# Image Encryption Using Chaos

By Sean White

# What is Encryption

## Definition of Encryption:

Image Encryption is the process of preventing unauthorized access to important and confidential images by making the image illegible without the proper decryption key

## Properties of a good Encryption Algorithm

1- Must be relatively easy to encrypt and decrypt the image given an appropriate key

2-It must be computationally infeasible to encrypt and decrypt the image without the appropriate key

# Why Chaos is good for Encryption

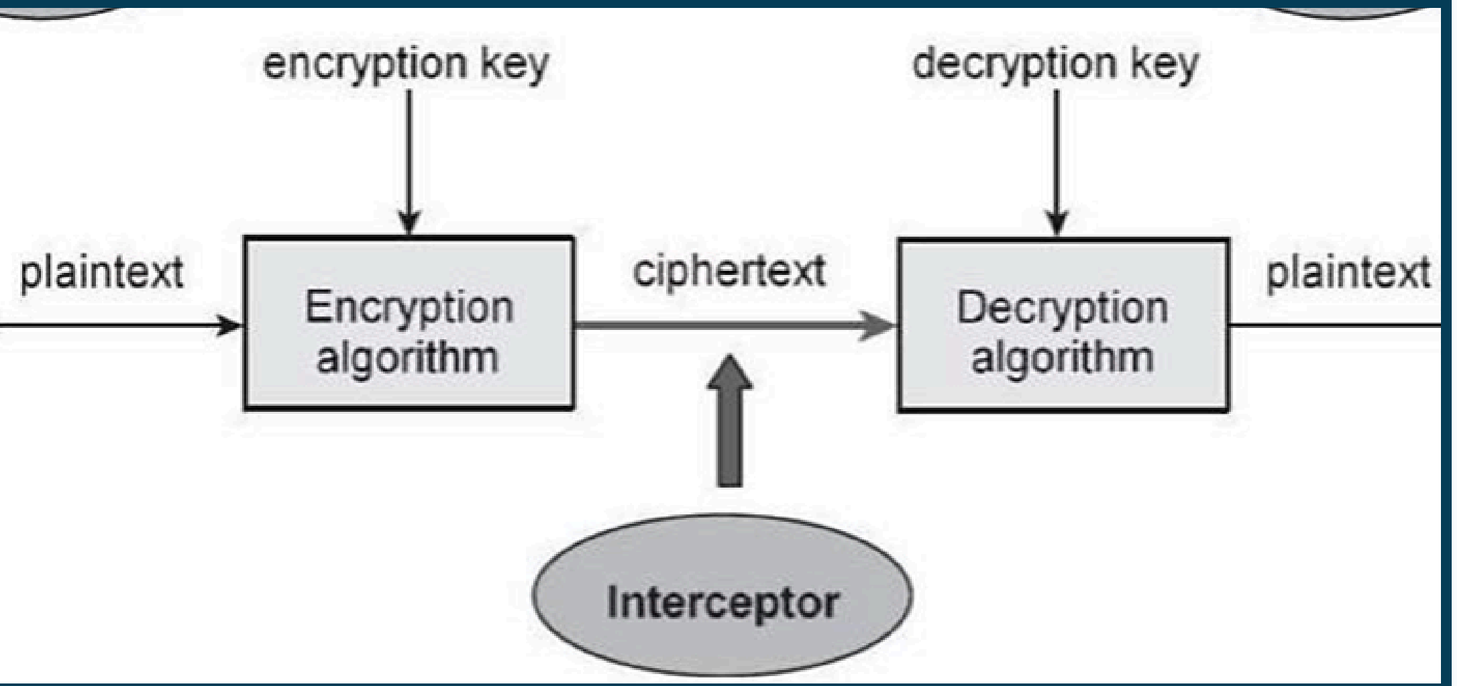
**Properties of  Chaos good for Encryption**

1- A chaotic system is a deterministic system.

2- A chaotic system is sensitive to initial conditions.

