$\mathbf{C_{AW} = C_A \cdot (1 + \alpha . W_i)}$$

where C_{AW} is the watermarked coefficient, C_A is the original one, α represents watermarking strength (e.g. 0.1), and W_i is the corresponding bit of the message data.

We can write the code for the method so far as follows (α =0.1):

CODE SNIPPET (FREQUENCY DOMAIN)

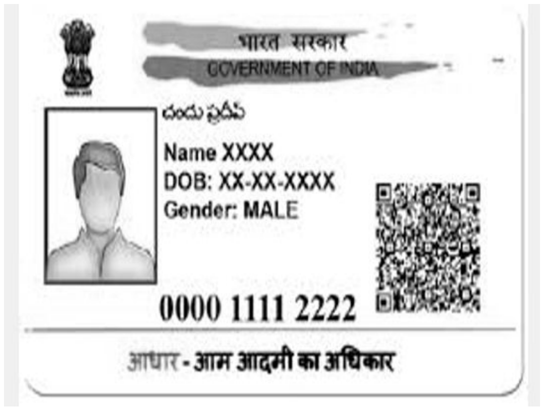
```
I = (imread('D:\imagepro\samp.jpeg'));

I=rgb2gray(I);

I=double(I);

I=I/255;

originalimage=I;

I=imresize(I,[716/2,1028/2]); % we are doing this because our image
size should be bigger than the watermart image by atleast 1 pixel

%figure,imshow(I);




W = (imread('D:\imagepro\thumb.jpeg'));

W=rgb2gray(W);

W=double(W);

W=W/255;

watermark=W;

W=imresize(W,[716/2,1026/2]); % we are doing this because our image
size should be bigger than the watermart image by atleast 1 pixel

figure()%,imshow(W);

 subplot(2,2,1),imshow(I);title('original asset image');

 subplot(2,2,2),imshow(W);title('original watermark image');


% figure,imshow(T);

% I=[1,3,5;2,4,6;7,8,9;10,12,11];

[r,c]=size(I); %size of image

[r1,c1]=size(W); %size of watermark

wmsz=r1*c1; %watermark size in a vector
```

```
D=dct2(I);%get DCT of the Asset image

figure,imshow(D) %showing DCT of the image

title('DCT of original asset image ');

D_vec=reshape(I,1,r*c); %putting all DCT values in a vector

[D_vec_srt,Idx]=sort(abs(D_vec),'descend');%re-ordering all the
absolute values

%where D_vec_srt shows the decreasing order of the D_vec and Idx shows
the

% position number the corresponding D_vec_srt was in D_vec




W_vec=reshape(W,1,r1*c1);%putting all watermark values in a vector

Idx2=Idx(2:wmsz+1);%choosing r1*c1 biggest values other than the DC
value

%we are excluding the DC value 1 and starting from 2 because of that

%finding associated row-column of the DCT image order for vector
values

IND=zeros(wmsz,2);% This is a matrix which will store the
corresponding destination of the vector which was in decreasing order

for k=1:1:wmsz

 m=mod(Idx2(k),r);

 if (m==0)

     x=floor(Idx2(k)/r);%associated culomn in the image

 else

     x=floor(Idx2(k)/r)+1;%associated culomn in the image

 end
```

```
modrow=mod(Idx2(k),r);

 if(modrow==0)

     modrow=r;

 end


 y=modrow;%associated row in the image

 IND(k,1)=y;

 IND(k,2)=x;

end

D_w=D;

for k=1:wmsz

 %insert the WM signal into the DCT values

D_w(IND(k,1),IND(k,2))=D_w(IND(k,1),IND(k,2))+.1*D_w(IND(k,1),IND(k,2)
).*W(k);

end


I2=idct2(D_w);%inverse DCT to produce the watermarked asset


figure()

subplot(2,2,1),imshow(I2);title('image after adding waterwark ');

subplot(2,2,2),imshow(I);title('image original image ');


%%now the procedure to extract the watermark and getting the image
after

%%removing watermark

W2=[];%will contain watermark signal extracted from the image
```

```
for k=1:1:wmsz

W2(k)=(((D_w(IND(k,1),IND(k,2))-D(IND(k,1),IND(k,2)))*10)/D(IND(k,1),I
ND(k,2)));%watermark extraction

end


yo = vec2mat(W2,r1); %converting the watermak wector W2 in a matrix

watermark=transpose(yo); %and transposing it to get the watermark



figure()

subplot(2,2,1),imshow(watermark);title('the retrieved watermark')%this
is the retrieved watermark

subplot(2,2,2),imshow(W);title('original watermark')


%now checking the image which was retrieved after removing waterwark

Watermarkvec=reshape(watermark,1,r1*c1);%putting all watermark images
spacial values in a vector


%taking the dct of the I2 which is the watermarked image

DW1=dct2(I2)

for k=1:wmsz

 %remove the WM signal from the DCT values

DW1(IND(k,1),IND(k,2))=DW1(IND(k,1),IND(k,2))-.1*DW1(IND(k,1),IND(k,2)
).*Watermarkvec(k);

end
```

```
image_after_removing_watermark=idct2(DW1);

figure()

subplot(2,2,1),imshow(image_after_removing_watermark);title('image_aft
er_removing_watermark');

subplot(2,2,2),imshow(I);title('original image');



difference_betwwen_original_image_and_retrieved_image_after_removing_w
atermark=I-image_after_removing_watermark;

figure,(imshow(difference_betwwen_original_image_and_retrieved_image_a
fter_removing_watermark));

title('black screen represents that there is no difference between the
original asset and image after removing watermark ')



original_watermark_minus_retrieved_watermark=W-watermark;

figure,(imshow(original_watermark_minus_retrieved_watermark));

title('black screen represents that there is no difference between the
original watermark and retrieved watermark ')



error_originalAsset_and_asset_after_adding_watermark=MSE(I2,I);

error_originalAsset_and_asset_after_removing_watermark=MSE(image_after
_removing_watermark,I);

error_originalWatermark_and_watermarkRetrieved=MSE(watermark,W);

error_originalImage_and_watermarkRetrieved=MSE(watermark,I);
```


