%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Decryption Algorithm

%from c1

%

%

%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% from p1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

p1 = zeros(256,256);

%taking size of the image

[a,b] = size(p1);

%initiating shuffled image matrix

shuffled = zeros(a,b, 'uint8');

twobytwo = zeros(2,2);

matrixA = [1 1; 1 2];

%initiate A^n as I

%eye(2,2) is identity matrix of 2X2 dimensions

matrixApowerN = eye(2,2);

% We are taking num\_iter as input and calculating A^num\_iter

for j = 1:5

matrixApowerN = matrixApowerN \* matrixA;

end

for k = 1:2

for l = 1:2

matrixApowerN(k,l) = mod (matrixApowerN(k,l), 256);

end

end

%We have now obtained the matrix M, that is, matrixApowerN

%For each pixel, [x' y'] = M[x y] mod256

%Calculating shuffled image pixels using above relation minus the mod256

for m = 1:a

for n =1:b

twobytwo = mod(matrixApowerN \* [m; n], 256);

shuffled(twobytwo(1) + 1, twobytwo(2) + 1) = p1 (m, n);

end

end

%lorentz chaotic mapping solutions lorentz(n,level,s,r,b,x0,y0,z0,h)

chaotic = lorentz(a\*b,0,35,28,3,1,1,1,0.001);

%Obtaining final sequence K with ( |x - floor(x)| X 10^14 )mod256

for i = 1:a\*b

chaotic(i) = floor(mod( (abs( chaotic(i) - floor(chaotic(i)) ) \* 1e14), 256 ));

end

%making shuffled matrix as 1d matrix and naming as shuffled2

shuffled2 = shuffled(:);

shuffled2 = uint8(shuffled2);

%converting decimal to binary

shuffled2 = de2bi(shuffled2);

chaotic = de2bi(chaotic);

%encrypted matrix = shuffled XOR key sequence

encrypted\_matrix = xor(shuffled2, chaotic);

%to make it into a 1D matrix from 2D of [65536,8]

reshape(encrypted\_matrix,[65536\*8, 1]);

%converting the 65536\*8 binary matrix into a\*b matrix after intialisation

encrypted\_image = zeros(a, b, 'uint8');

count = 1;

for index = 1 : 8 : 256\*256\*8

substring = encrypted\_matrix(index:index+7);

substring = num2str(substring);

encrypted\_image(count) = bin2dec(substring);

count = count+1;

end

c1 = encrypted\_image;

p1 = uint8(p1);

%whos c1 p1

key = bitxor(c1,p1,'uint8');

keyf = key;

%whos key

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%from p2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

p2 = zeros(256,256);

p2(2,2) = uint8(1);

p2(2,3) = uint8(2);

%taking size of the image

[a,b] = size(p2);

p2 = uint8(p2);

%disp(p2);

%whos p2

%initiating shuffled image matrix

shuffled = zeros(a,b, 'uint8');

twobytwo = zeros(2,2);

matrixA = [1 1; 1 2];

%initiate A^n as I

%eye(2,2) is identity matrix of 2X2 dimensions

matrixApowerN = eye(2,2);

% We are taking num\_iter as input and calculating A^num\_iter

for j = 1:5

matrixApowerN = matrixApowerN \* matrixA;

end

for k = 1:2

for l = 1:2

matrixApowerN(k,l) = mod (matrixApowerN(k,l), 256);

end

end

%We have now obtained the matrix M, that is, matrixApowerN

%For each pixel, [x' y'] = M[x y] mod256

%Calculating shuffled image pixels using above relation minus the mod256

for m = 1:a

for n =1:b

twobytwo = mod(matrixApowerN \* [m; n], 256);

shuffled(twobytwo(1) + 1, twobytwo(2) + 1) = p2 (m, n);

end

end

%lorentz chaotic mapping solutions lorentz(n,level,s,r,b,x0,y0,z0,h)

chaotic = lorentz(a\*b,0,35,28,3,1,1,1,0.001);

%Obtaining final sequence K with ( |x - floor(x)| X 10^14 )mod256

for i = 1:a\*b

chaotic(i) = floor(mod( (abs( chaotic(i) - floor(chaotic(i)) ) \* 1e14), 256 ));

end

%whos shuffled

%making shuffled matrix as 1d matrix and naming as shuffled2

shuffled2 = shuffled(:);

shuffled2 = uint8(shuffled2);

%converting decimal to binary

shuffled2 = de2bi(shuffled2,8);

chaotic = de2bi(chaotic);

chaotic = uint8(chaotic);

%whos shuffled2

%encrypted matrix = shuffled XOR key sequence

encrypted\_matrix = xor(shuffled2, chaotic);

%to make it into a 1D matrix from 2D of [65536,8]

reshape(encrypted\_matrix,[65536\*8, 1]);

%converting the 65536\*8 binary matrix into a\*b matrix after intialisation

encrypted\_image = zeros(a, b, 'uint8');

count = 1;

for index = 1 : 8 : 256\*256\*8

substring = encrypted\_matrix(index:index+7);

substring = num2str(substring);

encrypted\_image(count) = bin2dec(substring);

count = count+1;

end

c2 = encrypted\_image;

%whos c2

c2 = de2bi(c2,8);

key = de2bi(key);

%c2 = str2num(c2);

%key = str2num(key);

%disp(c2);

%disp(key);

tf = isequal(k,c2);

s2 = bitxor(c2,key);

s2 = uint8(s2);