Main Idea : Initial Conditions are the key of the Encryption Algorithm

# How  Encryption Works




Maybe Mention what a key is

1-Take in Original Image (Plain Text)

2-The Encryption Algorith “jumbles” the original image

3-Uses Substitution and Shuffling to jumble the original image

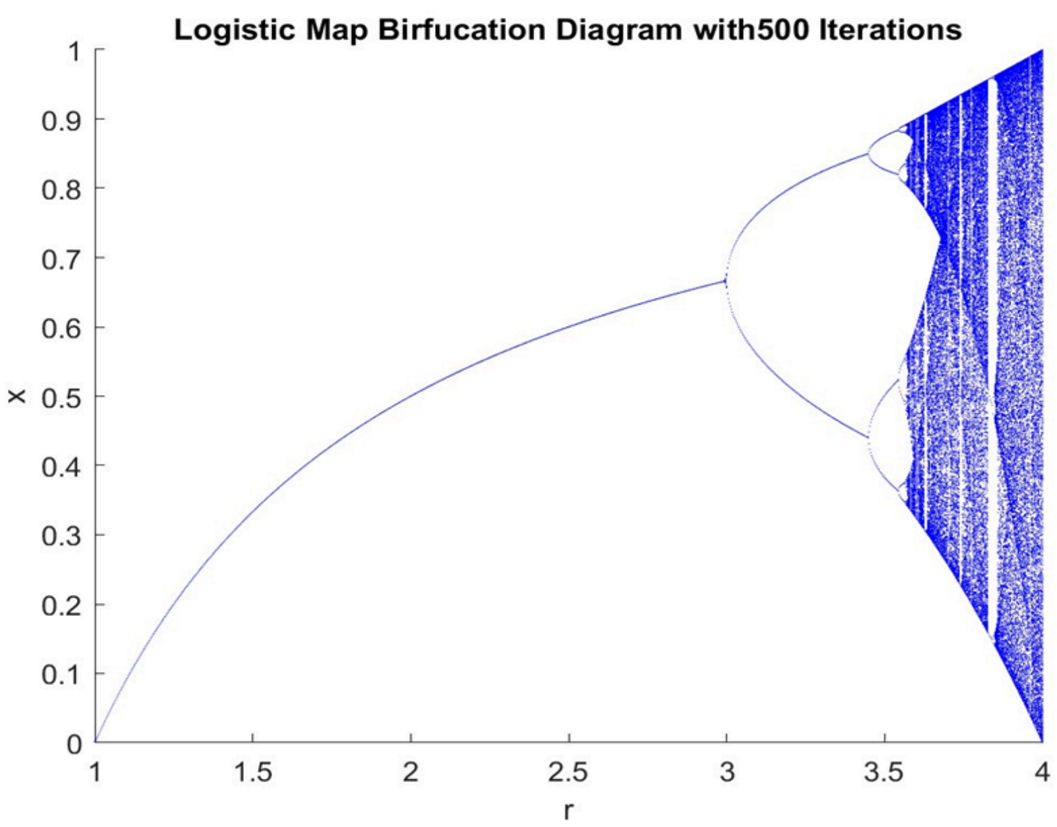
4-A cipher image (jumbled up image is produced).