भारत सरकार
GOVERNMENT OF INDIA
దండు ప్రదీప్
Name XXXX
DOB: XX-XX-XXXX
Gender: MALE
0000 1111 2222
आधार - आम आदमी का अधिकार

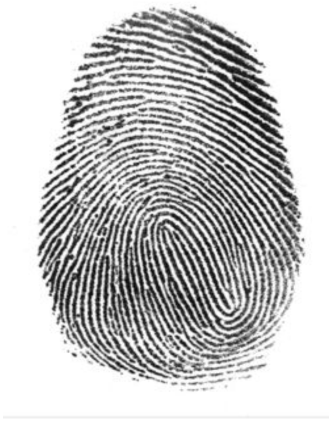
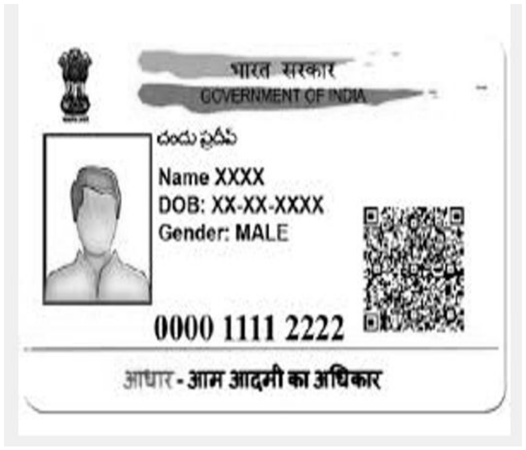


Figure 11: Original asset image and original watermark image.


भारत सरकार
GOVERNMENT OF INDIA
దండు ప్రదీప్
Name XXXX
DOB: XX-XX-XXXX
Gender: MALE
0000 1111 2222
आधार - आम आदमी का अधिकार

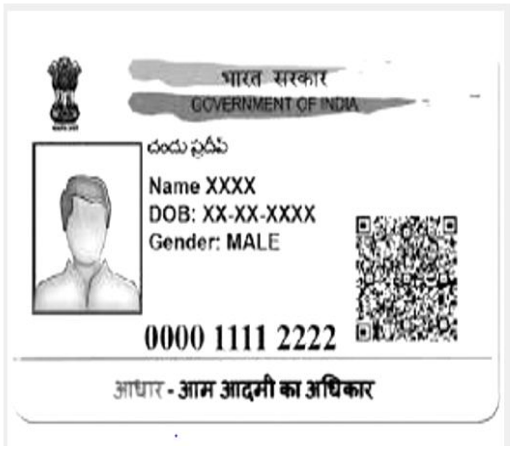
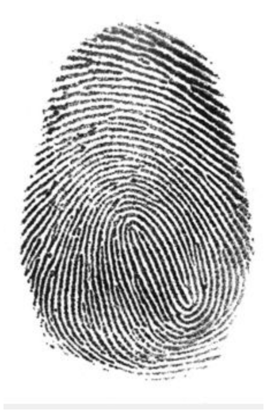

भारत सरकार
GOVERNMENT OF INDIA
దండు ప్రదీప్
Name XXXX
DOB: XX-XX-XXXX
Gender: MALE
0000 1111 2222
आधार - आम आदमी का अधिकार

Figure 12: Original image and image after adding watermark.



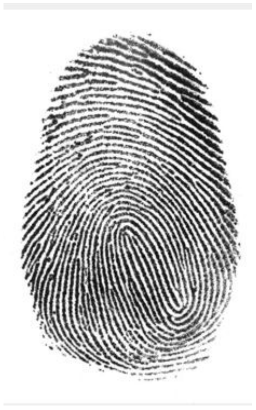


Figure 13: Original watermark image and retrieved watermark image.

