# Chaos

## Encryption using chaos

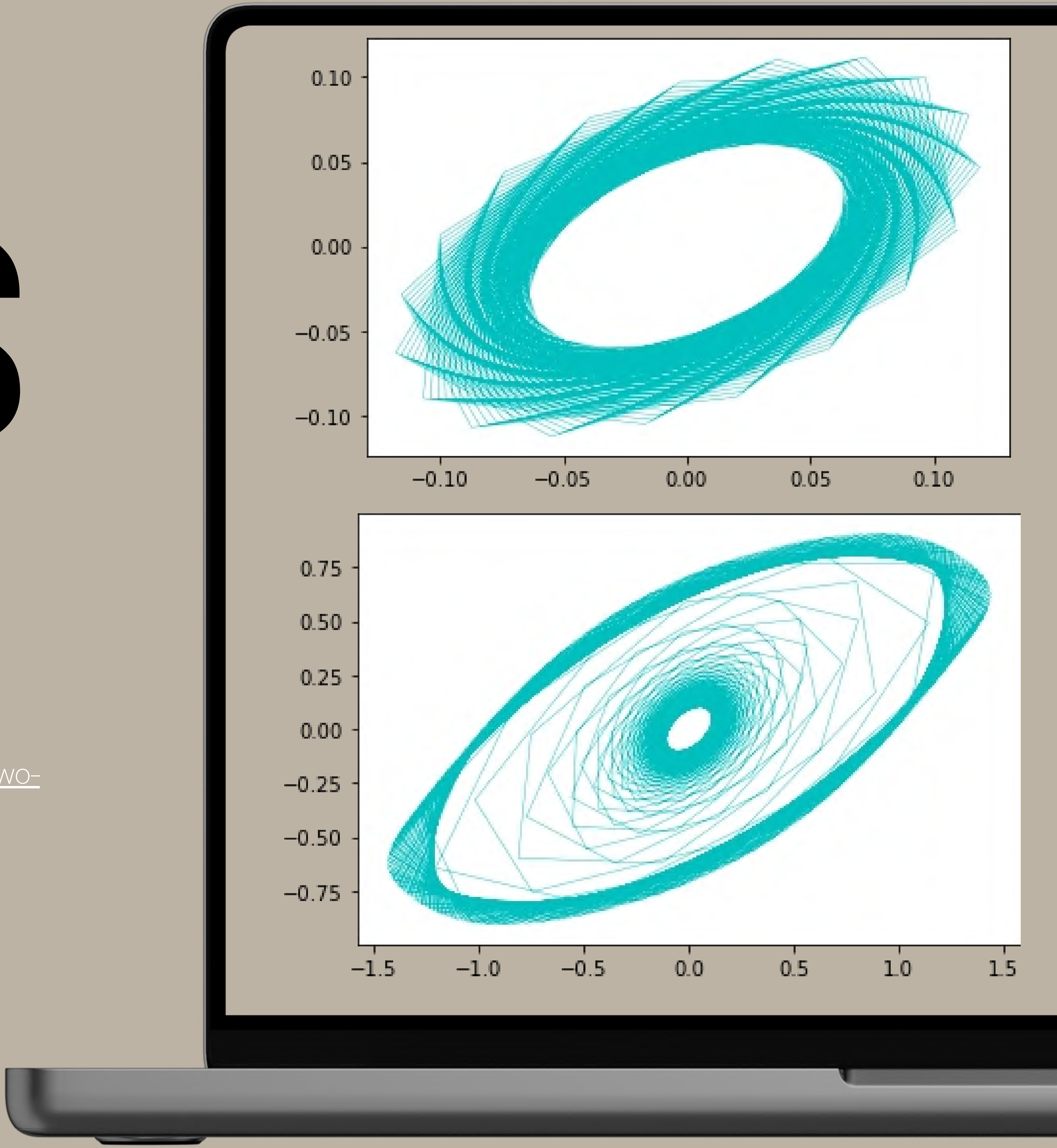
**Reference(s):**

- https://www.researchgate.net/publication/258660674_Image_encryption_using_the_two-dimensional_logistic_chaotic_map
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7712162/

Abhishek, Lokesh, Niare,  Rhishabh, Shadab

22 November 2022

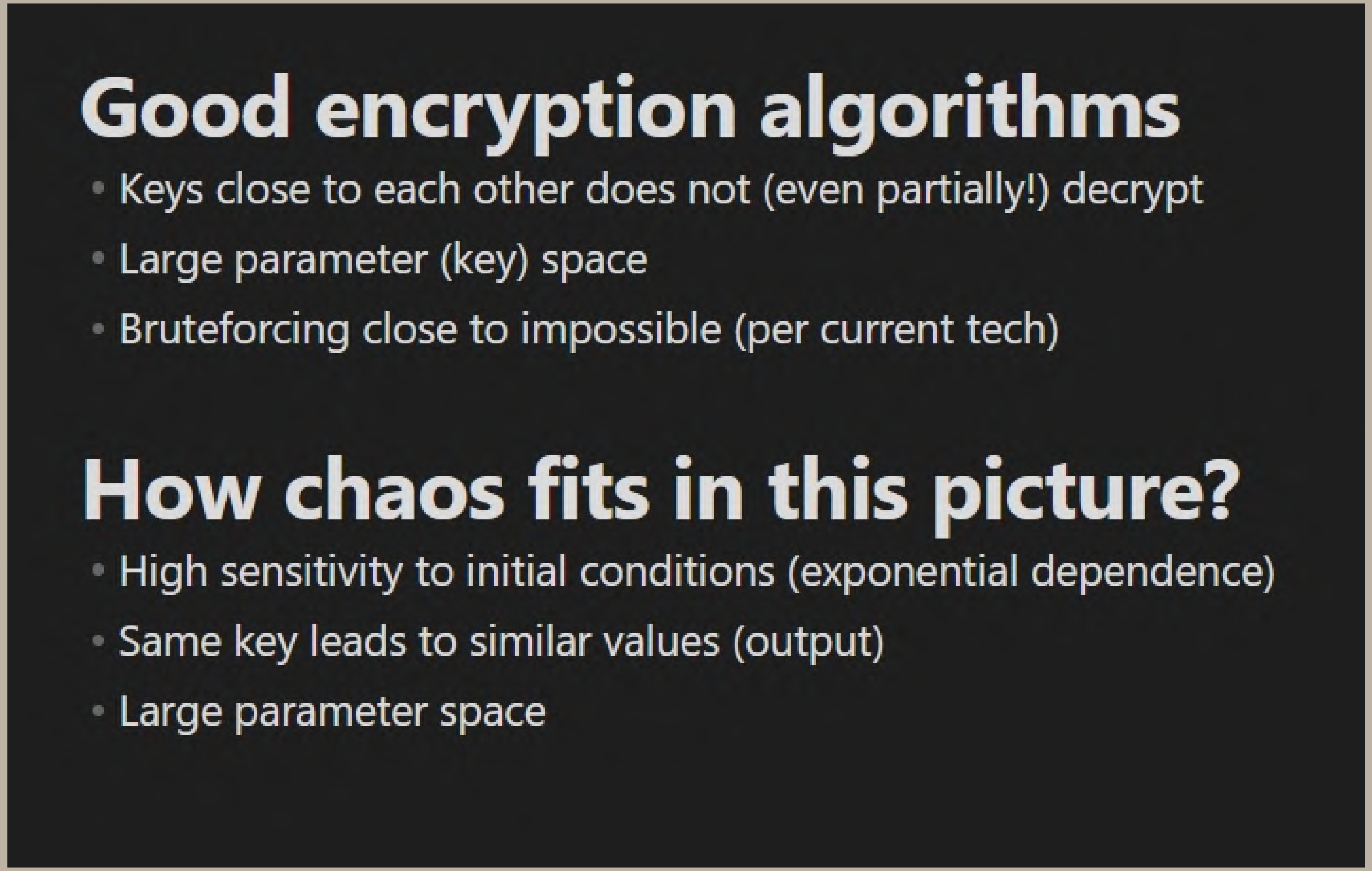
PH567- Non Linear Dynamics course project



Agenda

Pitch

# Introduction



**Good encryption algorithms**

- Keys close to each other does not (even partially!) decrypt
- Large parameter (key) space
- Bruteforcing close to impossible (per current tech)

