Image Encryption Using Chaos Maps

Aarya Arun, Indu Rallabhandi, Rachana Jayaram

November 2019

1 Introduction

1.1 What is encryption?

The process of encoding a message so that only authorised parties can access it is called **encryption**.The process of encrypting an image is called image encryption.

1.2 What is chaotic behavious?

Erratic-looking temporal behaviour is a superficial indicator of possible chaos. Chaotic sequences look haphazard but are **deterministic**. Chaotic sequences are also extremely sensitive to small changes in the initial condition. In fact, the 'Butterfly Effect' is based on this very sensitivity. Most chaotic sequences are calculated iteratively, with the value in each iteration being dependent on the previous one.

1.3 Chaos maps

Chaotic systems are a simple sub-type of nonlinear dynamical systems. They may contain very few interacting parts and these may follow very simple rules, but these systems all have a very sensitive dependence on their initial conditions. When it comes to chaos, a **map** (or a function) is an equation or a rule which specifies how a dynamic system evolves with time.

An **n-dimensional map** is a function that deals with n features.

1.4 Chaos Maps for encryption

Traditional encrypting mechanisms AES and RSA exhibit some drawbacks and weakness in the encryption of digital images and high computing

- Large computational time for large images
- High computing power for large images

Consequently, there might be better techniques for image encryption.

A few chaos based algorithms provide a good combination of speed, high security complexity, low computational overheads. Moreover, certain chaos-based and other dynamical systems based algorithms have many important properties such as

- sensitive dependence on initial parameters
- pseudorandom properties
- ergodicity
- non periodicity

Chaos technology encrypts images such that the possibility of deciphering is reduced greatly as the cipher text presents a randomness. Chaos based encryption is a very important means of modern digital encryption. The statistical characteristic if the original image is transformed, thereby increasing the difficulty of unauthorized breaking of the encryption.

2 Proposal

Images are one of the most important and popular forms of multi-media. Along with being stored on devices like phones, CD's, pen drives etc, images are shared across multiple platforms and across the internet. With this kind of large scale usage of images, it has become more difficult to keep images secure and safe from unauthorised users. With a more secure system, images can be shared and used more confidently. Most encryption algorithms were used to encrypt text messages. The conventional algorithms are not suitable for image encryption as images generally have large data capacity and require large computational volume. An efficient algorithm for image encryption is needed to make it difficult for unauthorised users to view or use the images.

3 Implementation

The above proposal can be implemented in the following way:

Arnold cat map Arnold's Cat chaotic mapping of two dimensions can be used to change the position of the pixel of the image without removing any information from the image. The pixel image can be assumed by:
S = {(x, y) $|x, y = 0, 1, 2 ... N - 1\}$

The 2-D image of Arnold's cat map can be written as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = A \times \begin{bmatrix} x \\ y \end{bmatrix} \times (modn)$$

$1 \quad p$

$q \quad (pq+1) \times \begin{bmatrix} x \\ y \end{bmatrix} \times (modn)$

Where p and q are positive integers, the determinant (A) = 1. (x ', y') is the new position of the original pixel position (x, y), when Arnold's Cat Map algorithm performed once.
Implementation of Arnold cat Map:

```
ArnoldCatTransform(img, num):
rows, cols
ch <- img.shape
n <-rows
arnold$_img < -np.zeroes([rows, cols, ch])$
$forxinrange(0, rows):$
    $foryinrange(0, cols):$
        $arnold_img[x][y] < -img[(x+y)$
$returnarnold_img$

$ArnoldCatEncryption(imageName, key):$
$img < -cv2.imread(imageName)$
$foriinrange(0, key):$
    $img < -ArnoldCatTransform(img, i)cv2.imwrite(imageName.split('.')[0]+$
$"_Arnoldcat Enc.png", img)$
$returnimg$
```

Duffing map:

$TheDuffingMapisa2Dimplementationofachaosmapthatfollowsthefollowingfunction:$

$x_{n+1} = y_n$
$y_{n+1} = -bx_n + ay_n - y^3$

$Thismaphaschaoticbehaviourforb = 0.2anda = 2.75.$

Algorithm to encrypt an image of size m:

Using this map we generate two random sequences savex$_1$, savey$_1$ and savex$_2$, savey$_2$ of size m^2 where savex$_i$ and savey$_i$ contain the x and y values generated from the function where each ith set has started from a different initial value of x and y.

x$_1$=-0.04
y$_1$=0.2

```
for i in range(1,m^2):
    x_{i+1} = y
    y_{i+1} = -0.2x + 2.75y - y^3
    savex_1=x_{i+1}
    savey_1=y_{i+1}
    x=x_{i+1}
    y=y_{i+1}
```

Similar iterations are run for initial values x_2=0.23 and y_2=-0.13 but saved to $savex_2$ and $savey_2$.

```
for i in range(0,m^2):
    h_1[i]=0 if savex_1 <savex_2, else 1
    h_2[i]=0 if savey_2 <savey_1, else 1
encryptedimage= bitwise-xor(h1,h2,imagematrix)
decryptedimage=bitwise-xor(h1,h2,imagematrix)
```

h1 and h2 can be found by knowing the initial values x_1, x_2, y_1,y_2, which makes them the secret keys for this encryption.