%whos c2 key s2

%disp(s2);

%disp(tf);

reshape(s2,[65536\*8, 1]);

di = zeros(a, b, 'uint8');

count = 1;

for index = 1 : 8 : 256\*256\*8

substring = s2(index:index+7);

substring = num2str(substring);

di(count) = bin2dec(substring);

count = count+1;

end

%whos decrypted\_image

%disp(di);

a = zeros(256,256,'uint8');

tf = isequal(a,di);

disp(tf);

%[row,col,v] = find(di);

[row,col,v] = find(di);

di(row(1),col(1)) = di(row(1),col(1))/2;

di(row(2),col(2)) = di(row(2),col(2))/2;

[row,col,v] = find(di);

di = mat2gray(di);

%[row,col,v] = find(di);

%figure

%imshow(di,[]);

%figure

%imhist(di);

m\_1(1,1) = row(1); %x1

m\_1(1,2) = col(1);%y1

m\_2(1,1) = row(2);%x2

m\_2(1,2) = col(2);%y2

m2 = mod(m\_2(1,1) - m\_1(1,1),256);

m1 = mod(m\_1(1,1) - m2,256);

m4 = mod(m\_2(1,2) - m\_1(1,1),256);

m3 = mod(m\_1(1,1) - m4,256);

M = zeros(2,2);

M(1,1) = m1;

M(1,2) = m2;

M(2,1) = m3;

M(2,2) = m4;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%

%

%Testing the Key obtained

%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Encrypting original image

%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%closing and clearing all open screens

%reading and opening the image

image = imread('D:\Shreyas\CryptoProject\lena.jpg');

%figure

%imshow(image)

%figure

%imhist(image)

%taking size of the image

[a,b] = size(image);

%initiating shuffled image matrix

shuffled = zeros(a,b, 'uint8');

twobytwo = zeros(2,2);

%number of iterations of catmap

num\_iter = input('Enter the number of arnold cat map iterations: ');

%arnold cat map

%Defining matrix A = [1 p; p pq+1]

p = input('Enter the value of p in arnold cat map: ');

q = input('Enter the value of q in arnold cat map: ');

matrixA = [1 p; p (p\*q+1)];

%initiate A^n as I

%eye(2,2) is identity matrix of 2X2 dimensions

matrixApowerN = eye(2,2);

% We are taking num\_iter as input and calculating A^num\_iter

for j = 1:num\_iter

matrixApowerN = matrixApowerN \* matrixA;

end

%We take modulus 256 on each element of matrix matrixApowerN

for k = 1:2

for l = 1:2

matrixApowerN(k,l) = mod (matrixApowerN(k,l), 256);

end

end

%We have now obtained the matrix M, that is, matrixApowerN

%For each pixel, [x' y'] = M[x y] mod256

%Calculating shuffled image pixels using above relation minus the mod256

for m = 1:a

for n =1:b

twobytwo = mod(matrixApowerN \* [m; n], 256);

shuffled(twobytwo(1) + 1, twobytwo(2) + 1) = image (m, n);

end

end

%lorentz chaotic mapping solutions lorentz(n,level,s,r,b,x0,y0,z0,h)

chaotic = lorentz(a\*b,0,35,28,3,1,1,1,0.001);

%Obtaining final sequence K with ( |x - floor(x)| X 10^14 )mod256

for i = 1:a\*b

chaotic(i) = floor(mod( (abs( chaotic(i) - floor(chaotic(i)) ) \* 1e14), 256 ));

end

%making shuffled matrix as 1d matrix and naming as shuffled2

shuffled2 = shuffled(:);

shuffled2 = uint8(shuffled2);

%converting decimal to binary

shuffled2 = de2bi(shuffled2);

chaotic = de2bi(chaotic);

%encrypted matrix = shuffled XOR key sequence

encrypted\_matrix = xor(shuffled2, chaotic);

%to make it into a 1D matrix from 2D of [65536,8]

reshape(encrypted\_matrix,[65536\*8, 1]);

%converting the 65536\*8 binary matrix into a\*b matrix after intialisation

encrypted\_image = zeros(a, b, 'uint8');

count = 1;

for index = 1 : 8 : 256\*256\*8

substring = encrypted\_matrix(index:index+7);

substring = num2str(substring);

encrypted\_image(count) = bin2dec(substring);

count = count+1;

end

c\_1 = encrypted\_image;

%Converting matrix into grayscale

encrypted\_image = mat2gray(encrypted\_image);

%figure

%imshow(encrypted\_image, [])

%figure

%imhist(encrypted\_image)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%

%Decrypting Image

%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c\_1 = de2bi(c\_1);

keyf = de2bi(keyf);

s = bitxor(c\_1,keyf);

s = uint8(s);

reshape(s,[65536\*8, 1]);

d\_i = zeros(a, b, 'uint8');

count = 1;

for index = 1 : 8 : 256\*256\*8

substring = s(index:index+7);

substring = num2str(substring);

d\_i(count) = bin2dec(substring);

count = count+1;

end

M\_i = inv(M);

M\_i = mod(M\_i,256);

original = zeros(256,256,'uint8');

for i = 1:256

for j =1:256

twobytwo = mod( M \* [i; j], 256);

original(twobytwo(1) + 1, twobytwo(2) + 1) = d\_i (i, j);

end

end

original = mat2gray(original);

figure

imshow(original);