**How chaos fits in this picture?**

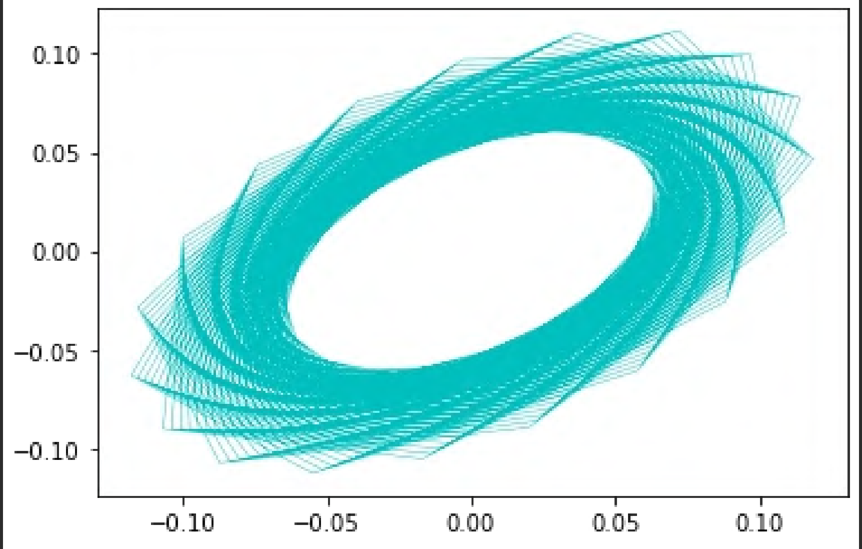
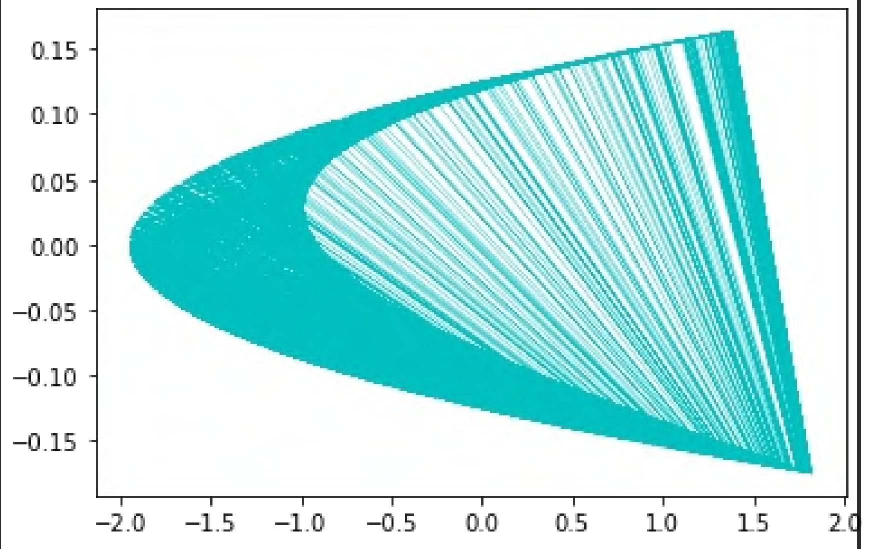
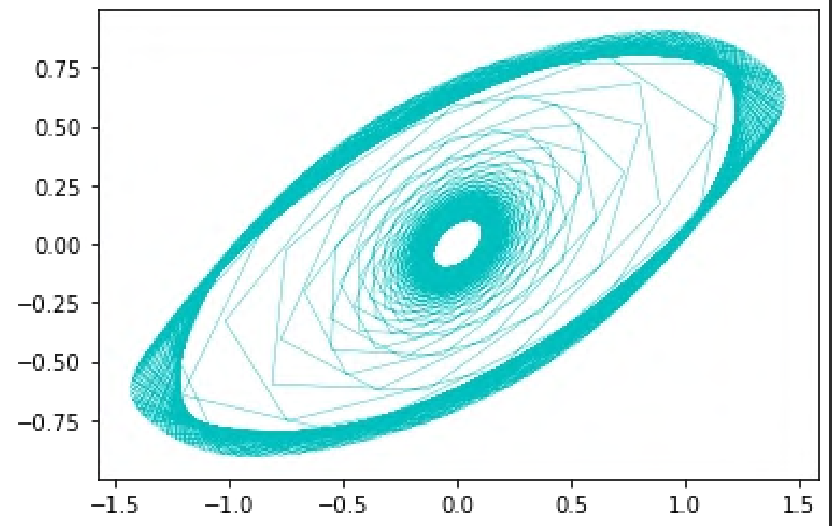
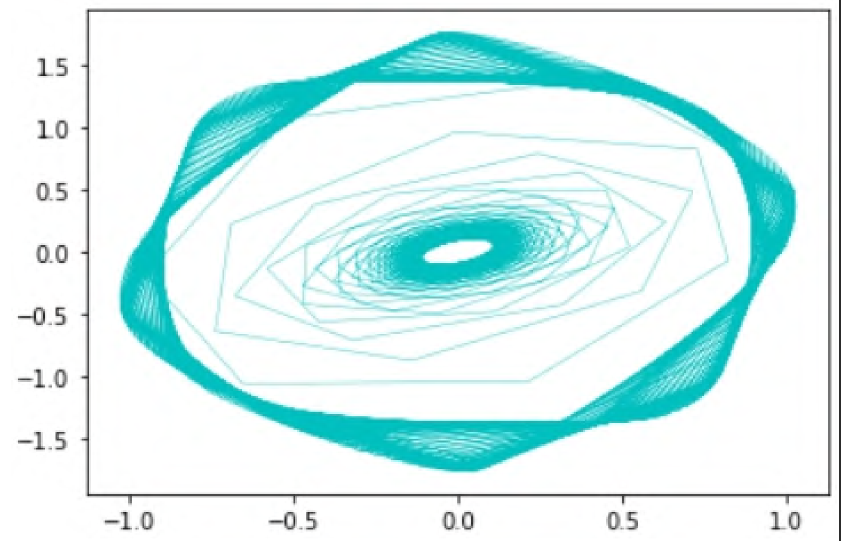
- High sensitivity to initial conditions (exponential dependence)
- Same key leads to similar values (output)
- Large parameter space

As the world progresses into the era of big data,  issues like privacy and secure data transfer become more important now than ever before. With the advent of technologies capable of bruteforcing traditional encryption algos, there is a need for a better encryption algorithms.

## Main idea

- Use chaotic maps to generate chaotic sequences which are used to encrypt text, audio, images and video.
- Chaotic sequences are generated using some parameters (keys). These maps tend to have a range of parameters for which the system shows chaotic behavior. This range of real numbers is the parameter space. Since the space is dense in the reals, and due to exponential sensitivity 2 keys close to each other will not generate the same chaotic sequence. However, if we have the exact key, we get the original object back by "filtering" out the chaotic mask (sequence).
- We then use permutation and diffusion methods for encryption (basically interchanging x and y columns)

# Chaotic maps used

| Finance map | Logistic map | The eye | Galaxy map |
|---|---|---|---|
|  |  |  |  |
| $y_{n+1} = y_n - \alpha \tan(x_n)$ <br> $x_{n+1} = \sin(x_n) + y_{n+1}$ | $y_{n+1} = b^2 x_n$ <br> $x_{n+1} = x_n^2 + y_n^2 - a^2$ | $y_{n+1} = \tan(y_n) - \alpha \sin(x_n)$ <br> $x_{n+1} = \sin(x_n) + \tan(y_{n+1})$ | $y_{n+1} = \sin(y_n) - \alpha \tan(x_n)$ <br> $x_{n+1} = \tan(x_n) + \sin(y_{n+1})$ |
| Parameters: 1 | Parameters: 2 | Parameters: 1 | Parameters: 1 |

# Text Encryption

The text to be encrypted is passed as a string of characters. The string is converted to a list.

- Python's ord( ) function is used to convert the ith character to its unicode which is a number.