Henon map:
It is a 2-D dynamic system. Two different chaotic sequences are generated by the Henon chaotic map. These sequences are then applied to the row and column permutations of the original/plain image. XOR models are used to produce pixel values diffusion by unimodal skew tent map. Hussain's substitution box is used to substitute each pixel into a new random pixel in the last stage of the algorithm.

Consider:

$$x_(i+1) = y_(i+1) + 1 - \alpha x_i^2$$

$$y_(i+1) = \beta x_i$$

Where, the initial point is (x_0, y_0) and α, β are the initial parameters. Each point (x_n, y_n) is mapped to a new point $(x_(n+1), y_(n+1))$ through the Henon map.

Algorithm for Henon map implementation:

```
for i in range (m^2):
    $x_(i+1) = y_(i+1) + 1 - \alpha x_i^2$
    $y_(i+1) = \beta x_i$
    bit=0 if $x_i$<=0.4 else 1
```

(Only the value of x is used. This is appended as rows of lists of 8-bit numbers, so that we can have a bitwise xor with the image).

encryptedimage=bitwisexor(imagematrix,bitmatrix for R,G,B values in each pixel of imagematrix)
decryptedimage=bitwisexor(imagematrix,bitmatrix for R,G,B values in each pixel of imagematrix)

The bit matrix can be generated by knowing the inital values applied in the Henon map, making them the secret keys in this operation.

Logistic map with key mixing: The logistic map instead uses a nonlinear difference equation to look at discrete time steps. It's called the logistic map because it maps the population value at any time step to its value at the next time step.

The basic formula is: $X_(t+1) = r.X_t.(1 - X_t)$
where X_t is a number between zero and one that represents the ratio of existing population to the maximum possible population. The values of interest for the parameter r (sometimes also denoted) are those in the interval [0,4].

For this implementation we have included key mixing based on the paper. The initial values of the chaos map is recalculated after every pixel encryption based on the previous encryption value as well as the changed key value.

3.1 Proposal including modified paper implementation: Arnold-Henon composite map

Based on the paper "*A Chaotic Cryptosystem for Images Based on Henon and Arnold Cat Map*" by Ali Soleymani, Md Jan Nordin, and Elankovan Sundararajan, a modified version of the scheme proposed in the paper has been implemented as part of this project.

Algorithm:

- Use Henon map to generate two secret images A and B based on the secret key (i.e initial value of the Henon map) where the x values make up A and y values generated make up B and this is done m^2 times, so that it can correspond to each pixel in the original image using:

  *pixA.append(abs(math.floor(xN*paramlambda))%256)*
  *pixB.append(abs(math.floor(yN*paramgamma))%256)*

  where

  *paramlambda=math.floor(x*m*5000)*
  *paramgamma=math.floor(y*m*5000)*

  paramlambda and paramgamma can be found using x y, or the initial values, and therefore are not part of the secret key, as they can be derived from it.

- Simultaneously, we record values in arrays Pvalues and Qvalues as follows:

  *Pvalues.append(abs(math.floor(savethexmat[i]*math.pow(10,14)))%alpha)*
  *Qvalues.append(abs(math.floor(savetheymat[i]*math.pow(10,14)))*%beta)

  where
  alpha=26010
  beta=15080

  Here, 26010 and 15080 were randomly chosen because they were the dates for Republic Day and Independence Day. Thus, *alpha* and *beta* too come under the secret key.

- Run the Arnold-Cat map m/3 times on the original image with the matrix-multiplication algorithm mentioned earlier in the paper, but with different values of P and Q based on the iterator i, with the values being taken from the lists

  Perform a bitwise XOR of secret images A and B, and then run the Arnold-Cat algorithm on the xor-ed imaged 5m/7 times.

- Note: m/3 and 5m/7 are values chosen to be part of the alogirthm as the algorithm was found to be most robust for these values. As such, these can be found through knowing the image size and are public knowledge in terms of the values of m/3 and 5m/7 and do not form part of the secret key.

- Perform a bitwise XOR of the final images obtained in the previous step and the step before it. This is the encrypted image.

4 Analysis

The measure the effectiveness and the security of the encryption algorithms we have used three analyses - namely, Intensity histogram analysis, Adjacent pixel autocorrelation test and the key sensitivity tests.

4.1 Histogram Analysis

The ciphertext image histogram analysis is one of the most straight-forward methods of illustrating the image encryption quality. A good image encryption method tends to encrypt a plaintext image to a random incomprehensible form. Thus a good image encryption technique generates a cipher image that has a uniformly distributed intensity histogram.

