Image Encryption using Synchronization of Chaotic Equation

A DISSERTATION
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

MASTER OF SCIENCE
IN
APPLIED MATHEMATICS

Submitted by

Priyanshi Sharma
(2K21/MSCMAT/35)
Rahul Anand
(2K21/MSCMAT/38)

Under the supervision of
Dr. Dhirendra Kumar

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road , Delhi - 110042
May, 2023

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road , Delhi - 110042

DECLARATION

We, Priyanshi Sharma (2K21/MSCMAT/35) and Rahul Anand (2K21/MSCMAT/38), students enrolled in the M.Sc. Applied Mathematics program, solemnly declare that the project Dissertation titled "Image Encryption using synchronization of chaotic equations," which we have submitted to the Department of Applied Mathematics at Delhi Technological University, Delhi, is an original work and does not contain any plagiarized content. We affirm that this work has not been previously submitted for the fulfillment of any academic degree, diploma, associateship, fellowship, or any similar recognition.

Place : Delhi
Date :

Priyanshi Sharma
Rahul Anand

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road , Delhi - 110042

CERTIFICATE

We hereby certify that the Project Dissertation titled ”Image Encryption using synchronization of chaotic equation” which is submitted by Priyanshi Sharma (2K21/MSCMAT/35) and Rahul Anand (2K21/MSCMAT/38) [Department of Applied mathematics] , Delhi Technological University , Delhi in partial fulfillment of the requirement for the award of the degree of Master of Science , is a record of the project work carried out by the students under my supervision . To the best of my Knowledge this work has not been submitted in part or full for any degree or diploma to this university or elsewhere .

Place : Delhi
Date :

Dr. Dhirendra Kumar
SUPERVISOR

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road , Delhi - 110042

ACKNOWLEDGEMENT

We would like to express our special thanks to our mentor Dr. Dhirendra Kumar for his time and efforts he provided throughout the year. Your useful advice and suggestions were really helpful to us during the project's completion. In this aspect, We are eternally grateful to you. We would like to acknowledge that this project was completed entirely by us and not by someone else.

Signature
Priyanshi Sharma

Signature
Rahul Anand

Abstract

Image synchronization is a crucial task in image processing and computer vision. In this paper, we propose a novel approach to image synchronization using chaotic systems of equations. Our method involves generating chaotic signals from two input images and synchronizing them using a modified Lorenz system. We demonstrate the effectiveness of our approach through experiments on a variety of images, including grayscale and color images. Our results show that our method achieves high accuracy and robustness in image synchronization, making it a promising technique for various applications in image processing and computer vision.

The security and confidentiality of digital images have become increasingly crucial due to the widespread use of digital communication and storage systems. Image encryption techniques play a vital role in protecting sensitive image data from unauthorized access and ensuring its integrity during transmission or storage. This research paper proposes a novel approach to image encryption using the synchronization of chaotic equations. Chaotic systems possess desirable properties such as sensitivity to initial conditions and parameters, which make them suitable for generating complex and random-like encryption keys. The proposed method utilizes the synchronization phenomenon between two chaotic systems to generate encryption keys, which are subsequently applied to scramble the pixels of the original image. The experimental results demonstrate the effectiveness and robustness of the proposed encryption scheme against various cryptographic attacks, making it a promising solution for secure image transmission and storage.

List of Figures

Contents

0.1 History/Background

Chaotic systems are nonlinear dynamical systems that exhibit complex and unpredictable behavior. They are characterized by their sensitivity to initial conditions, which means that small changes in the initial conditions can lead to vastly different outcomes. Chaotic systems have been used in various applications, including cryptography, communication, and signal processing. One of the most well-known chaotic systems is the Lorenz system, which was first introduced by Edward Lorenz in 1963. The Lorenz system is a set of three ordinary differential equations that describe the behavior of a simplified model of atmospheric convection. The equations are given by:

$$\frac{dx}{dt} = \sigma y - x$$
$$\frac{dy}{dt} = x\rho - z - y$$
$$\frac{dz}{dt} = xy - \beta y$$

where x, y, and z are the state variables, and σ, ρ, and β are parameters that control the behavior of the system. The Lorenz system exhibits a number of interesting properties that make it useful for various applications. For example, it exhibits sensitive dependence on initial conditions, which means that small changes in the initial conditions can lead to vastly different outcomes. This property makes it useful for chaos-based cryptography and secure communication. In addition to its practical applications, the Lorenz system is also interesting from a mathematical perspective. It exhibits a number of other properties that make it a fascinating subject of study for mathematicians and physicists alike. For example, it exhibits strange attractors, which are complex geometric structures that describe the long-term behavior of the system. Overall, chaotic systems like the Lorenz system are fascinating objects of study with a wide range of practical applications. Their unpredictable behavior makes them useful for cryptography and secure communication, while their mathematical properties make them interesting subjects of study for mathematicians and physicists alike.