- Chaotic keys are generated using the chaotic map

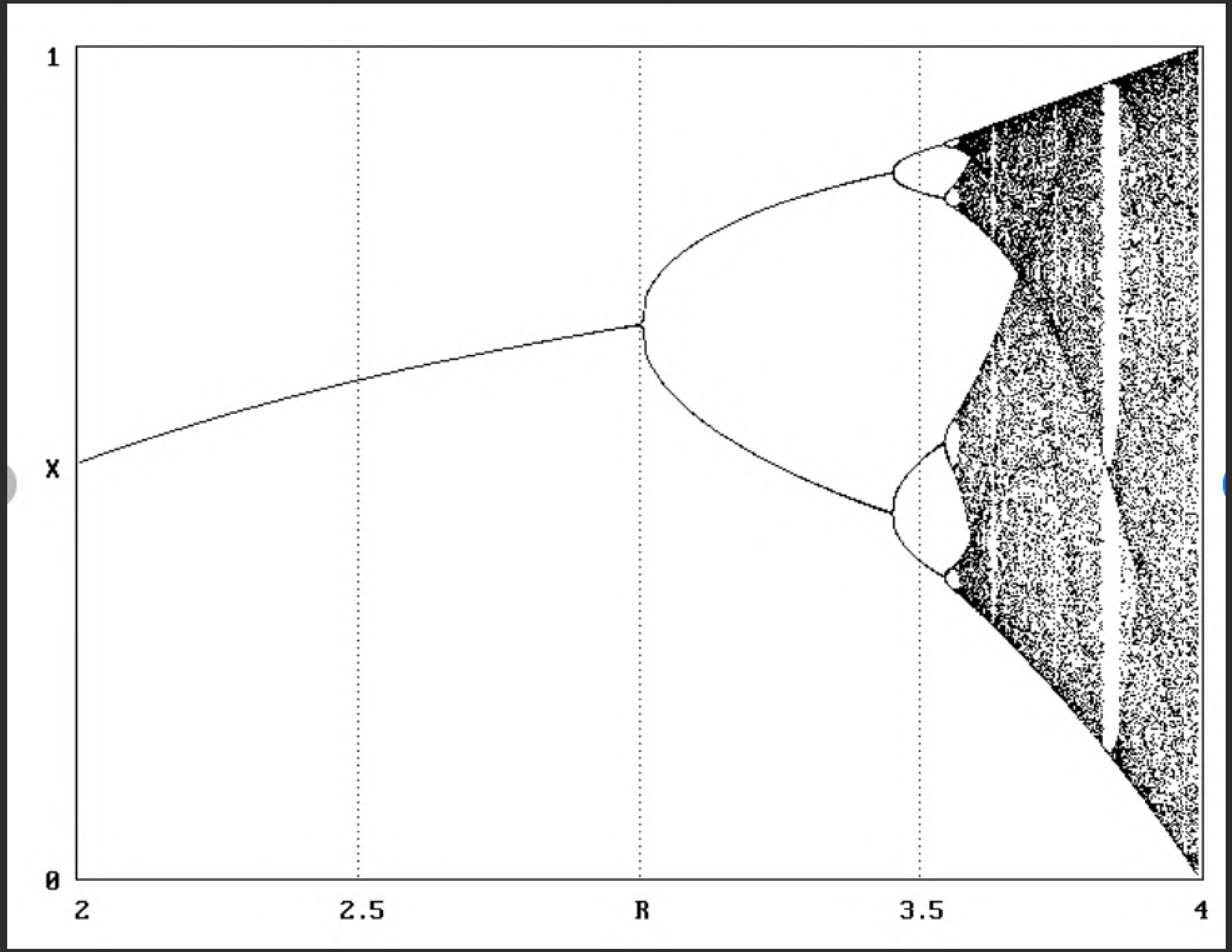
$$x_{n+1} = rx_n(1 - x_n)$$

- The encryption is done by doing bitwise XOR operation between the unicode and for the ith character and the keys generated by the above map.

- For decrypting the text the bitwise XOR operation is performed again which gives back the original list of unicodes and hence the text.

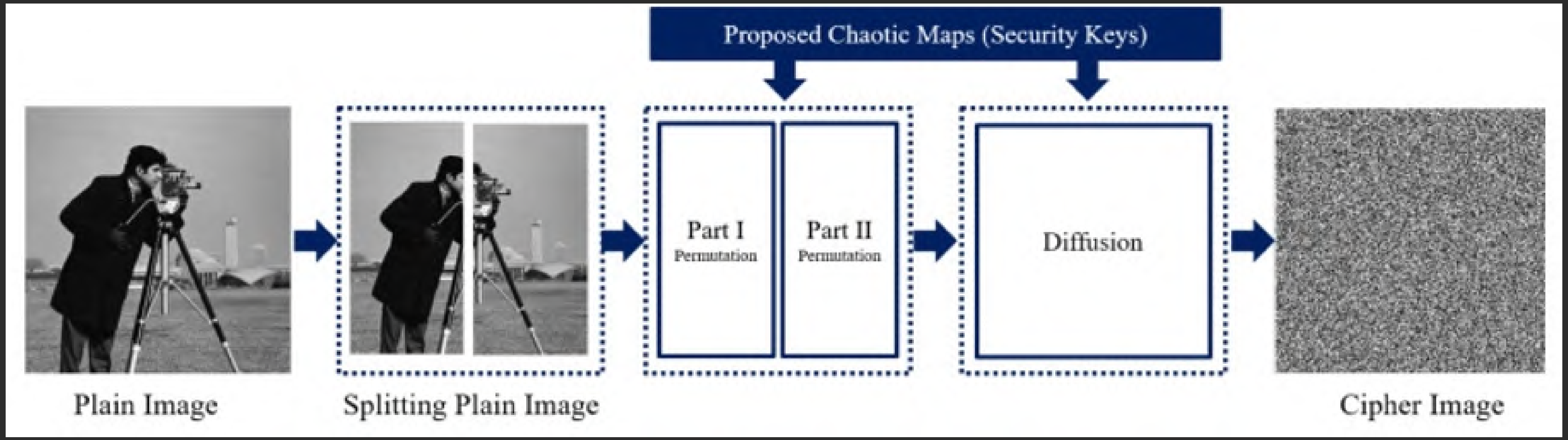
Pitch

## Bifurcation diagram for logistic map

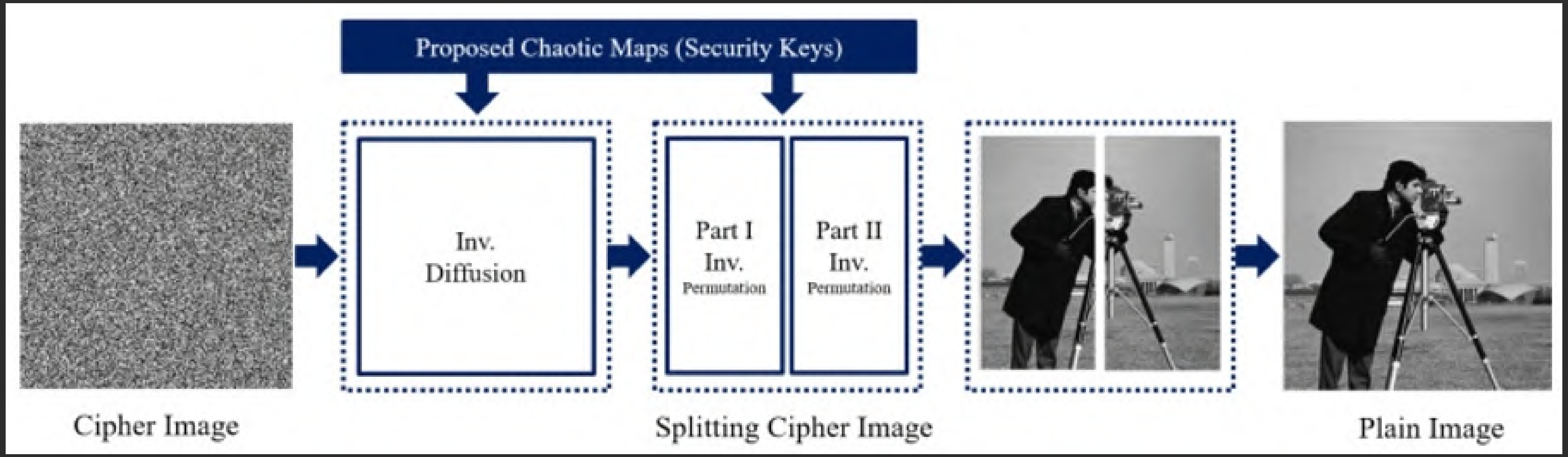


# Image encryption

## Encryption



## Decryption



## Permutation

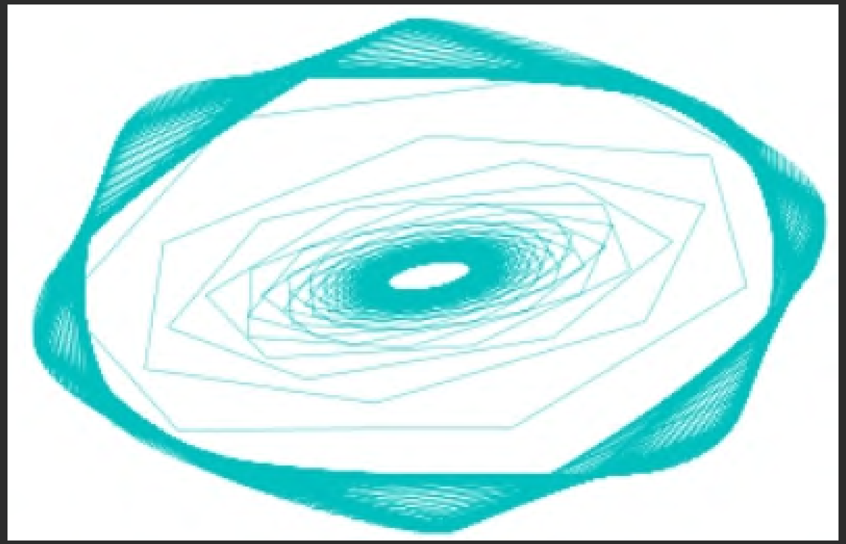
- Keys generated through chaotic maps are labeled with indices
  - [(0 , key0), (1 , key1), (2 , key2), … ]
- Keys are then sorted forming a chaotic sequence of indices
  - [ (143, key143), (3, key3), (69, key69), …]
  - key143 < key3 < key69<…
- The resulting indices are then used as the new new position for the column or row of pixels