Figures 3, 4 and 5 show the histograms for the various encryption schemes discussed. The Arnold cat encryption system has an identical intensity histogram to the original image. This is because the Arnold Cat method essentially just shuffles the pixels but does not change any pixel values. The logistic map

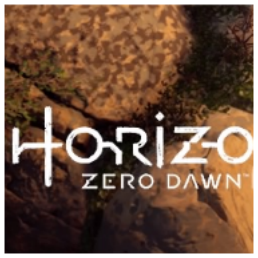

Figure 1: The original image

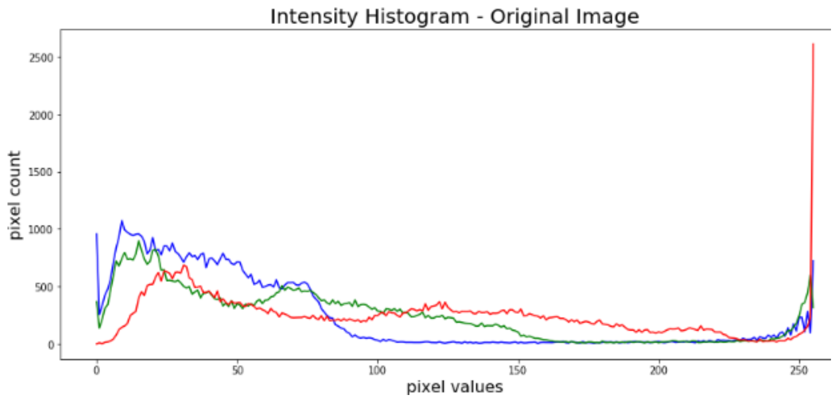

Figure 2: The intensity histogram for original image

based encryption system has the most uniform histogram indicating that is an effective encryption algorithm.

4.2 Adjacent Pixel Autocorrelation Test

Since images exhibit high information redundancy, it is desirable to have an encryption algorithm that breaks this redundancy. Thus as a metric of encryption performance we find the correlation between adjacent pixels in a direction (Horizontal, Vertical or Diagonal). We have considered the Horizontal direction.

For this test, 1024 random pixels are picked up from the image and its correlation between it's right most neighbour is found and plotted. For a good algorithm, the correlation plot should appear random with no discern able pattern.

Figures 6, 7, 8 and 9 shows the Adjacent pixel autocorrelation plots for the original image and its encrypted forms. shows that there is high information redundancy in the original image with a clear linear pattern in the graph. The Arnold cat map produces a cipher image with less obvious patterns. The outcome of the autocorrelation test for the Arnold cat map relies heavily on the number of iterations of transformations preformed. The autocorrelation graph

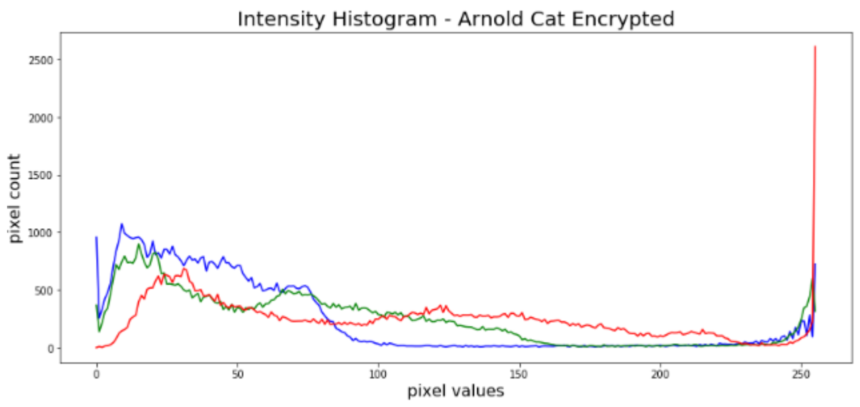

Figure 3: The intensity histogram for Arnold cat map

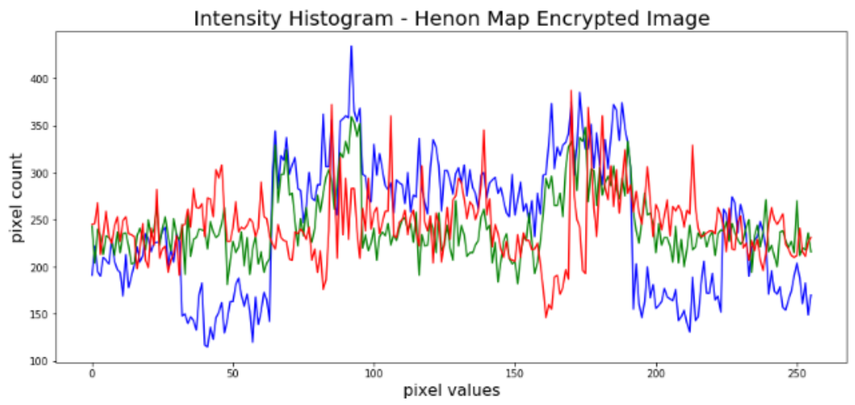

Figure 4: The intensity histogram for Henon map

for the image encrypted with logistic chaos maps seems to be more random than the graph corresponding to the Henon encrypted image, once again showing that logistic chaos maps with key mixing functions as a better algorithm.

4.3 Key sensitivity

An ideal image encryption algorithm should be sensitive with respect to the secret key i.e a small change in the key should produce a completely different encrypted image.

