Secure Transmission of Grayscale Images using Discrete Fourier Transform

Pankesh Bamotra, Prashant Dwivedi

Abstract — The paper presented here deals with image encryption using the well-known algorithm of discrete Fourier transform (DFT). The DFT of real values has a special property called the conjugate symmetry property X (N-m) = X*(m) | m = 0, 1,..N-1 which is the basis for this paper. The grayscale pixel values of the image to be encrypted are read into a matrix. Using this property we need to find the 1D -DFT for only the first ⌈N/2⌉ terms. Then we make two images of size N/2*N which are termed as 'real image' and 'imaginary image' using suitable modifications. The two images are appended after encrypting their pixel values using two key values which serve as the shared secret between two parties. On the receiver side the two images are separately read and their pixel values are retrieved and decrypted and the image can be regenerated by finding the inverse 1D-DFT of the obtained pixel values.

Index Terms — Complex conjugate symmetry, DFT, Grayscale images, IDFT, Image encryption.

I. INTRODUCTION

Image encryption has found its use at many places where security of the graphical images is an issue. The popular use has been stenography where unlike cryptography the message in any form is not at all visible to the eavesdropper. Here in this paper we have exploited a very interesting property of discrete Fourier transform (DFT) which is known as complex conjugate symmetry property. This property allows us to calculate only ⌈N/2⌉ terms of the DFT while rest can be computed using the equations as X (N-m) = X* (m). To send an encrypted image we first compute the grayscale pixel values of the image and store it as a huge matrix (512x512, preferably some power of 2). Next we compute the DFT of the grayscale values and obtain the N/2*N terms. This gives us the DFT elements as complex numbers. We then make two images of size N/2xN which we call as real image and imaginary image. The two images are appended and transmitted to the intended user. The receiver would scan the two images as per the described algorithm and then decrypts and reconstructs the image to obtain the original image. We have used 1-D DFT because many other parts of the algorithm would require the input data to be linear rather than in 2-D matrix.

II. CONJUGATE SYMMETRY PROPERTY OF DFT

DFT is a mathematical transform used in Fourier analysis. The motive is to convert a given time-domain discrete signal into a signal in frequency domain. This is necessary because of several reasons like efficient processing, mathematical simplicity and computational relevance. The DFT of a discrete time signal x(n) is given by the equation.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}$$

Where k = 0, 1… N-1, N is size of the sample.

Now if the input signal x(n) is a real-valued discrete time signal then as the complex conjugate symmetry property of DFT says,

$$X(N-k) = X^*(k)$$

Where X* denotes the complex conjugate of X(k).

III. METHODOLOGY

To begin with we first scan the input grayscale image in horizontal direction while retracing to next scan line after scanning till the width. We store these pixel values into a file. After this step we will have a file containing a NxN sized matrix form of the pixel values ranging from 0-255. Next we find the DFT of the grayscale values obtained line by line while restricting the DFT computation upto first ⌈N/2⌉ values. This gives us a DFT of grayscale image with total N/2xN terms which are complex numbers. Next we form two arrays arr [] and brr [] each having real and imaginary part of the complex term respectively. Now since these two arrays may have both positive and negative terms we have to convert each term into a positive quantity. This is achieved by adding absolute value of the minimum value (R_m and I_m) of each array to each term in the array. These terms are used as the key between the communicating parties to encrypt and decrypt the graphic information. After normalizing the terms into positive quantities we now represent the two arrays as combination of two images. To achieve this we first converted the values in each array to hexadecimal values followed by appending required number of 0s to convert it into a legitimate RGB hex code which is then displayed as an image of blue dots. The two images then obtained are appended to each other. This process has been shown in Fig. 1. This amalgamated image is transmitted to the receiver end who first scans the image as in a specific pattern as shown in Fig. 2. The result is again two arrays arr [] and brr [] which are then decrypted back to the original terms using the key value pair (R_m and I_m). After this the inverse DFT is calculated using both the array values giving N/2xN terms which are extrapolated using the conjugate symmetry property giving a total of NxN terms. The inverse DFT gives the values of the original grayscale pixel values which are then converted to give the original image. In section V we discuss some comparison measures between the original image and the image obtained after performing inverse DFT.

Manuscript Received on November, 2012.
Pankesh Bamotra, SCSE, VIT University, Vellore, Tamil Nadu, India..
Prashant Dwivedi, SCSE, VIT University, Vellore, Tamil Nadu, India.