- Each half of the image is first permuted along columns then rows

## Maps used

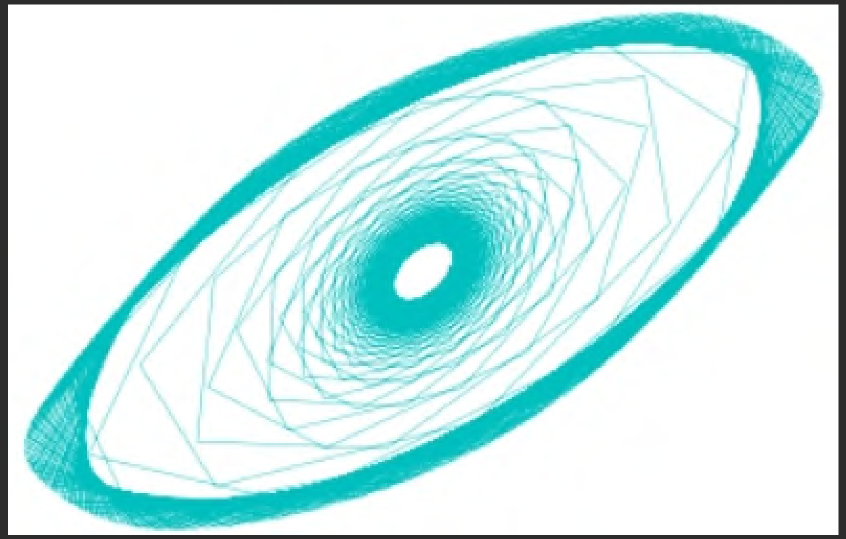
Galaxy Map for 1st half

$$y_{n+1} = \sin(y_n) - \alpha \tan(x_n)$$
$$x_{n+1} = \tan(x_n) + \sin(y_{n+1})$$



Eye map for the 2nd half

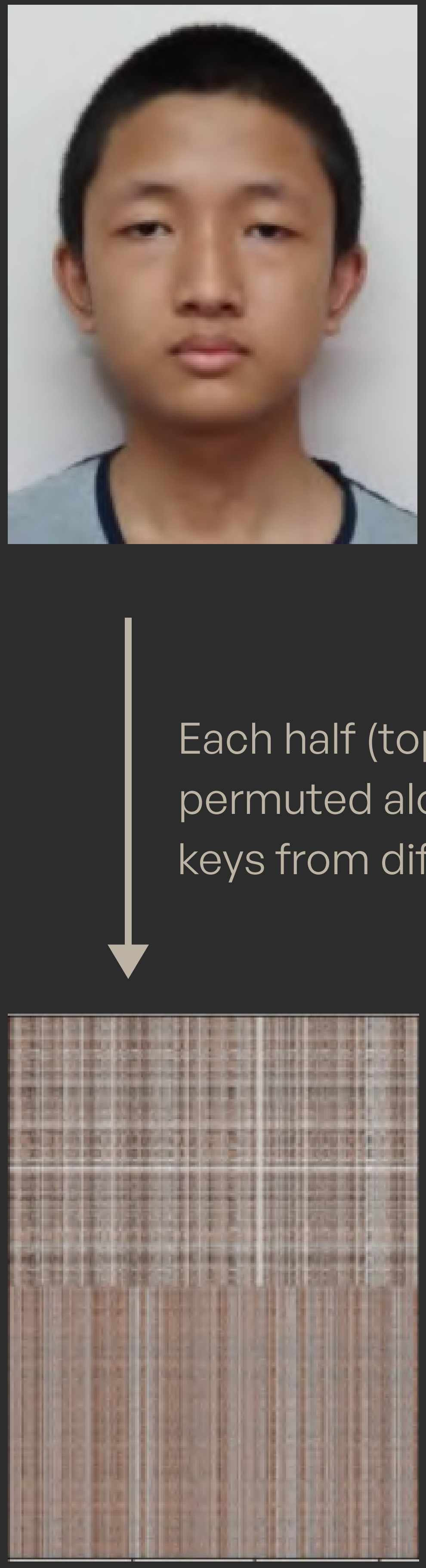
$$y_{n+1} = \tan(y_n) - \alpha \sin(x_n)$$
$$x_{n+1} = \sin(x_n) + \tan(y_{n+1})$$



## Permutation

- Keys generated through chaotic maps are labeled with indices
  - [(0 , key0), (1 , key1), (2 , key2), … ]
- Keys are then sorted forming a chaotic sequence of indices
  - [ (143, key143), (3, key3), (69, key69), …]
  - key143 < key3 < key69<…
- The resulting indices are then used as the new new position for the column or row of pixels

- Each half of the image is first permuted along columns then rows

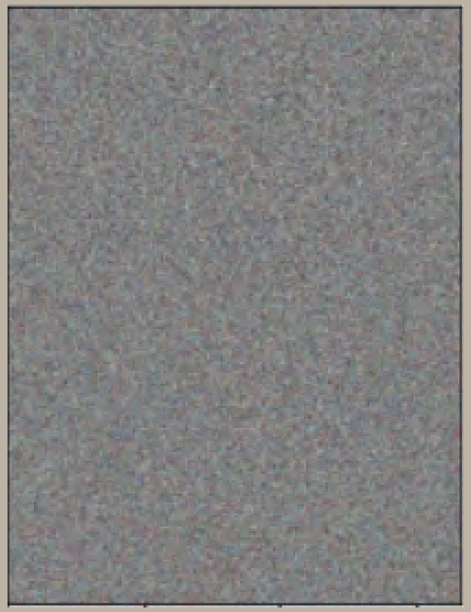

Each half (top and bottom half) permuted along X and Y axis using keys from different chaotic maps

Pitch

## Diffusion

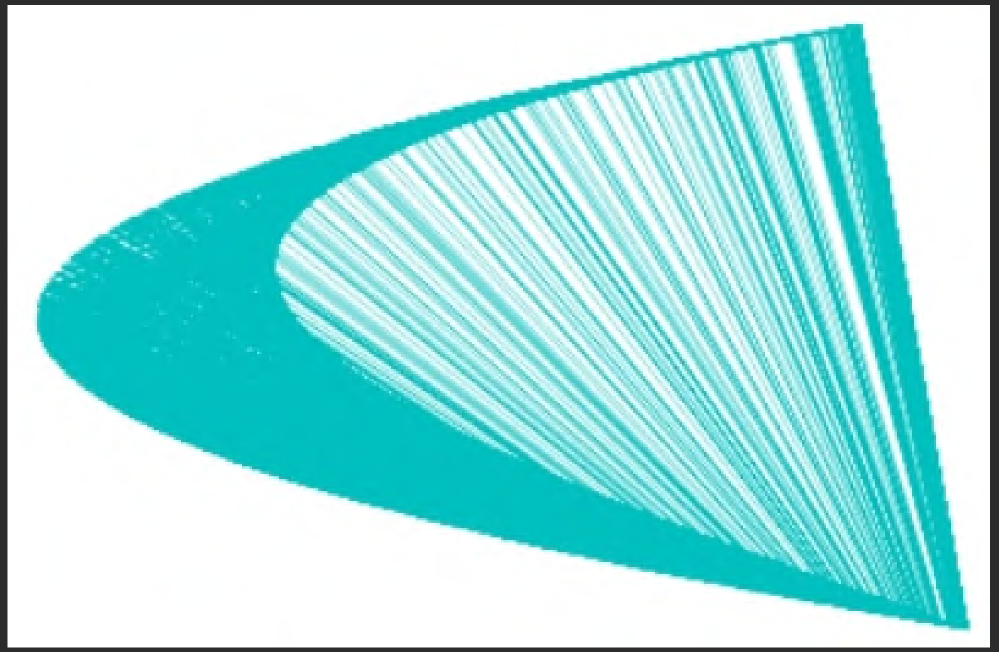
- We generate another set of keys using a different chaotic map
- The resulting keys are then XORed with each pixel value of the image