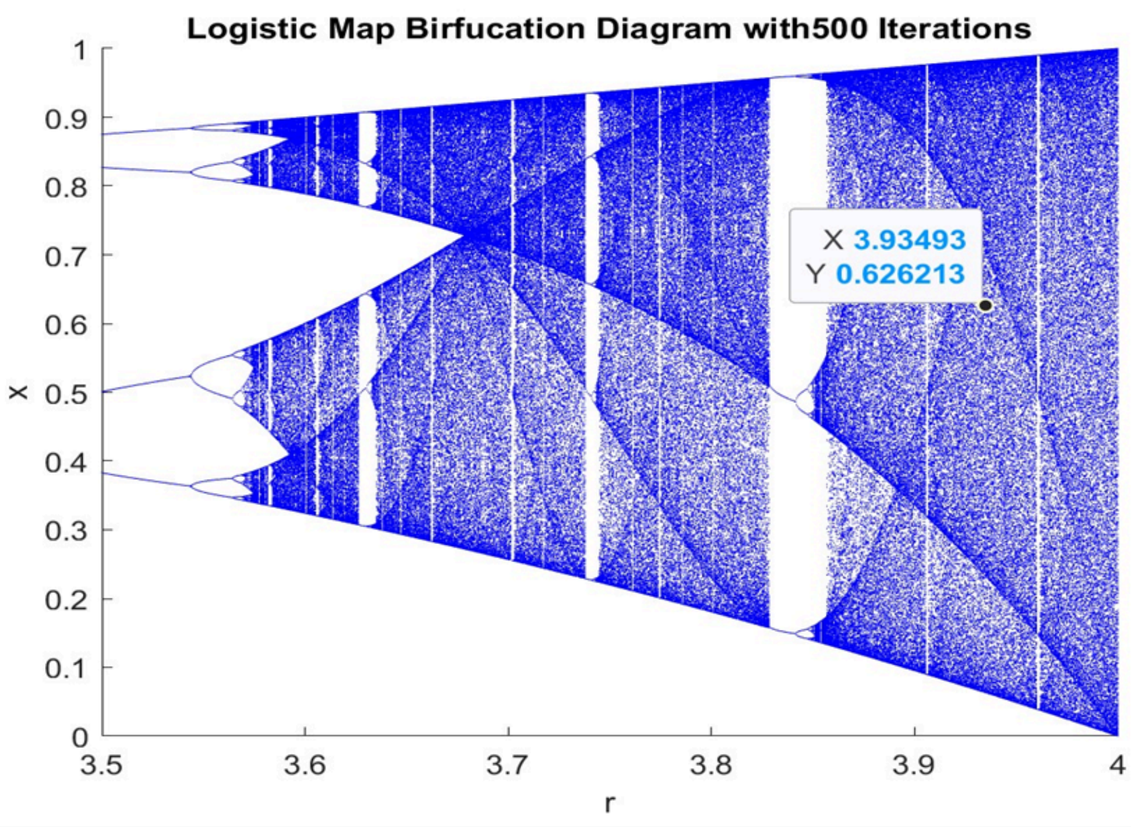
5-The Decryption Algorithm takes the cipher image and the key as inputs and reverses the Encryption Process resulting in the original image

# Basic Encryption Model Using the Logistic Map

Made a basic model in python etcetc

# The Logistic Map



| r = 2 | r = 3.3 | r = 3.83 | r = 3.93 |
|---|---|---|---|
| 0.5 | 0.4788241086051664 | 0.6196668543989299 | 0.9434430207589 16 |
| 0.5 | 0.823520219357992 8 | 0.9026538023601561 | 0.2096980692470 16 |
| 0.5 | 0.479604403299635 5 | 0.3365417761539989 | 0.6512984207742825 |
| 0.5 | 0.82362726479627 9 | 0.8551677966887337 | 0.8925375363338481 |
| 0.5 | 0.4793757984855810 4 | 0.4743678526278855 | 0.3769431304959847 |
| 0.5 | 0.8235963196292456 | 0.9549836632707908 | 0.9229880369907 36 |
| 0.5 | 0.4794418923439434 | 0.1646511873801291 5 | 0.2793488038115 0456 |
| 0.5 | 0.8236053018916865 | 0.5267826959391158 | 0.7911602850089033 |
| 0.5 | 0.4794227083390485 | 0.954752691969232 | 0.6493369555437893 |
| 0.5 | 0.8236026977240704 | 0.1654559584320021 | 0.8948550016760254 |
| 0.5 | 0.479428270284823 | 0.5288474886826693 | 0.3697718436701 0413 |
| 0.5 | 0.8236034529905348 | 0.9543127597793735 | 0.9158496652847409 |
| 0.5 | 0.4794266572015889 | 0.1669876794353168 | 0.3028813896184961 |
| 0.5 | 0.823603233968127 | **0.5327637023686376** | 0.8297969160241709 |
| 0.5 | 0.4794271249857199 | 0.9533886474611906 | 0.5550495971311176 |

# MAIN FUNCTIONS USED

## For the Key

$$x = r^{*}x^{*}(1 - x)$$

$$(x \times 10^{16})\ mod 256$$

## Encrypting the Image

$$\text{encrypt}[i, j] = \text{img}[i, j] \oplus \text{key}[z]$$

$$\oplus = \text{XOR Operator}$$

$$12 = 1100$$

$$4 = 0100$$

$$12 \oplus 4 = 1000 = 8$$

The XOR (Exclusive Or) Operator:

Same inputs =0 i(e.g. 11 and 00 go to 0)

Different inputs =1  i(e.g. 10 and 01 go to 1)