To test the key sensitivity the we encrypt the plain image with the three algorithms. We then try decrypting them with a slightly changed key. Figure ? shows us that for the Arnold cat algorithm, while a change in key doesn't decrypt the image correctly,the resultant image has discernible features that appear similar to the plain image.This is due to the fact that on applying a

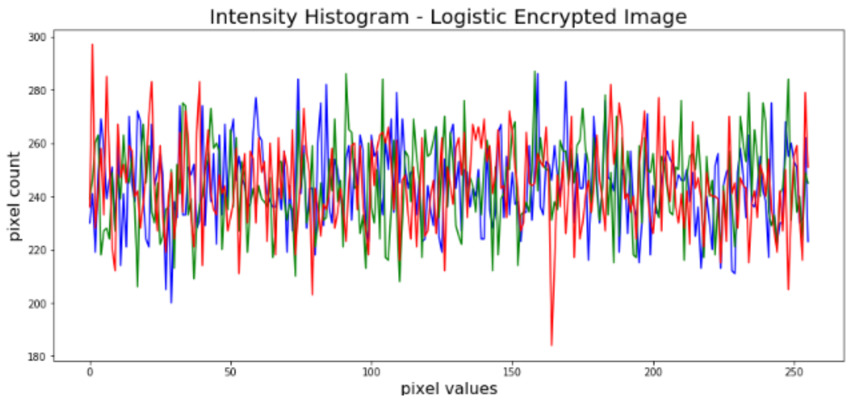

Figure 5: The intensity histogram for Logistic map

certain number of Arnold Cat transformations, the initial image can be easily recovered. By applying brute force it becomes possible for a person without a key to decrypt an image encrypted using Arnold cat maps. This proves that the Arnold cat algorithm, owing to its periodicity is unsafe for practical use. Both Henon map and Logistic map encryption appear to be sensitive to key changes.