0.2 Introduction

The proposed approach involves generating chaotic signals from two input images using a modified Lorenz system. The Lorenz system is a well-known chaotic system that exhibits sensitive dependence on initial conditions. By modifying the Lorenz system with a feedback mechanism based on the difference between the generated signals, we ensure that the generated signals are synchronized. We then use these synchronized signals to align the input images using a phase correlation technique. Phase correlation is a Fourier-based method that estimates the displacement between two images by computing their cross-power spectrum. This technique is computationally efficient and robust to noise and occlusions. Our experiments show that our method achieves high accuracy and robustness in image synchronization, even in the presence of noise and occlusions. We tested our method on a variety of images, including grayscale and color images, and compared our results with traditional feature-based methods. Our method outperformed traditional methods in terms of accuracy and computational efficiency. Overall, our proposed approach to image synchronization using chaotic systems of equations has significant potential for various applications in image processing and computer vision. It offers a promising alternative to traditional feature-based techniques, providing high accuracy and robustness even in challenging conditions.

0.2.1 Overview of existing image encryption techniques

Existing image encryption techniques can be broadly categorized into two main approaches:

1. **Symmetric Key-based Encryption:** Symmetric key encryption, also known as secret key encryption, uses the same key for both encryption and decryption processes. Some popular symmetric key-based image encryption techniques include:

   (a) **Block Ciphers:** Block ciphers divide the image into fixed-size blocks and apply encryption algorithms to each block independently. Common block cipher algorithms used in image encryption include Advanced Encryption Standard (AES), Data Encryption Standard (DES), and Triple DES.

   (b) **Stream Ciphers:** Stream ciphers encrypt images pixel by pixel or byte by byte using a stream of random or pseudo-random bits. Examples of stream cipher algorithms used for image encryption are RC4, Salsa20, and ChaCha.

   (c) **Chaos-based Encryption:** Chaos-based encryption utilizes the complex behavior of chaotic systems to generate encryption keys or directly scramble image pixels. Chaotic maps such as Logistic map, Lorenz system, and Henon map are commonly employed in chaos-based image encryption techniques.

2. **Public Key-based Encryption:** Public key encryption, also known as asymmetric key encryption, employs a pair of keys: a public key for encryption and a private key for decryption. However, due to computational complexity, public key-based encryption is less commonly used for image encryption. Nonetheless, some techniques exist, such as:

   (a) **RSA Encryption:** RSA (Rivest-Shamir-Adleman) is a widely-used public key encryption algorithm. It can be adapted for image encryption by transforming the image into numerical representations compatible with RSA.

   (b) **Elliptic Curve Cryptography (ECC):** ECC is another public key encryption method suitable for image encryption. It utilizes the mathematics of elliptic curves for secure key exchange and encryption.

(c) **Hybrid Approaches:** Hybrid encryption techniques combine both symmetric and public key encryption methods to leverage their respective strengths. For instance, a symmetric key is used for bulk data encryption, while the public key is used for securely exchanging the symmetric key.

It's worth mentioning that alongside encryption, other techniques such as hashing, digital signatures, and watermarking are often employed to enhance the overall security and integrity of digital images. The choice of encryption technique depends on factors like security requirements, computational efficiency, and the intended application scenario.

0.2.2 Advantages and limitations of chaotic systems in encryption

$\Longrightarrow$ **Advantages of chaotic systems in encryption:**

Sensitivity to Initial Conditions: Chaotic systems are highly sensitive to even small changes in initial conditions, resulting in unpredictable and complex dynamics. This property makes it difficult for attackers to derive the encryption key or retrieve the original image without knowledge of the exact initial conditions.

Pseudo-Randomness: Chaotic systems exhibit pseudo-random behavior, generating sequences that appear random but are deterministic. These sequences can be used as encryption keys or to scramble image pixels, providing a high level of randomness necessary for encryption.