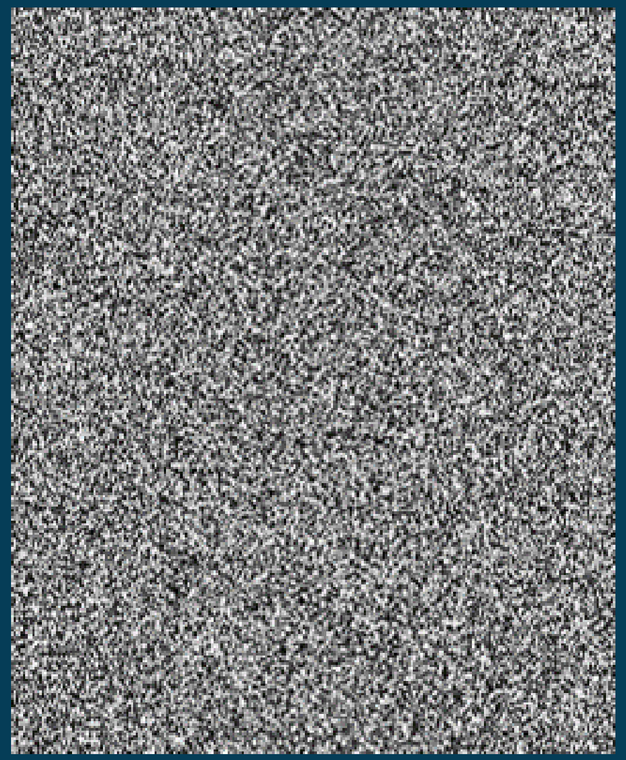
```
key=keygen(0.01,3.9394,height*width)

#Encryption
z=0
enimg=np.zeros(shape=[height,width,4],dtype=np.uint8)
for i in range(height):
    for j in range(width):
        #Pixel Value is XORed with key
        enimg[i,j]=img[i,j]^key[z]
        z+=1
plt.imshow(enimg)
plt.show()
plt.imsave('encryptedImage.bmp',enimg)

#Decryption
z=0
decimg=np.zeros(shape=[height,width,4],dtype=np.uint8)
for i in range(height):
    for j in range(width):
        decimg[i,j]=enimg[i,j]^key[z]
        z+=1
plt.imshow(decimg)
plt.show
plt.imsave('DecryptedImage.bmp',decimg)
```

```
def keygen(x,r,size):
    key=[]
    for i in range(size):
        x=r*x*(1-x)   # logistic map
        key.append(int((x*pow(10,16))%256))
    return key
```

# The Results

## Original Image

## Cipher Image



## Decrypted Image

# More Advanced Encryption Algortithm

A more Advanced Encryption Model using the Logistic Map

# Preliminary 1- Chaotic Window

1-We generate a sequence using the logistic map

2-Arrange the sequence in ascending order

3-Using index from 2 arrange a list of integers from 1-256

4-Arrange the sequence from 3 in a matrix and take 1 from each therefore integers from 0-255

$$X = \{x_1, x_2, x_3, \cdots\cdots\cdots, x_{256}\}$$

$$X_{asc} = \{x_{90}, x_{240}, x_3, \ldots\ldots\ldots, x_{86}\}$$

$$I = \{90,240,3, \ldots\ldots\ldots,86\}$$

$$I = \begin{pmatrix} 89 & 239 & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & \cdots & 85 \end{pmatrix}$$

# Preliminary 2- Pixel Shuffling

1-Convert Grey Scale Image into a vector of size M*N

2-We Iterate the LM p+M*N times. Have a sequence of size M*N

3-Again arrange in ascending order

4-Arrange a list of integers using 3's index’s

5-Use 4 to reorder 1

$$P_{MN} = \{p_1, p_2, \cdots\cdots\cdots, p_{mn}\}$$

$$S_{MN} = \{s_1, s_2, s_3, \cdots\cdots\cdots, s_{mn}\}$$

$$S_{asc} = \{s_{80}, s_8, s_{154}, \cdots\cdots\cdots, s_5\}$$

$$I_{sh} = \{80,8,154, \cdots\cdots\cdots, 5\}$$

$$P_{MN} = \{p_{80}, p_8, p_{154} \cdots\cdots\cdots, p_5\}$$

# Preliminary 3 - Generating Xs

1-Using Preliminary 1 we build a chaotic window

2-We XOR the chaotic window with the Image matrix

3-We get the sum of the XORed Pixels and divide by largest possible number

4-We now Iterate this number p times in the LM and the resulting X is labelled Xs