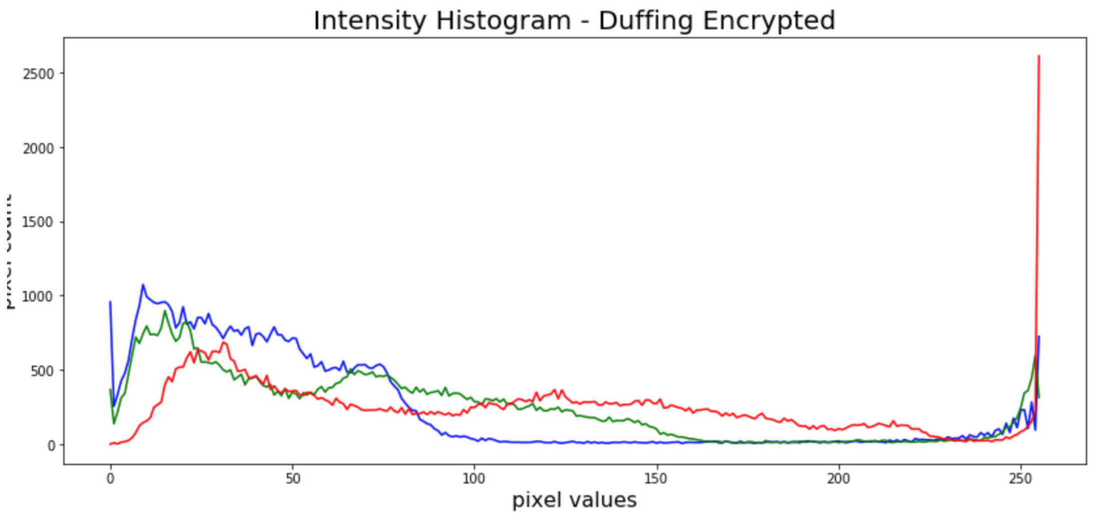

Figure 6: Intensity histogram for Duffing Map

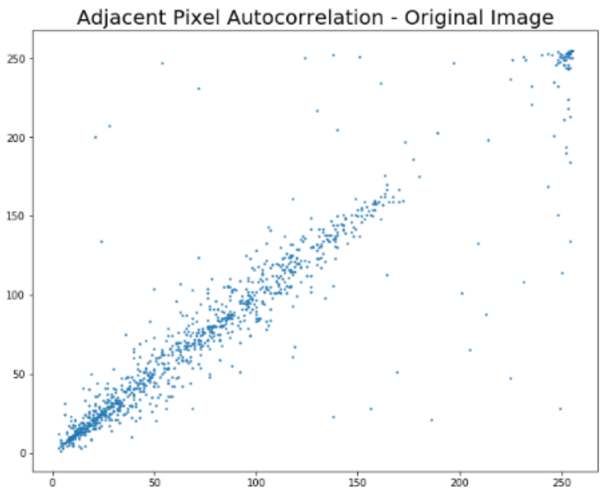

Figure 7: Adjacent pixel auto-correlation for the original image

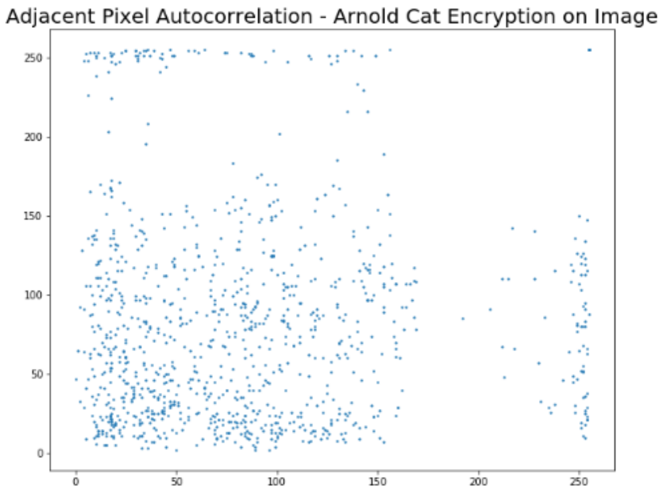

Figure 8: Adjacent pixel auto-correlation for the Arnold Map

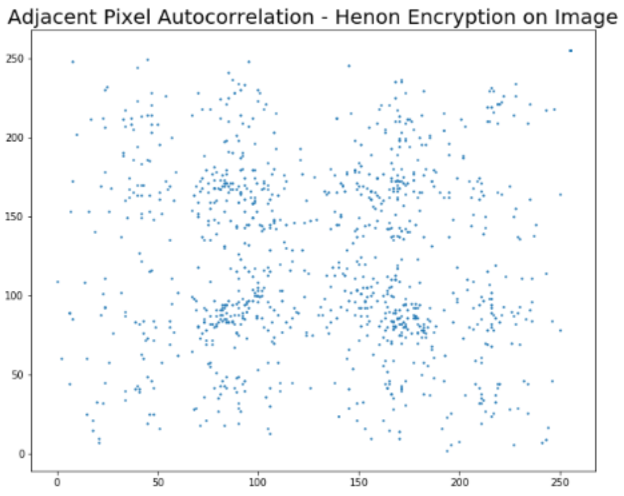

Figure 9: Adjacent pixel auto-correlation for the Henon Map

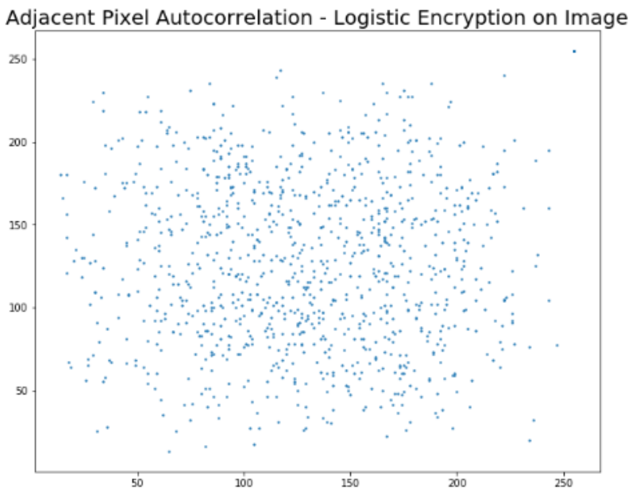

Figure 10: Adjacent pixel auto-correlation for the Logistic Map