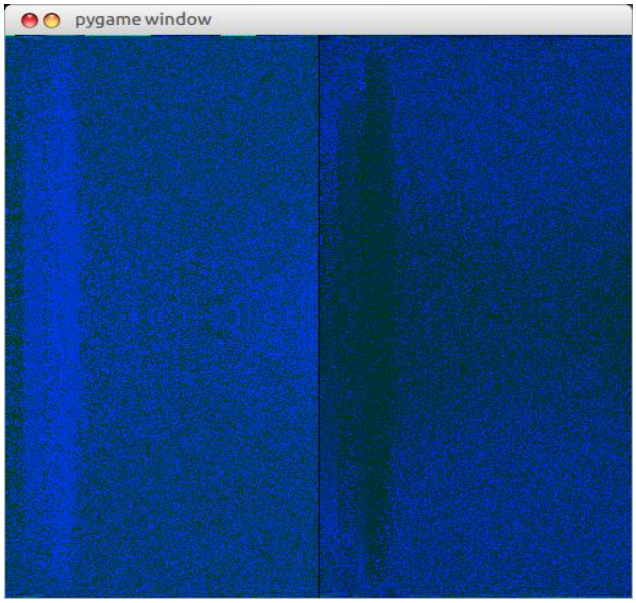

Fig. 1 Combination of real and imaginary image

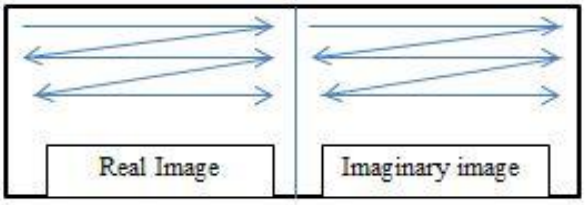

Fig. 2 Scanning pattern

IV. IMPLEMENTATION IN PYTHON

```
from cmath import exp, pi
import csv
from Tkinter import *
import pygame
import re
```

#DFT/ FFT function

```
def fft(x):

    N = len(x)
    if N <= 1: return x
    even = fft(x[0::2])
    odd =  fft(x[1::2])
    return [even[k] + exp(-2j*pi*k/N)*odd[k] for k in
xrange(N/2)] + \
          [even[k] - exp(-2j*pi*k/N)*odd[k] for k in
xrange(N/2)]
```

#Inverse DFT/ FFT function

```
def  ifft(x):

    N = len(x)
    if N <= 1: return x
    even = ifft(x[0::2])
    odd =  ifft(x[1::2])
    return ([even[k] + exp(2j*pi*k/N)*odd[k] for k in
xrange(N/2)] + \
          [even[k] - exp(2j*pi*k/N)*odd[k] for k in
xrange(N/2)])
def app(n):
    n=n.replace("0x","")
    difference = 6-len(n);
    append=["0","00","000","0000","00000","000000"];
    n=append[difference-1]+n;
    return n
def dep(n):
    n=re.sub(r'^0{1,4}',"",n)
    return n
```

#Pixels.txt has the grayscale values of the image

```
file = open("pixels.txt","r")
f = open("fft.csv","w")
for line in file.readlines():
    arr=[]
    for x in line.split():
        arr.append(int(x))
    wr = csv.writer(f, quoting=csv.QUOTE_ALL)
    wr.writerow(fft(arr))
f.close()
i=0+0j
fx=open("final","w")
f = open("fft.csv","r")
count =0
count1 =0
for row in f.readlines():
        count=0
        for element in row.split(","):
          element= str(element).replace("(","")
          element= str(element).replace(")","")
          element= str(element).replace(" ","")
          fx.write(element)
          if count<511:fx.write("\n")
          count=count+1
fx.close()
arr_real=[]
arr_imag=[]
c=0
fx=open("final","r")
for x in fx.readlines():
    x=x.replace("\n","")
    x=x.replace("\"","")

 if(x.replace("\s+","")==""):continue
    arr_real.append (int(complex(x).real))
    arr_imag.append (int(complex(x).imag))
    c=c+1
i=0
```

