FRACTAL BASED IMAGE ENCRYPTION

OUTLINE OF THE PROJECT-

The below steps gives us a brief whereabouts of the project. Fractals are never-ending complex patterns,which embed a smaller picture inside a bigger one and the cycle keeps repeating,for example - when you are standing between two parallel mirrors and each mirror reflects an image which then reflects an image causing infinite mirroring. Fractals are vaguely similar to this but may differ in shapes and structure.

As the main component for encryption revolves around randomness my idea was to incorporate fractals for encryption.I have decided to use the Mandelbrot set to generate the values of the key, thereby bringing in some chaos to the system in order for it to be secure.

A brief pseudocode:

1. input an image MxN

generate_image()

img = image.open('<>')

extract the dimensions of the image

width,height = M,N

2. Reordering pixels, shuffle the pixels by rows

suffle()

split the channels according to the number of channels

obtained after calculation

reorder the rows with a random permutation

obtain the shuffled image = shuffle_image

3. Multiply the original and shuffled image array

mod_img=img*shuffle_image [#using Galois Field of GF(2^8)]

$$

4. Generate key using mandelbrot set

generate_key()

$z_{n+1} = (z_n)^2 + c$, using this equation the key is generated

$z_n=0$

c= a random complex number (x+iy)

#We iterate the above equation for the given c,if the equation

#remains bounded c is in the mandelbrot set, if the equation goes

#to infinity c is not in the mandelbrot set

```
    check if c is in the mandelbrot set:
        if yes:
          take |c|, extract last 3 digits of mod(c)
          compute hash(using SHA-1) and record the last 2 bytes of the hash value
        if no:
          take |c|, extract last 2 digits of mod(c)
          compute hash(using SHA-1) and record the last 2 bytes of the hash value
   append these values to the key[]
5. Encryption-
      xor(mod_img,key)
```

An encrypted image is thus obtained.