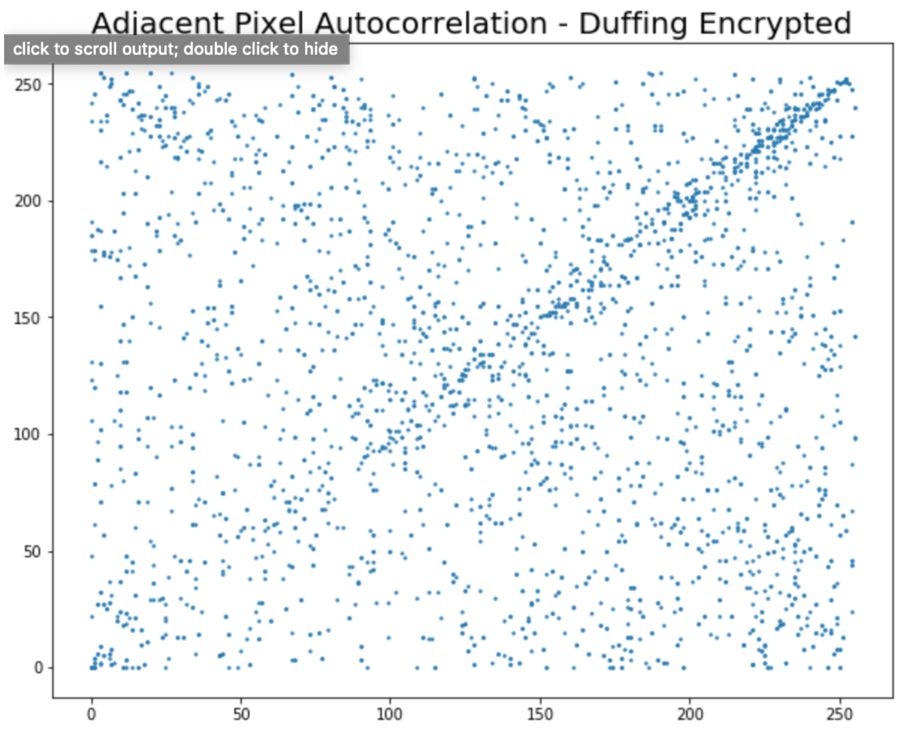

Figure 11: Auto-correlation plot for Duffing Map

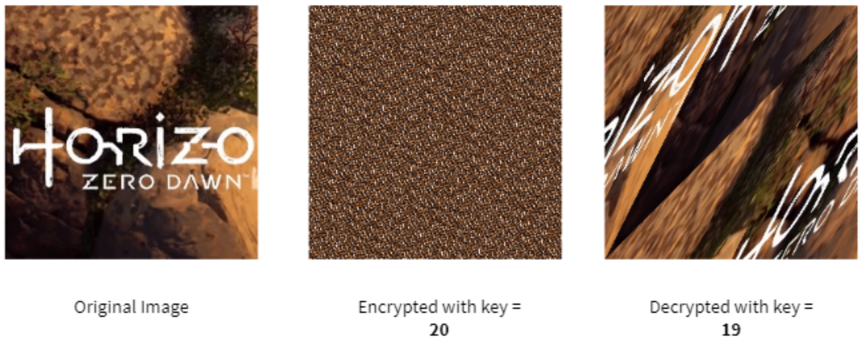

Figure 12: Key sensitivity for Arnold Map

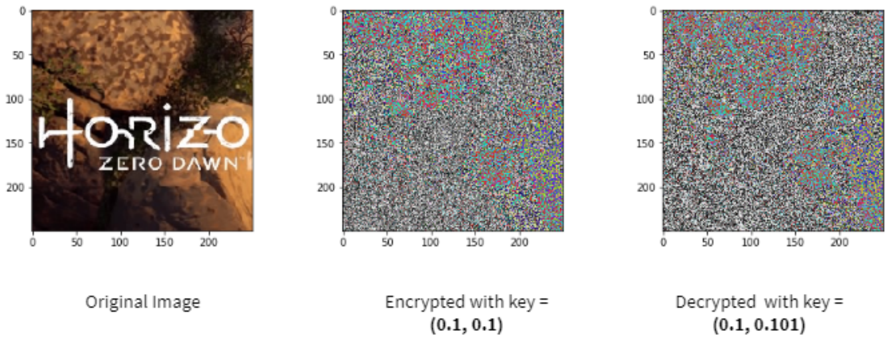
Original Image
Encrypted with key = (0.1, 0.1)
Decrypted with key = (0.1, 0.101)

Figure 13: Key sensitivity for Henon Map

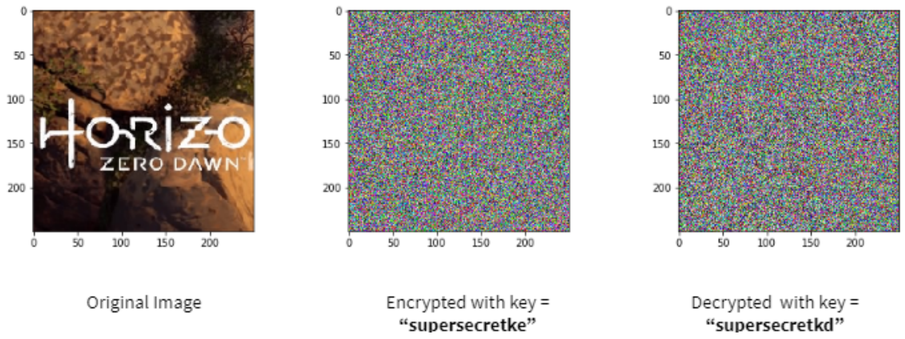
Original Image
Encrypted with key = “supersecretke”
Decrypted with key = “supersecretkd”