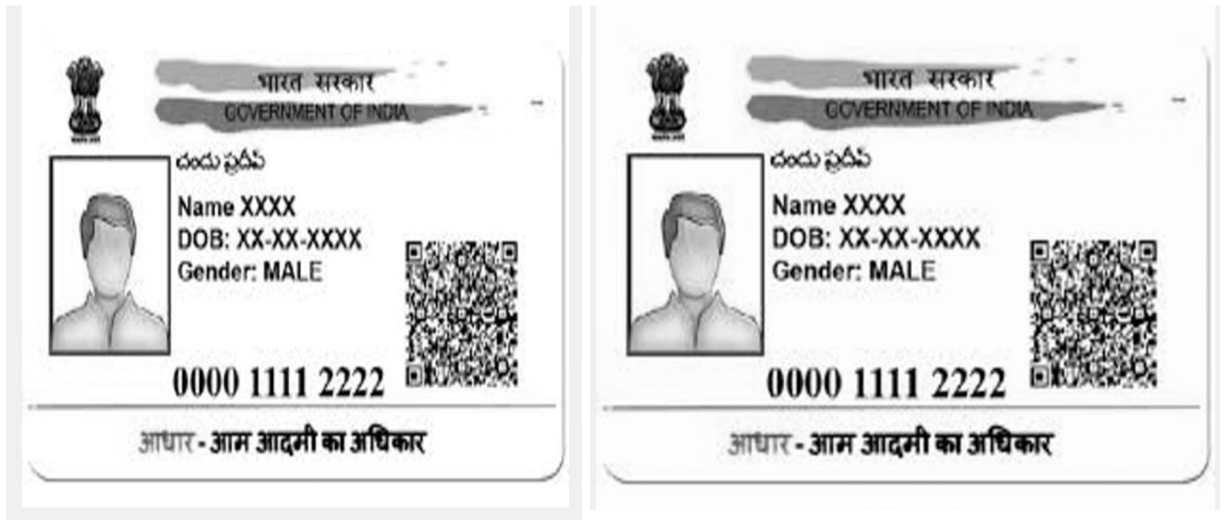


Figure 14: Original image and image retrieved after removing the watermark.

Mean Square Error analysis:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - X_i^*\right)^2$$

As already shown in the spatial domain, similar code was used for error analysis.

Interceptibility:

We will take MSE of the two images to see if what is the error.

1. Original asset image and image after watermarking

   Error between original Asset and asset after adding watermark = 0.0082

2. Original watermark image and retrieved watermark

   Error between original watermark image and retrieved watermark = 2.7319e-31

3. Original asset image and image after removing watermarking

   Error between original asset image and image after removing watermark = 8.2301e-05

As we can clearly see the MSE of the original watermark and the retrieved watermark is approximately zero.

RESULTS

We've successfully applied the 2 methods of Digital Image Watermarking that are Bit-plane slicing and Watermarking using Discrete Cosine Transform. We got extremely efficient results using the Bit-plane slicing as well as DCT approach.

The subsequent error analysis between an external fingerprint sample and the extracted watermark fingerprint was also done as shown above, following the Bit-plane slicing method and the effectiveness is evident as the Mean Square Error (MSE) was zero when the fingerprints matched, while MSE was non-zero in case of a mismatch.

In addition. following the DCT method, we can see that the MSE between the original fingerprint and retrieved fingerprint is approximately zero.

FUTURE SCOPE

The work can be extended in future to the field of Cryptography where the watermarked images can be kept secure by cryptographic methods like RSA or ECC, and transmission can be done in the same secured forms to prevent vulnerability.

REFERENCES

1. Digital Image Processing (3rd edition), Rafael C. Gonzalez, Richard E. Woods
2. Fingerprints were taken from www.google.com/images

3. Cox, J.; Miller, M. L.; Bloom, J. A.; Fridrich J. & Kalker T. (2008). Digital Watermarking and Steganography, Morgan Kaufmann Pub., Elsevier Inc. Barni M. & Bartolini F. (2004). Watermarking Systems Engineering, Marcel Dekker Inc., Italy

4. Cox, J.; Kilian, J.; Leighton F. T. & Shamoon T. (1997). Secure spread spectrum watermarking for multimedia. IEEE Transactions on Image Processing, Vol. 6, No. 12,(December 1997), pp. 1673-1687 www.intechopen.com 480 Engineering Education and Research Using MATLAB

5. Vatsa, M.; Singh, R.; Noore, A.; Houck M. M. & Morris K. (2006). Robust biometric image watermarking fingerprint and face template protection. IEICE Electronics Express, Vol. 3, No. 2, pp. 23-28

6. Wang, Z.; Bovik, A. C.; Sheikh, H. R. & Simoncelli E. P. (2004).Image quality assessment: From error visibility to structural similarity. IEEE Trans. Image Processing, vol. 13, no. 4, pp. 600-612.