%closing and clearing all open screens

close all;

clear all;

%reading and opening the image

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

figure

imshow(image)

imhist(image)

%initiating shuffled image matrix

shuffled = zeros(256,256);

twobytwo = zeros(2,2);

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

chaotic = lorentz(256\*256,0,35,28,3,1,1,1,0.001);

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

for i = 1:256\*256;

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

end

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

matrixA = [1 2; 2 5];

%initiate A^n as I

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

matrixApowerN = eye(2,2)

% We are taking n = 3 for now later make n as input and calculating A^n

for j = 1:3;

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 = 1:256;

for n =1:256;

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

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

end

end

%encrypted matrix = shuffled XOR key sequence

encrpyted\_matrix = bitxor(shuffled, chaotic)

%Converting matrix into grayscale

encrypted\_image = mat2gray(encrypted\_matrix);

figure

imshow(encrypted\_image)