Figure 14: Key sensitivity for Logistic Map

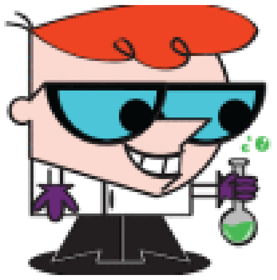

Figure 15: The original image for Arnold-Henon map

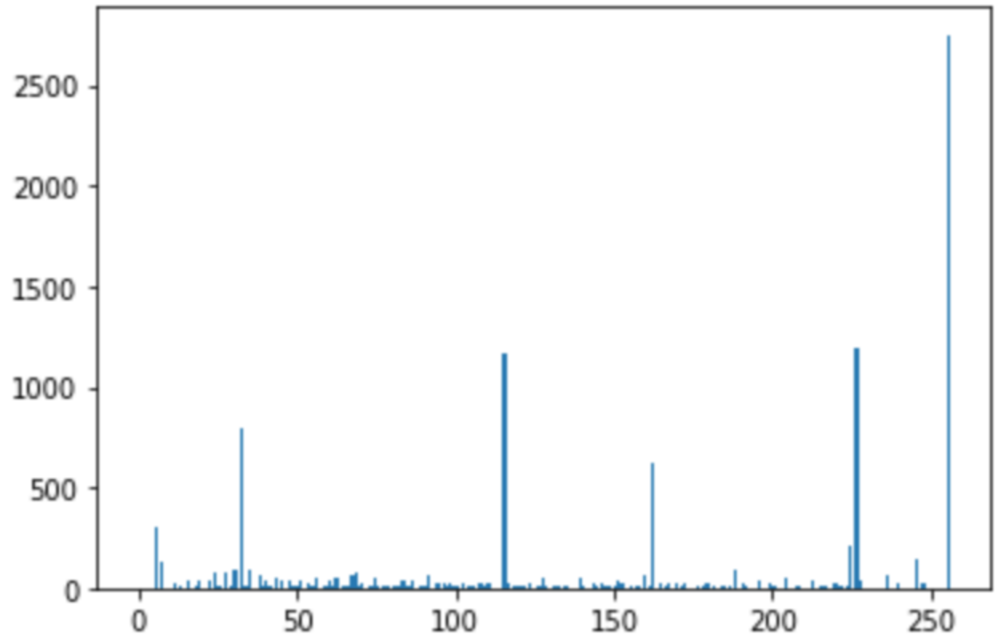

Figure 16: Histogram for original image before Henon-Arnold composite map

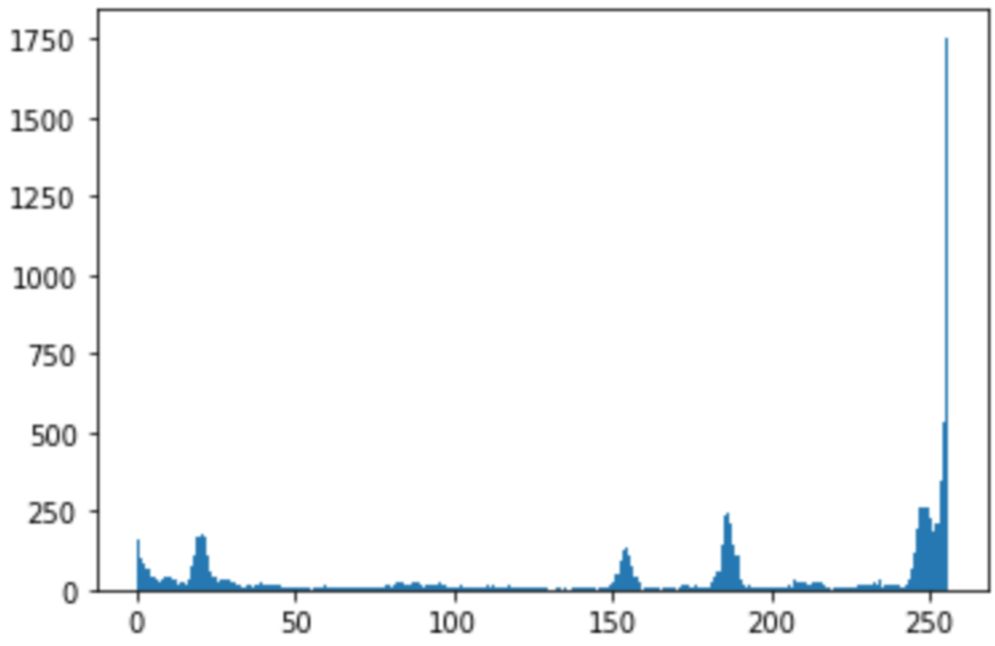

Figure 17: Histogram for image after Arnold encryption in the Henon-Arnold composite map