$$\xrightarrow{\text{Xor with keys}}$$



## Map used

$$y_{n+1} = b^2 x_n$$
$$x_{n+1} = x_n^2 + y_n^2 - a^2$$



Modified logistic map

Pitch

# Basic encryption Vs 2D permutation encrytion

Basic encryption using just diffusion

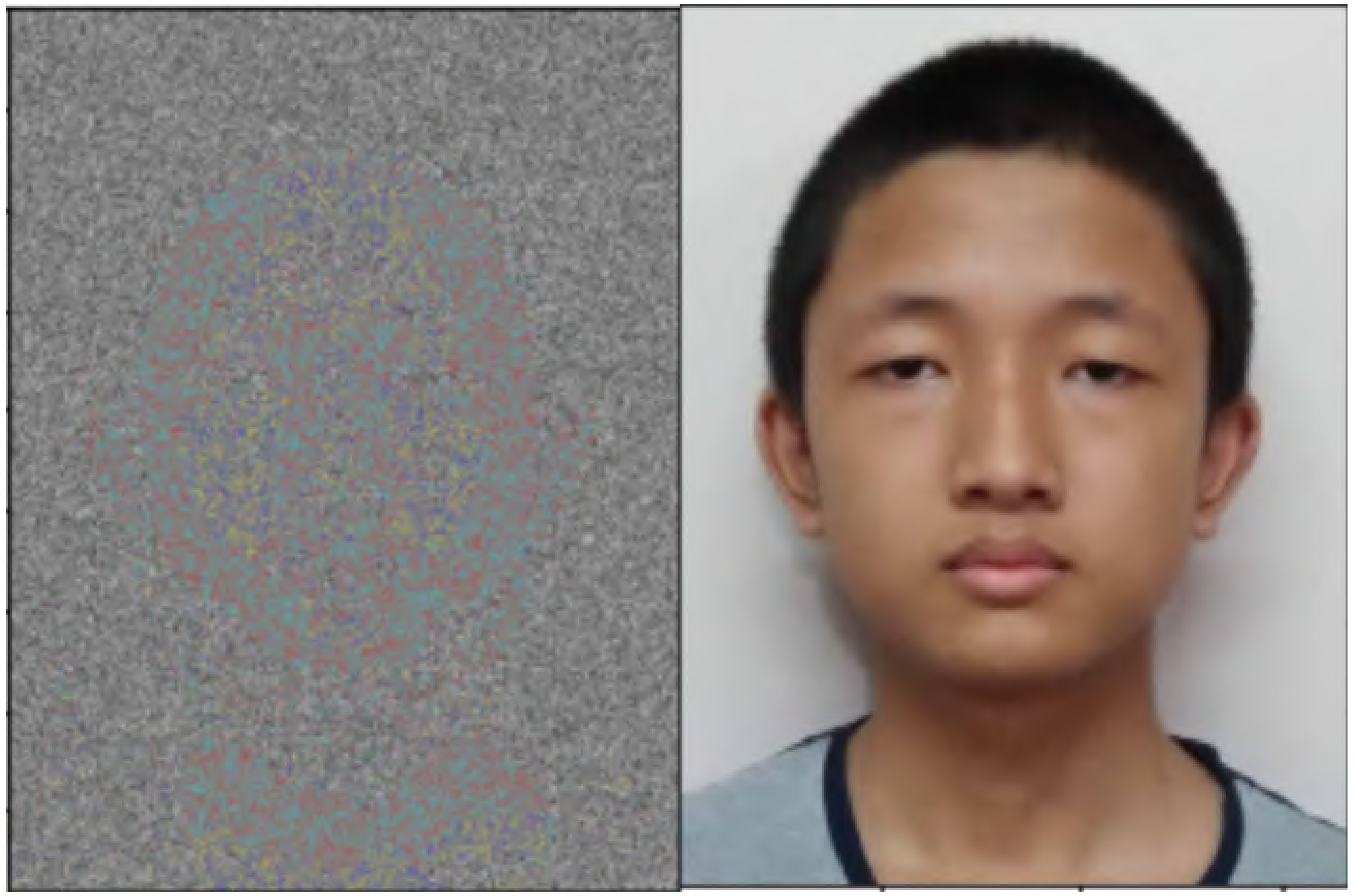
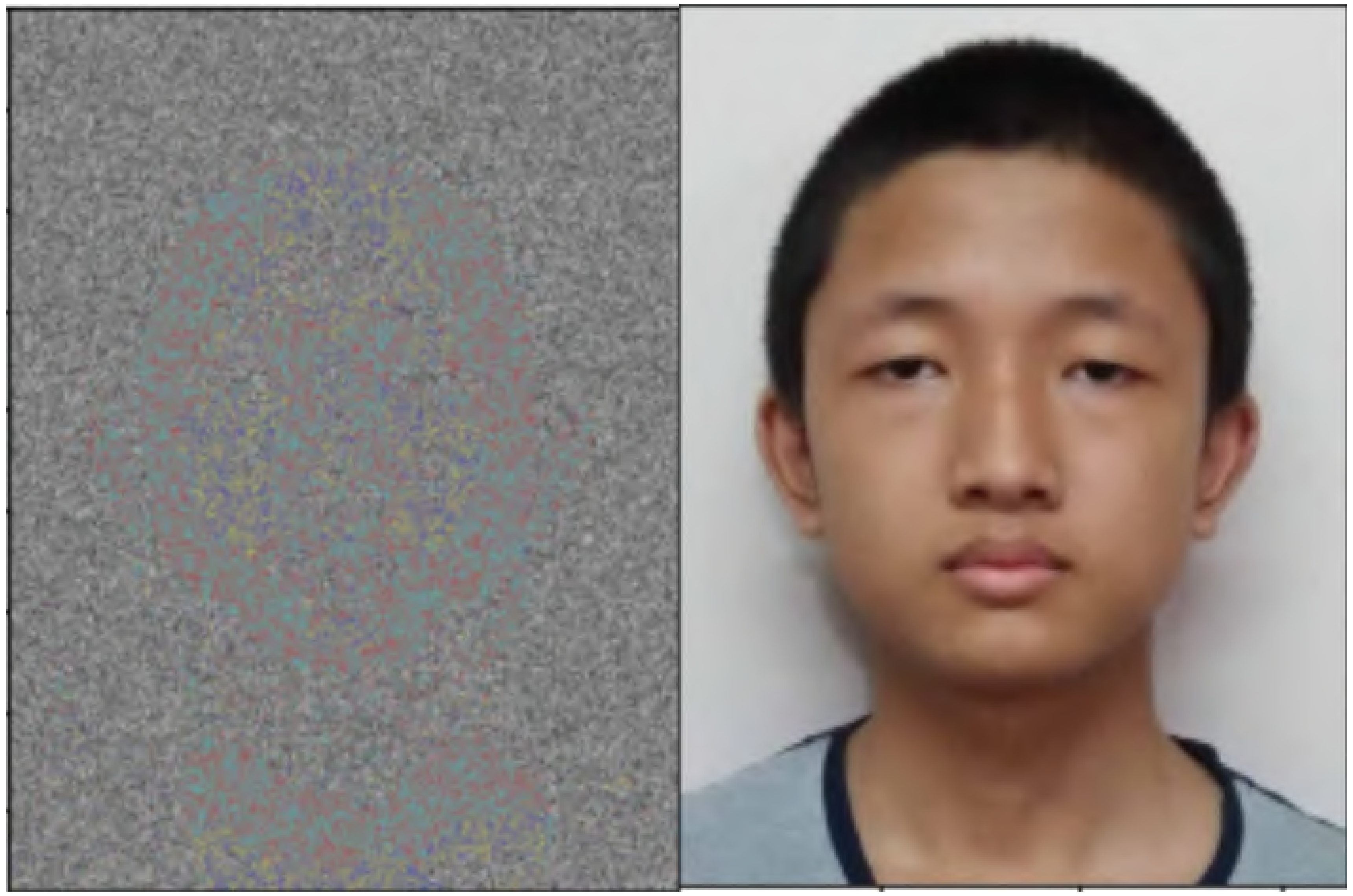


Fig. 3 - After image of the face is visible!!

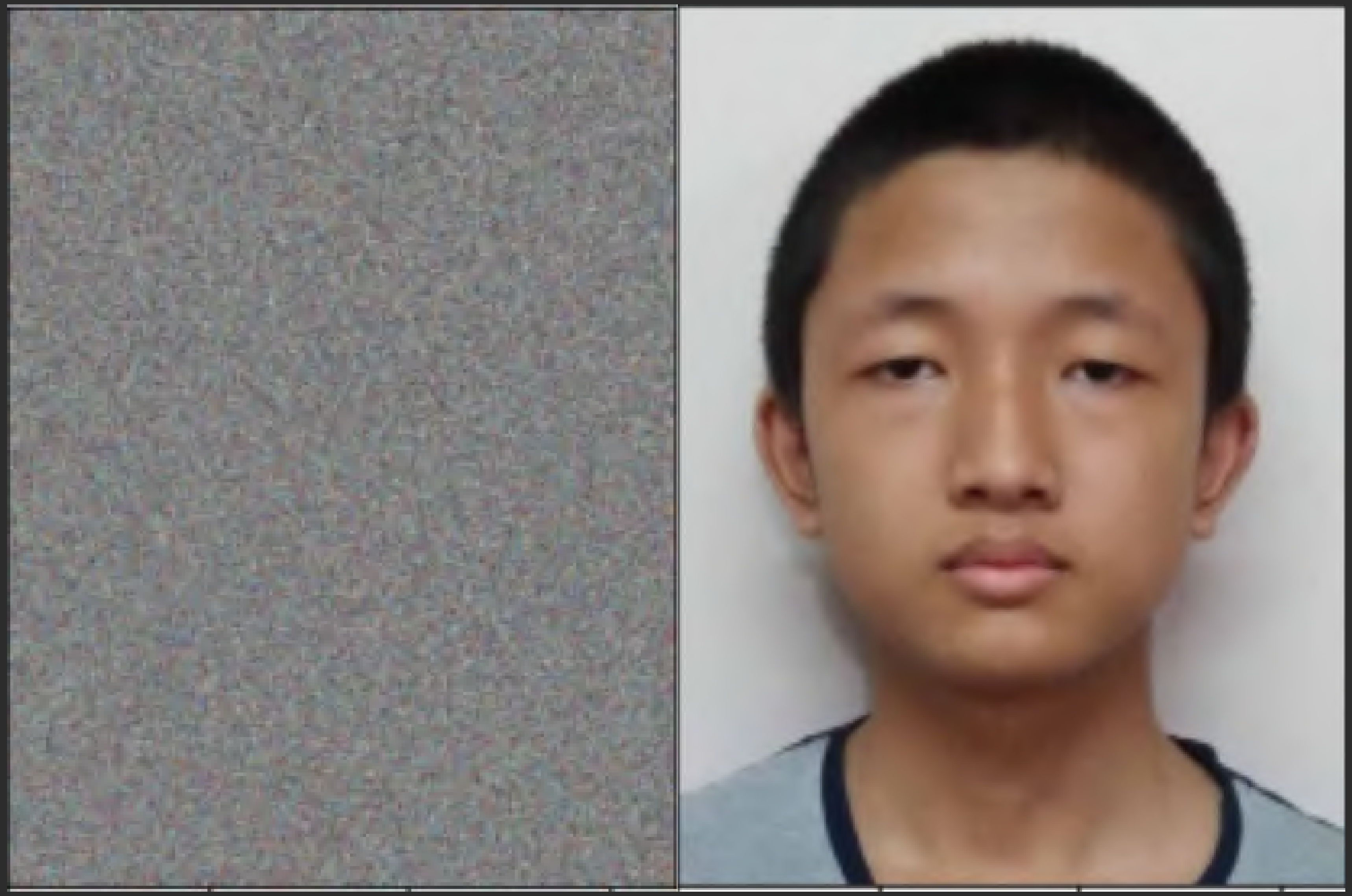
# Basic encryption Vs 2D permutation encrytion