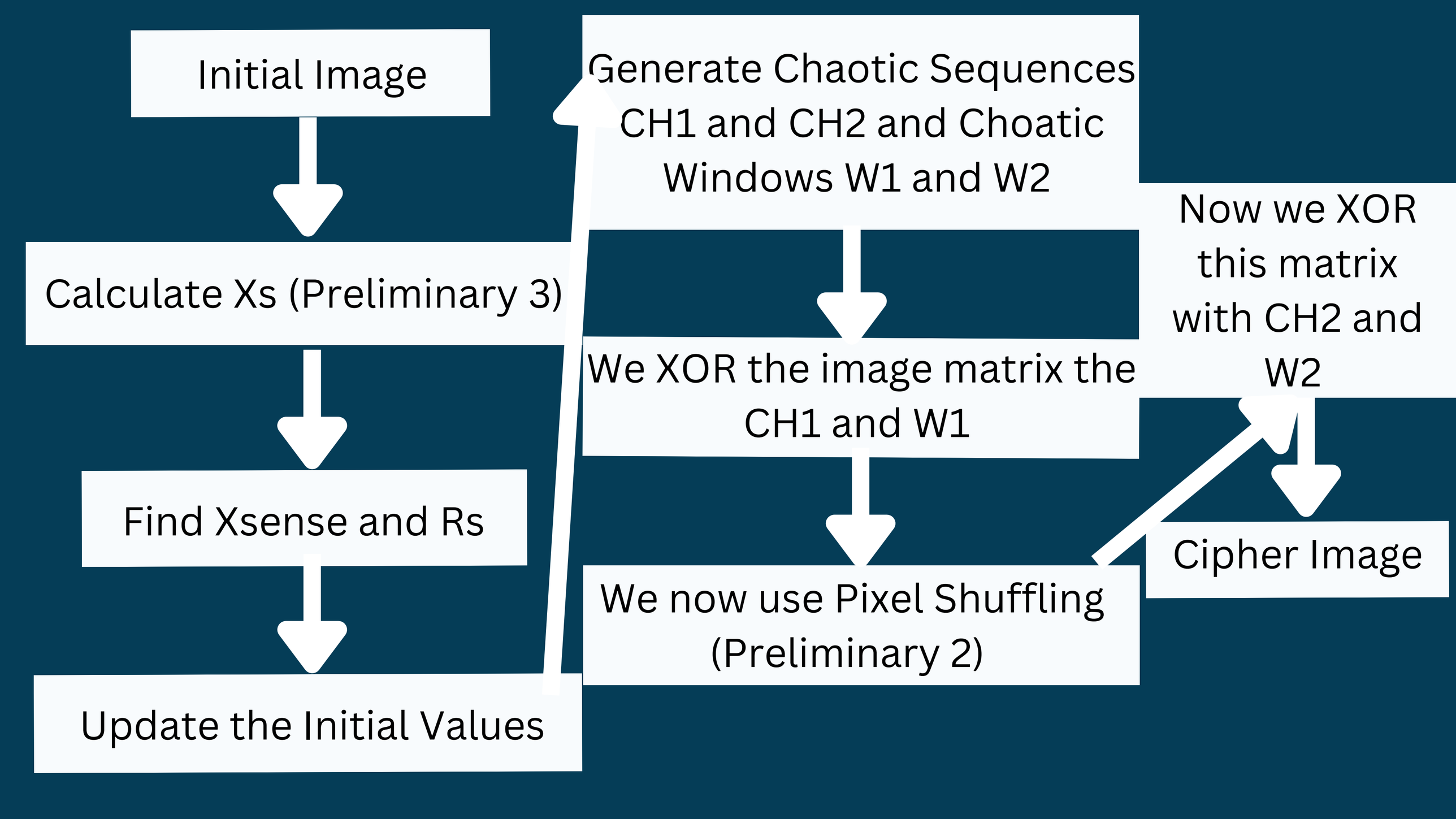
$$I = \begin{pmatrix} 89 & 239 & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & \cdots & 85 \end{pmatrix}$$

$$I_{new} = I \oplus img$$

$$I_{newsum} = \frac{\sum I_{new}}{M \times N \times 255}$$

$$x = r * I_{newsum} * (1 - I_{newsum})$$

$$x_s = x \; after \; p \; iterates$$


Initial Image
Calculate Xs (Preliminary 3)
Find Xsense and Rs
Update the Initial Values
Generate Chaotic Sequences CH1 and CH2 and Choatic Windows W1 and W2
We XOR the image matrix the CH1 and W1
We now use Pixel Shuffling (Preliminary 2)
Now we XOR this matrix with CH2 and W2
Cipher Image