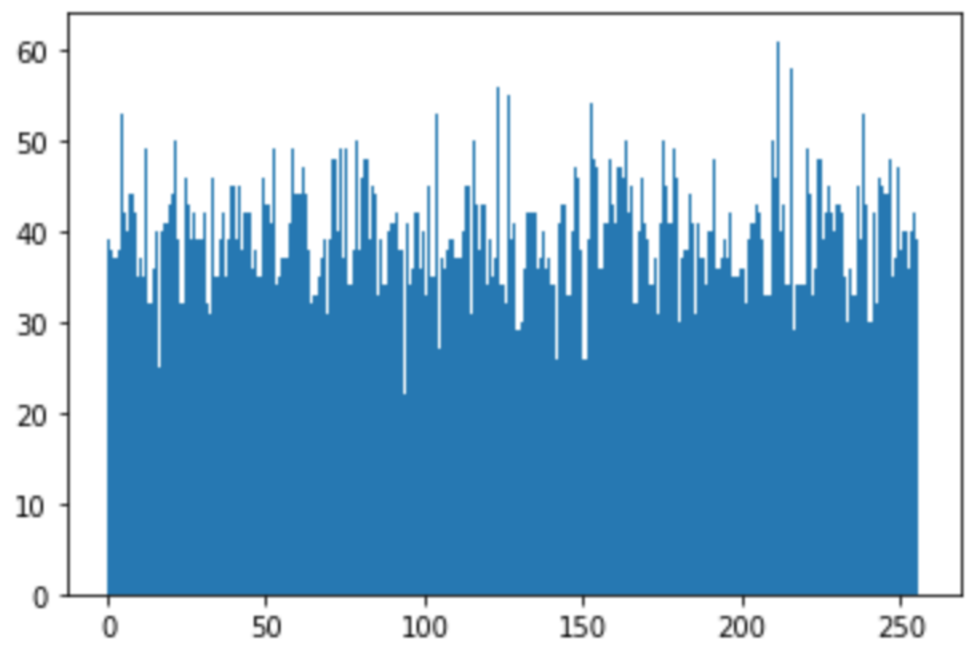

Figure 18: Histogram for Henon-Arnold composite

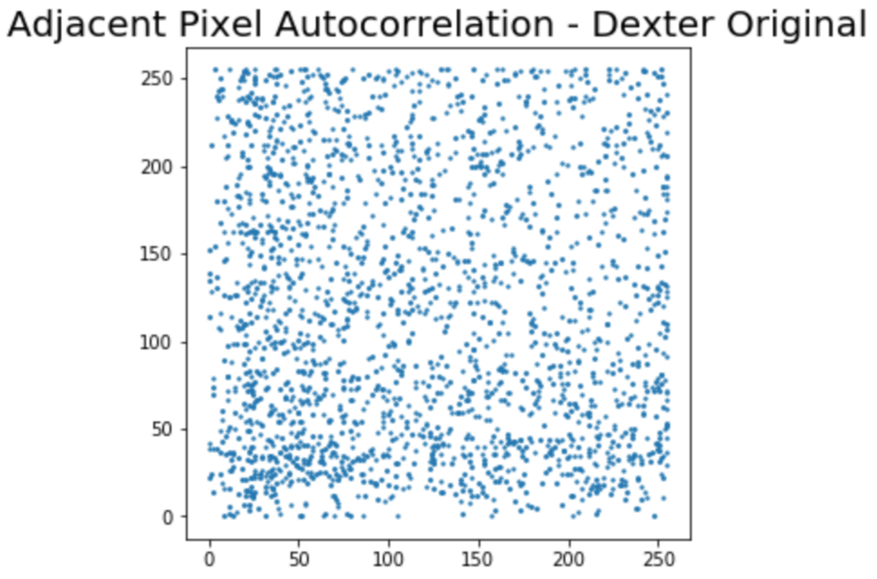

Figure 19: Auto-correlation for original image before Henon-Arnold map

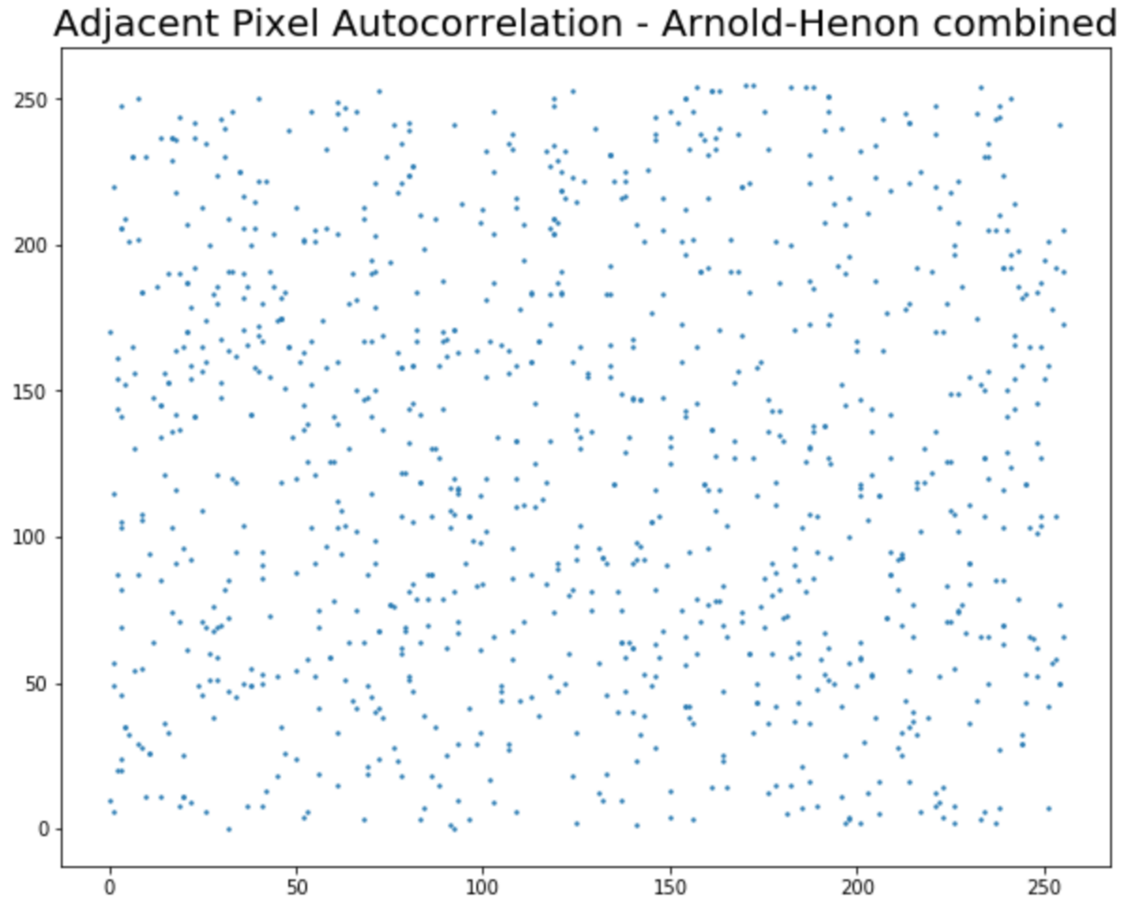

Figure 20: Auto-correlation plot for Henon-Arnold composite map