Basic encryption using just diffusion



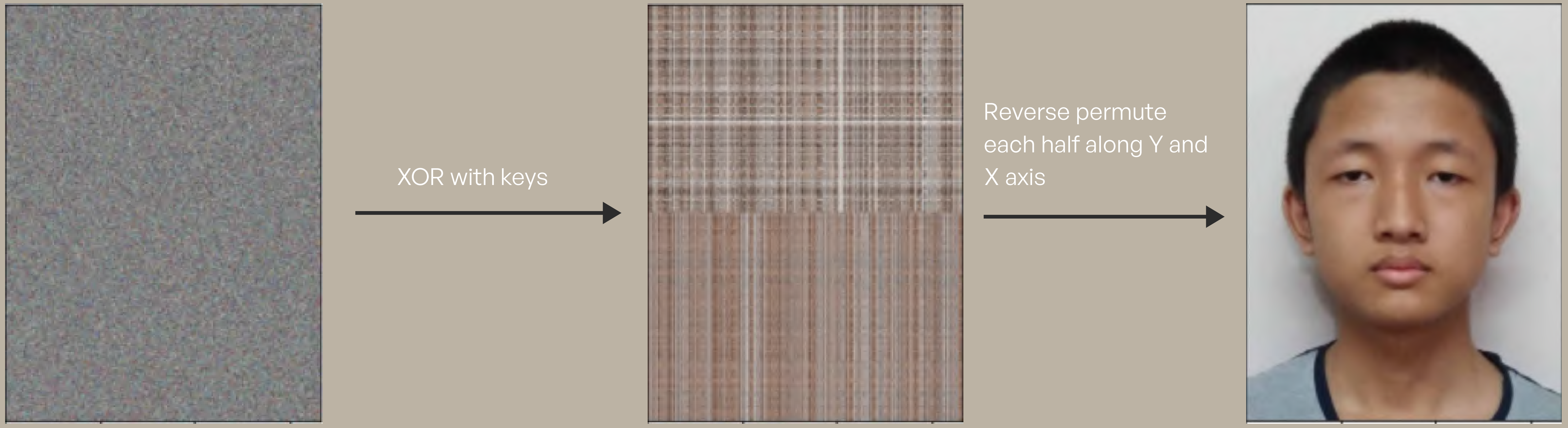
Afterimage of the face is visible!!

Permutation diffusion using 2d chaotic map



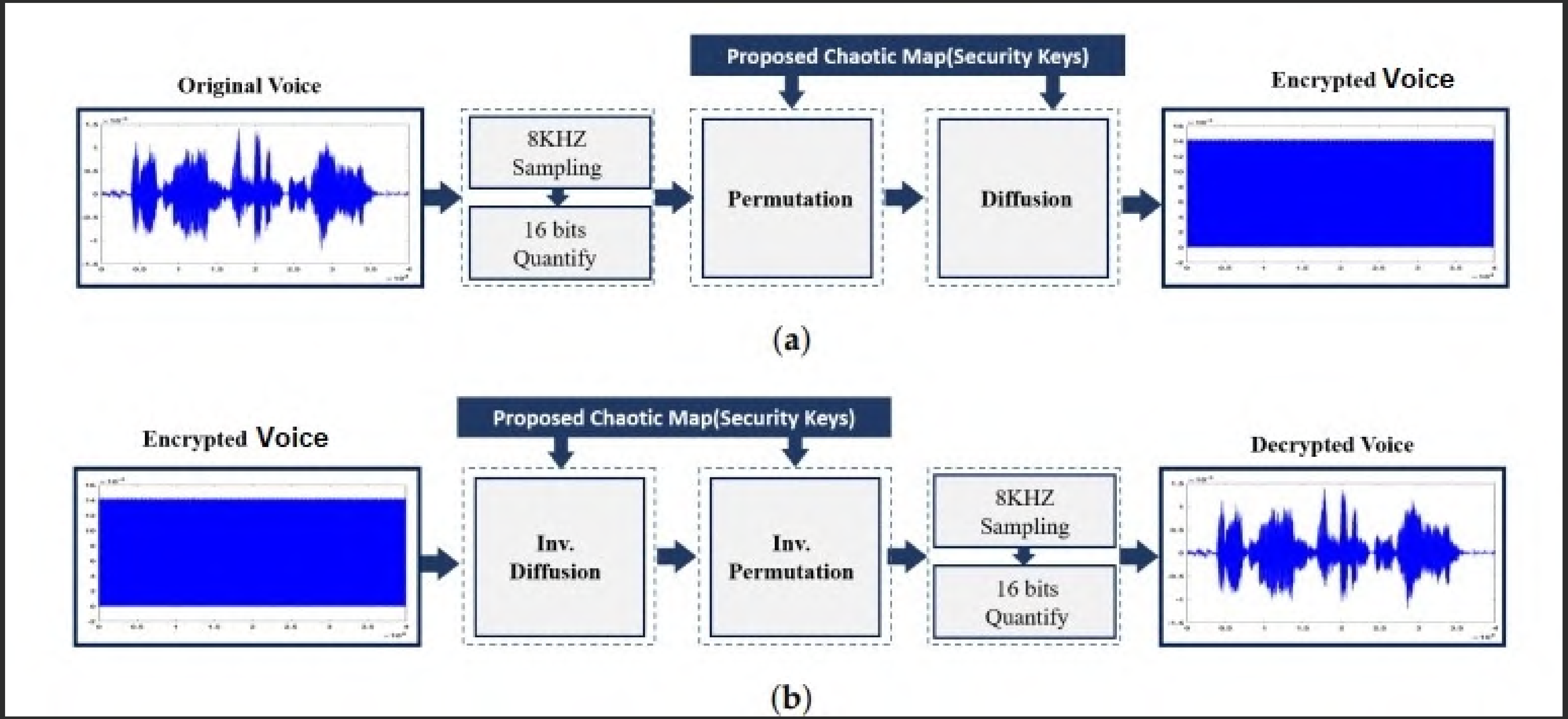
Complete randomness to a human eye !

# Image Decryption



# Audio encryption and decryption

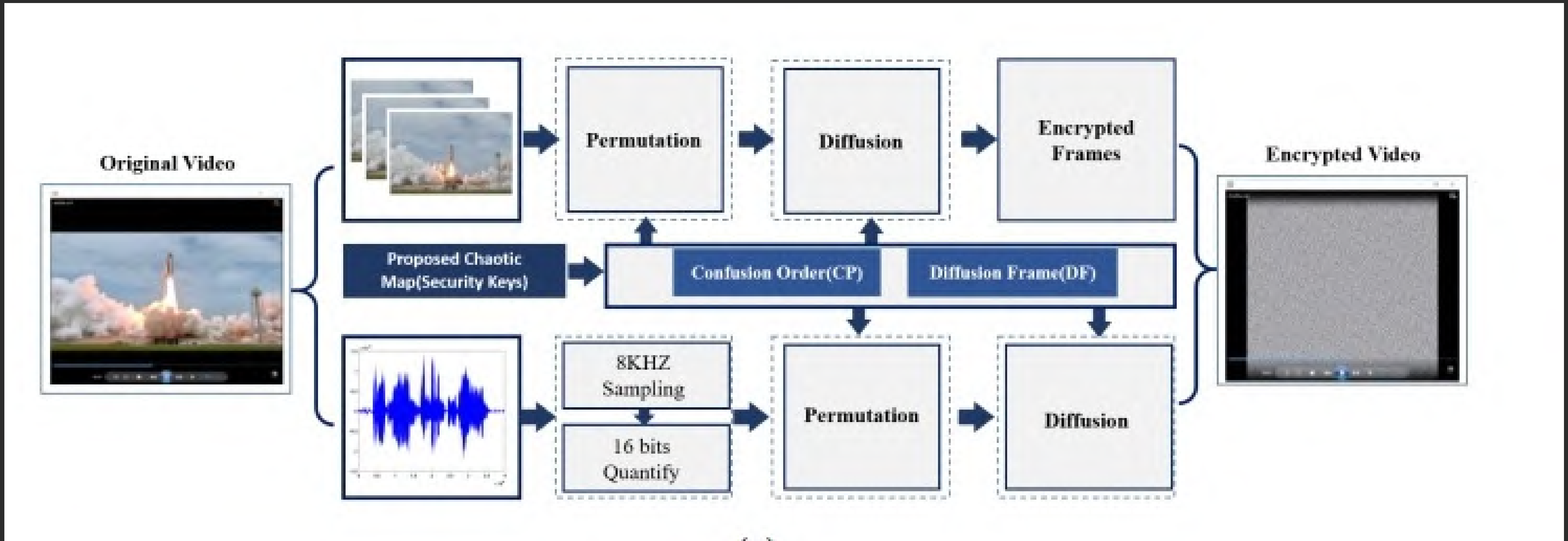
The structure of the proposed scheme for audio encryption is schematized in below figure and consists of two operations: permutation and masking of the speech signal using the proposed maps. The encrypted speech signal will then be sent to the receiver over a channel, which will be decrypted to recover the original speech signal according to the chaotic map. Decryption is done by inverting the permutation and diffusion (using the chaotic sequence) to obtain back the original audio (similar to image case). The signal is converted to a numpy array using scipy.io.