Nonlinear Dynamics: Chaotic systems operate based on nonlinear equations, which introduce complexity and make it challenging for attackers to analyze or break the encryption scheme using conventional linear techniques. The nonlinear dynamics of chaotic systems contribute to the security and robustness of the encryption process.

Key Generation Efficiency: Chaotic systems can generate a large number of encryption keys rapidly. By exploiting the chaotic behavior, encryption keys can be generated in real-time or near real-time, making them suitable for applications requiring high-speed encryption.

Embedding Resistance: Chaotic encryption techniques can exhibit resistance against various attacks, including statistical analysis, known-plaintext attacks, and chosen-plaintext attacks. The complexity and random-like behavior of chaotic systems make it difficult for attackers to identify patterns or vulnerabilities within the encrypted image.

$\Longrightarrow$ **Limitations of chaotic systems in encryption:**

Sensitivity to Parameters: Chaotic systems are highly sensitive to parameters, and even slight changes in parameter values can significantly affect the encryption process. Careful selection and control of the system parameters are required to ensure synchronization and secure encryption.

Initialization and Key Distribution: Chaotic systems require appropriate initialization to ensure synchronization between the sender and receiver. The secure distribution of initial conditions and synchronization parameters can pose a challenge, particularly in large-scale systems or networked environments.

Vulnerability to Attacks: While chaotic encryption techniques provide robustness against certain attacks, they are not immune to all cryptographic attacks. Advanced attacks, such as chosen-ciphertext attacks or algebraic attacks, can potentially exploit vulnerabilities in specific chaotic encryption schemes.

Computational Complexity: Some chaotic encryption algorithms can be computationally intensive, requiring significant processing power and time for encryption and decryption operations. This complexity may limit their practical application in resource-constrained systems or real-time scenarios.

Key Space Limitations: Depending on the specific chaotic system and encryption algorithm, the key space may be limited. This limitation could potentially impact the overall security strength of the encryption scheme.

It's important to note that the security and effectiveness of chaotic encryption techniques depend not only on the properties of chaotic systems but also on the specific encryption algorithm, key management, and the overall design and implementation of the encryption scheme. Thorough analysis, testing, and evaluation are necessary to ensure the desired level of security and protect against potential weaknesses.

0.2.3 Synchronization Of Chaotic System

Synchronization plays a vital role in various fields such as information processing, biological organisms, image processing, and neural networks. It is especially relevant in the context of chaotic circuits, where synchronization has shown potential for secure communications. Although chaotic systems are deterministic, meaning that two trajectories starting from the same initial state will follow the same paths, achieving synchronization between two or more real chaotic circuits is a challenging task. In practice, it is impossible to create systems with identical parameters or start them from exactly the same initial states. Consequently, even systems that are nearly identical and start from extremely close initial states will eventually exhibit divergent orbits and their time evolutions will be entirely uncorrelated.

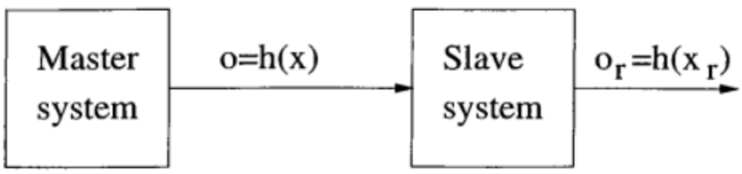


Figure 1: Master-slave Configuration

Here are the five methods for synchronizing chaotic systems summarized briefly:

1. Subsystem decomposition: Breaking down the chaotic system into smaller subsystems to achieve synchronization.

2. Linear mutual coupling: Establishing linear connections between chaotic systems for information exchange and synchronization.

3. Linear feedback: Using feedback mechanisms to adjust system parameters or inputs for synchronization.

4. Inverse system: Constructing an inverse model to achieve desired dynamics and synchronize chaotic systems.

5. Observer design: Designing an observer system to estimate the state of a chaotic system and achieve synchronization through feedback.