# The Key contains

Initial Values $\{x_{01}, x_{02}, x_{03}, x_{04}, x_{05}\}$

Initial Parameters $\{\mu_1, \mu_2, \mu_3, \mu_4, \mu_5\}$

## Generate Chaotic Sequences CH1 and CH2 and Choatic Windows W1 and W2

CH1 and CH2 calculated from first two initial parameters. W1 and W2 calculated from next two initial parameters

$$CH_i = (floor(CH_i(j) \times 10^{14}))\ mod 256$$

## Find Xsense and Rs

We get the mean of the initial values and Xs. Then iterate this p times and get Xsense. Rs is the mean of the inital parameters

## Update the Initial Values

$$x_{0i} = (x_{0i} + x_{sense}) mod 1$$

$$\mu_i = \frac{(\mu_i + \mu_s)}{2}$$

We XOR the image matrix the CH1 and W1

$$C_b(i_b, j_b) = P_b(i_b, j_b) \oplus W1 \oplus CH_1(n_b)$$

Now we XOR this matrix with CH2 and W2

$$C_{encrypted}(i_b, j_b) = C_p(i_b, j_b) \oplus W_2 \oplus CH_2(n_b)$$

# Security

## Differential Attack:

The attacker aims to observe how the system behaves differently by making slight changes in the plaintext or the key and seeing what the result is

### NPCR Number of Pixels Change Rate

$$\frac{\sum_{i=i}^{M}\sum_{j=1}^{N} D(i,j)}{M \times N} \times 100\%$$

$$D(i,j) = \begin{cases} 1 & if\ C_1 \neq C_2 \\ 0 & if\ C_1 = C_2 \end{cases}$$

**Basic Model:** 0.7614%

**Advanced LM:** 99.6092%

**AES:** 99.61%

### UACI Unified Average Changing Intensity

$$\frac{\sum_{i=i}^{N}\sum_{j=1}^{M}|c_1(i,j) - c_2(i,j)|}{255 \times M \times N} \times 100\%$$

**Basic Model:** 0.1267%

**Advanced LM:** 33.4656%

**AES:** 33.43%

# Correlation Coefficient

Measures the relationship between two variables. Find the correlation coefficient of two adjacent pixels.

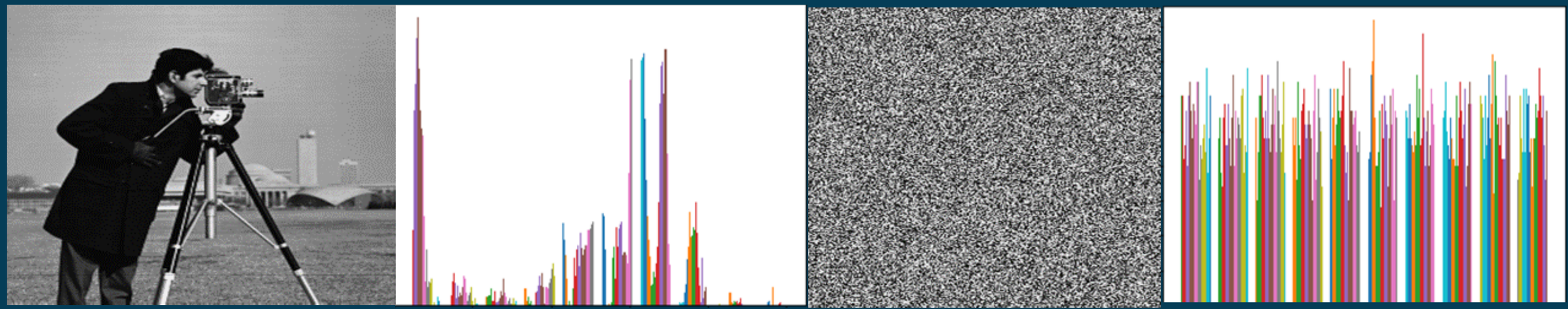
## Correlation Coefficient of Encrypted Image

| Basic Model | Advanced LM | AES |
| --- | --- | --- |
| r=0.771 | r=0.02608 | r=-0.0182 |