Pitch

## Video Encryption

The steps of the proposed technique for video encryption is schematized in the below picture. In general, video files consists of a sequence of image frames each of which is represented as a 2D array of pixels. Video files exhibit high correlation not only for the adjacent pixels in each frame but also between successive frames. Therefore, the basis of a good cryptosystem is to devise effectual key generation procedures to decorrelate adjacent image locations. The analysis starts with separating a given video file into two parts: a sequence of image frames using python CV and voice using moviepy library. This image sequence is processed using the proposed image encryption as we have used earlier.

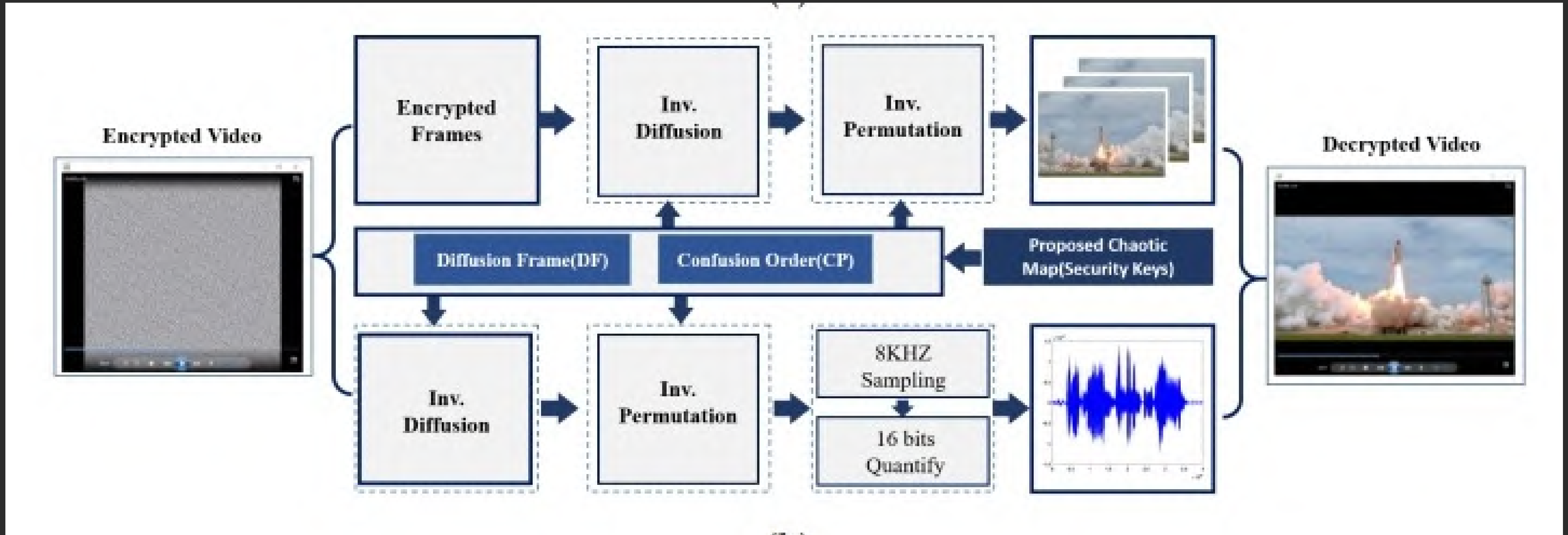


(a)

Pitch

## Video Decryption

On the other hand, the voice files are processed using the proposed voice encryption method which also we have used earlier. Then, both ciphered image and voice data are combined using python CV libraries to produce the cipher color video frame.
For decryption process we just reverse the encryption process as first extracting the encrypted image frames and audio and the processing the decryption of image frames and audio. And then we merge these to decrypted file to create one decrypted video which will be same as the original one.



(b)

Pitch

## Performance Analysis

The better the encryption model is if it is more resistant to attacks and the speed of encryption and decryption is fast.

Following are the important factors to analyse the algorithm :

- NPCR (Number of Pixel Change Rate)
- Correlation of adjacent pixels
- Key space size
- Speed performance

# NPCR

The NPCR(R,G,B) is used to measure the number of pixels in difference of a color component in two images.

$$\mathrm{NPCR}_{\mathrm{R,G,B}} = \frac{\sum_{i,j} D_{\mathrm{R,G,B}}(i,j)}{N} \times 100\%$$

where N is the total number of pixels in the image and D(R,G,B(i, j)) is defined as

$$D_{\mathrm{R,G,B}}(i,j) = \begin{cases} 0 & C_{\mathrm{R,G,B}}(i,j) = C'_{\mathrm{R,G,B}}(i,j) \\ 1 & C_{\mathrm{R,G,B}}(i,j) \neq C'_{\mathrm{R,G,B}}(i,j) \end{cases}$$

NPCR(R,G,B) = 99.609375% for image in Fig. 3 encryption.

# Correlation of adjacent pixels

For an ordinary image, each pixel is usually highly correlated with its adjacent pixels either in horizontal, vertical or diagonal directions.

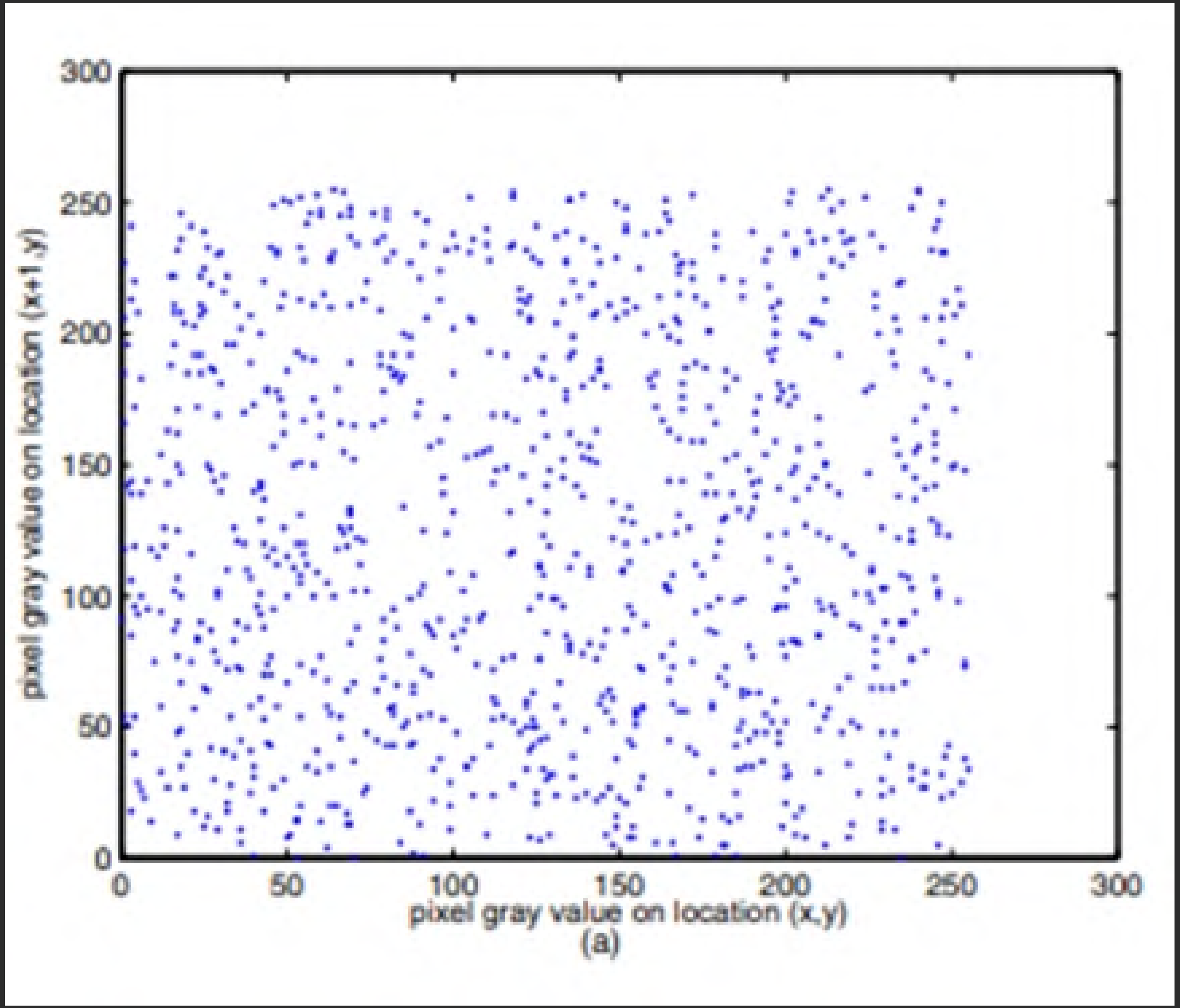
$$cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))(y_i - E(y)),$$