#add and bdd serves as the keys (R_m and I_m)

```
add=abs(min(arr_real))
bdd=abs(min(arr_imag))
while i<len(arr_real):
    arr_real[i]+=add
    i=i+1
i=0
while i<len(arr_real):
    arr_imag[i]+=bdd
    i=i+1
fo=open("afile","w")
i=0
j=0
```

```
ar=[]
index=0
for X in xrange(512):
    ar.append([])
    for Y in xrange(512):
        ar[X].append(app(str(hex(arr_real[index]))))
        index=index+1
for X in xrange(512):
    fo.write(str(ar[X])+"\n")
fo.close()
foo=open("bfile","w")
br=[]
index=0
for X in xrange(512):
    br.append([])
    for Y in xrange(512):
        br[X].append(app(str(hex(arr_imag[index]))))
        index=index+1
for X in xrange(512):
    foo.write(str(br[X])+"\n")
foo.close()

screen = pygame.display.set_mode((512,512))
clock = pygame.time.Clock()
while 1:
    clock.tick(60)
    for event in pygame.event.get():
    if event.type == pygame.QUIT:system.exit()
    for x in xrange(256):
        for y in xrange(512):
            PO=ar[x][y]
            red = int(PO[0:2],16)
            green = int(PO[2:4],16)
            blue = int(PO[4:6],16)
            screen.set_at((x, y), (red, green, blue))
    x_dir=0
    y_dir=0
    for x in xrange(257,512):
        for y in xrange(512):
            PO=br[x_dir][y_dir]
            red = int(PO[0:2],16)
            green = int(PO[2:4],16)

 blue = int(PO[4:6],16)
            y_dir=y_dir+1
            screen.set_at((x, y), (red, green, blue))
        x_dir=x_dir+1
        y_dir=0
        pygame.display.flip()

ibr = []
ind = 0

#inverse DFT is performed here
results = open("result.txt","w")
for X in xrange(512):
    ibr.append([])
    for Y in xrange(512):
        IMAG=int(dep(br[X][Y]),16)-bdd
        REAL=int(dep(ar[X][Y]),16)-add
        CMP = complex(REAL,IMAG)
        ibr[X].append(CMP)
    IDFT = ifft(ibr[X])
    for ilu in xrange(len(IDFT)):
        results.write(str(int(IDFT[ilu].real/512))+" ")
    results.write("\n")
    results.close()
```

V. RESULTS

In our case, we took a 512x512 image (lena.bmp) Fig 3. Next we read the grayscale values into a file named pixels.txt containing values ranging from 0-255. After this we performed the DFT using FFT algorithm to obtain the combined images i.e. real and imaginary image using the key values R_m and I_m. Then we decrypted the image values after scanning it in zigzag pattern (see Fig. 5). At last inverse DFT was performed to obtain the original image pixel values.

Fig. 3 Lena.bmp (Original)

Fig. 4 Reconstructed image PSNR = 52.7203

The steps of the process can be summarized in the following diagram:-

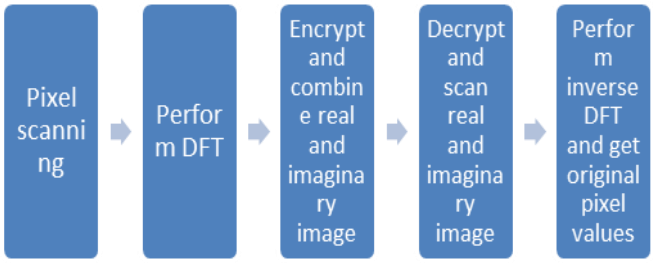

Fig. 5 Procedure flow

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

VI. CONCLUSION

The complex conjugate symmetry property of the DFT was successfully used to encrypt a grayscale image. Initially we calculate DFT only for half of the terms while rest can be extrapolated using the property. The results of the simulation were very impressive and gave a PSNR (Peak Signal to Noise Ratio) of 52.7203 which is considered to be very good. The algorithm employed here can be used for various purposes wherever there is a need for encrypting the images to prevent being seen by the eavesdropper. This is obviously taken care of, by using the key values which can encrypt and decrypt the original image.

REFERENCES

1. Vilardy, J.M.., "Digital image phase encryption using fractional Fourier transform", Electronics, Robotics and Automotive Mechanics Conference, 2006.
2. John A. Stuller, 'Introduction to signals and systems', Cengage learning, 2010
3. Cornell University library archives, http://arxiv.org/list/cs/recent.

AUTHORS PROFILE

Pankesh Bamotra, is a student of B.Tech in computer science and engineering studying in final year at VIT University, Vellore, Tamil Nadu. He has keen interest in network security and other areas interest include concurrent and distributed systems. He is an active member of Computer Society of India.

Prashant Dwivedi, is a student of B.Tech in computer science engineering studying in final year at VIT University, Vellore, Tamil Nadu. His interests include Data mining, network security and machine learning. He has co-authored paper with Pankesh Bamotra. He is an active member of Computer Society of India.