# Histogram Analysis

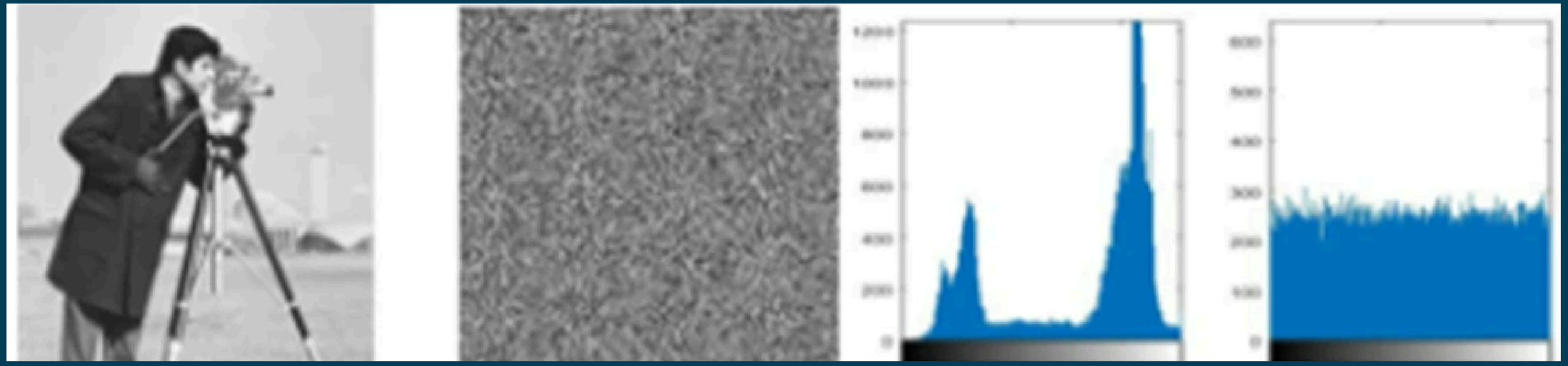
We plot the intensity of the pixels in the image before and after encryption occurs on ahistogram. We want a homogenous histogram after encryption.A homogeneous histogram is one in which the distribution of pixel values is comparatively uniform or consistent throughout the image

## Basic Model

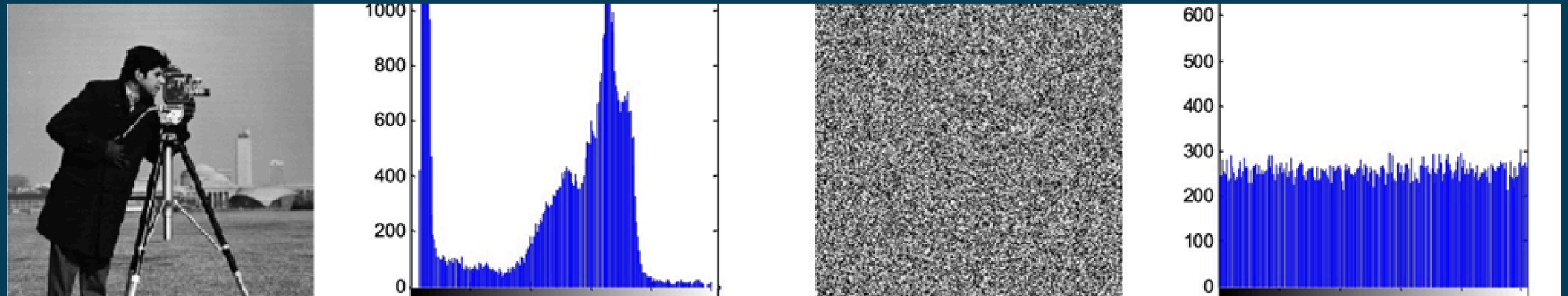


# Histogram Analysis

## AES



## Advanced LM



# Conclusion

Chaos is very useful for Encryption in my opinion since:

1-It’s properties almost mimic the properties of a good encryption model

2-Compared to AES the security metrics of the Advanced Logistic Map are very similar