$$r_{xy} = \frac{cov(x,y)}{\sqrt{D(x)}\sqrt{D(y)}}$$

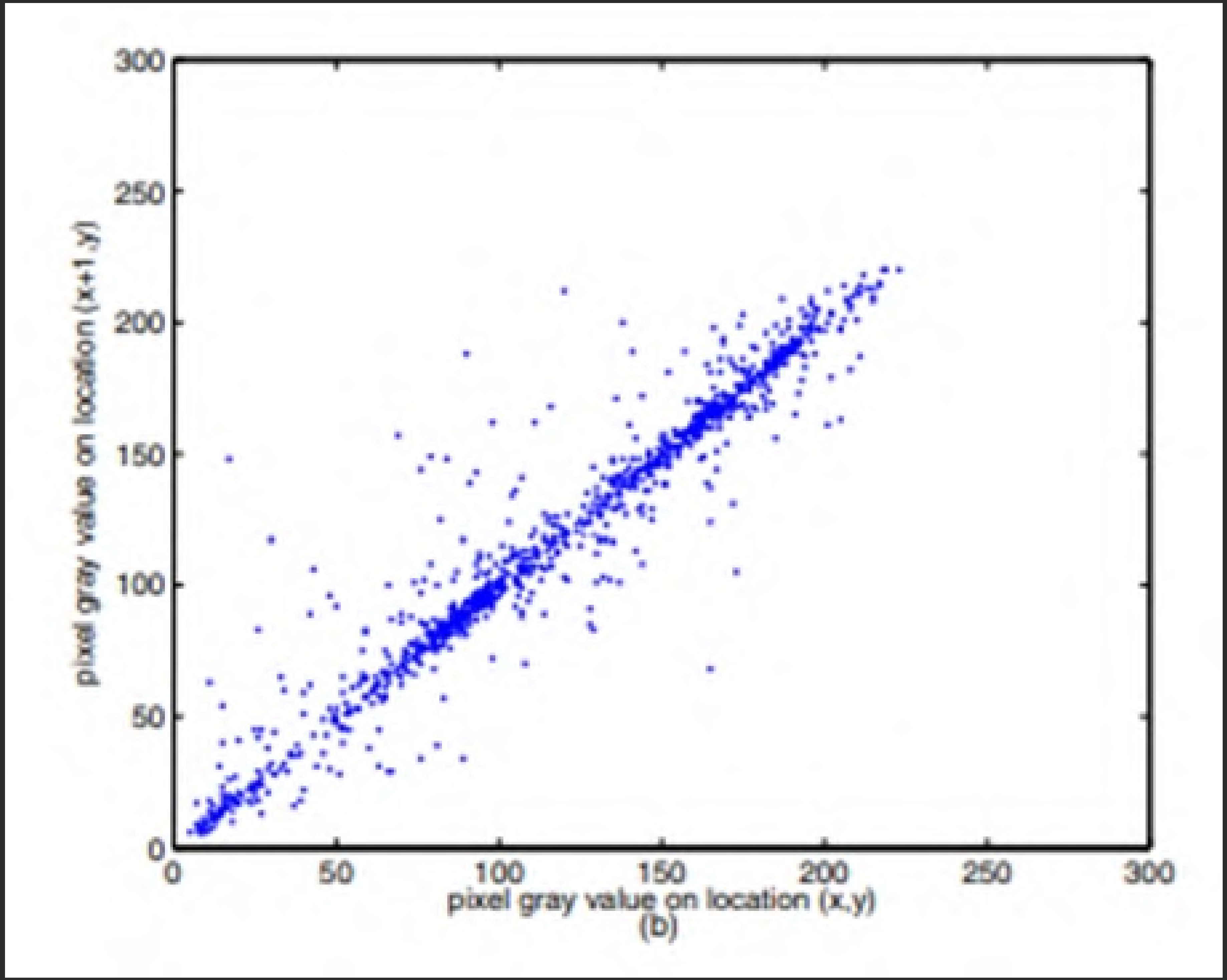
$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i,$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2,$$

For Plain image, cov(x,y) = 0.9686
(In horizontal direction)
For encrypted image, cov(x,y) = 0.00094
(In horizontal direction)



Correlation of Ciphered Image



Correlation of Plain Image

## Key space size :

The number of bits in a secret key used for encryption. It defines the complexity of the algorithm by putting an upper bound on it. Here our algorithm encrypted/decrypted an 8 × 8 bytes block primarily based on permutation and substitution the byte. For 192 bits, key space size = $2^{192}$ which is about 6.277 × $10^{57}$ and is sufficient to resist brute force attack

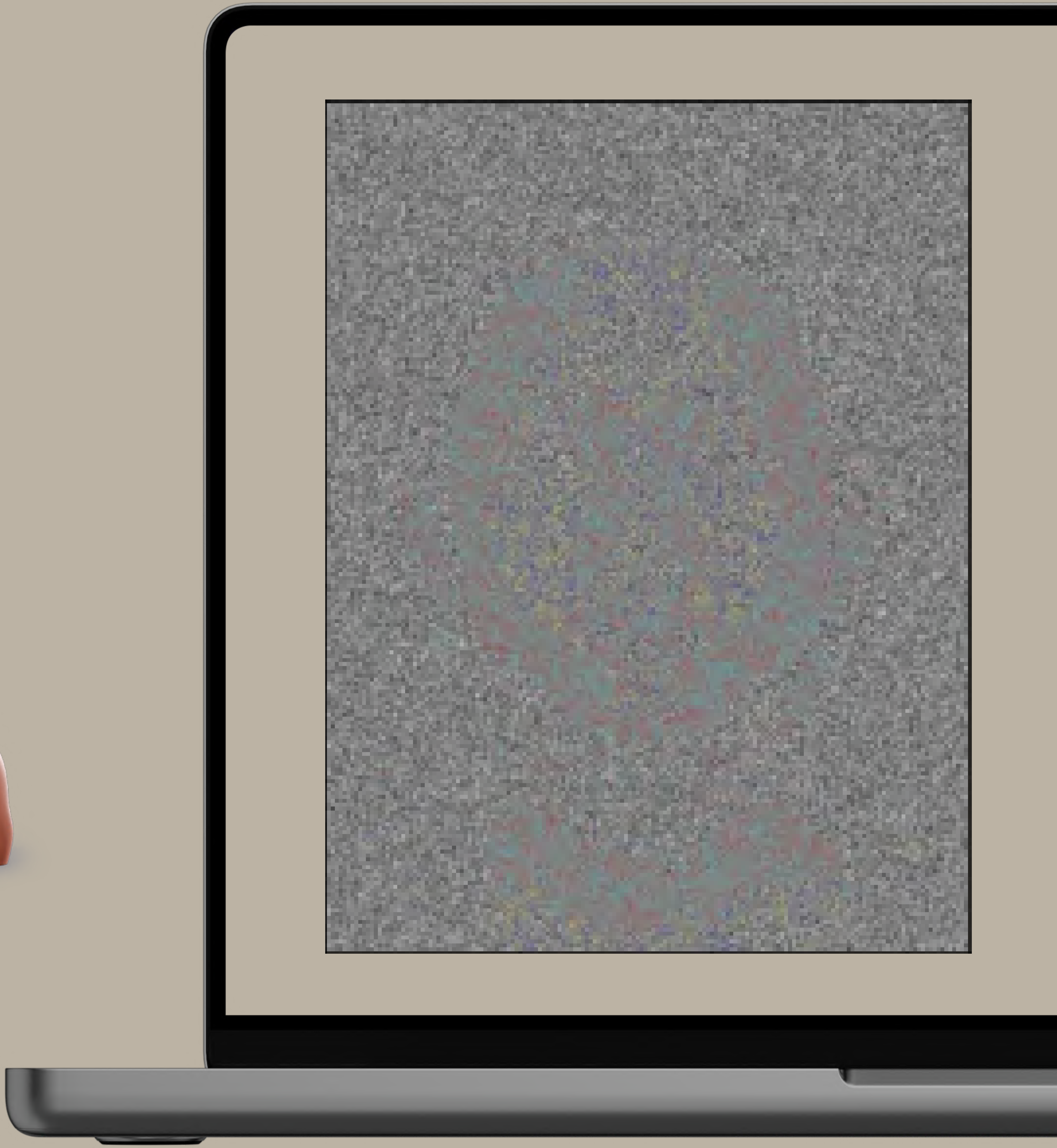
## Speed performance :

Speed of conventional encryption methods like RSA, DES is about 6.483 MB/s while the speed of the proposed system is up to 13.36 MB/s. Chaos seems to be a good candidate due to its ergodicity and complex dynamics.

## Conclusion

- Demonstrated applications of chaos theory in encryption
- Suitable chaotic maps were used to generate chaotic sequences
- Chaotic sequences along with standard permutation, diffusion or XOR methods give rise to good encryption algorithms
- Saw applications of above concepts in audio, text and image encryption
- Saw performance metrics to quantify the effectiveness and speed of encryption

# Thank You



22 September 2022

Pitch