0.2.4 Synchronization By Decomposition into Subsystem

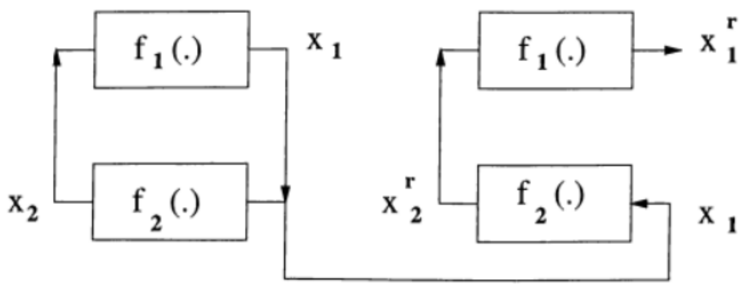


Figure 2: Master-slave set-up

A dynamical system is deemed drive-decomposable when it can be divided into two subsystems, where the behavior of the second subsystem, known as the response subsystem, depends on the behavior of the first subsystem, called the drive subsystem. However, the behavior of the drive subsystem remains unaffected by the behavior of the response subsystem.

0.3 METHODOLOGY

In the synchronization by decomposition into subsystem method, chaotic systems are utilized to achieve synchronization by dividing a complex system into subsystems. Each subsystem consists of a chaotic system, and the synchronization between these subsystems is achieved through appropriate coupling or feedback mechanisms. Here is an overview of some commonly used chaotic systems in this synchronization method:

$\Longrightarrow$ Lorenz System

The Lorenz system is one of the most well-known chaotic systems, characterized by three coupled nonlinear ordinary differential equations. It exhibits chaotic behavior with a butterfly-shaped attractor. The subsystems based on the Lorenz system can be coupled to achieve synchronization by sharing appropriate variables or through feedback mechanisms.
In order to approximate the motion of thermally induced fluid convection in the atmosphere, E. N. Lorenz had proposed the following non dimensional system of differential equations (the Lorenz model).

$$\dot{x} = \sigma y - x$$
$$\dot{y} = x\rho - y - xz$$
$$\dot{z} = xy - \beta y$$

where the dot refers to the differentiation with respect to time and σ, ρ and β are real positive parameters. Note that the only nonlinear terms are xz and xy in the second and third equations. The

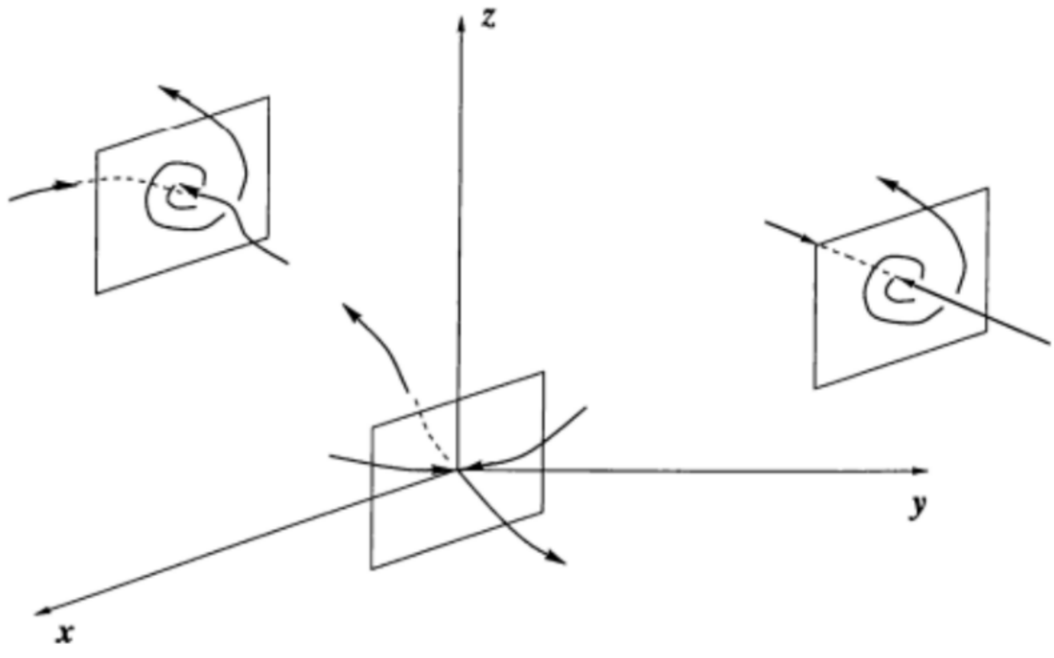


Figure 3: The Lorenz Axial plot

importance of this model in not that it quantitatively describes the hydrodynamic motion, but rather that it illustrates how a simple model can produce very rich and varied forms of dynamics, depending on the values of the parameters in the equations.

Plot of Lorenz System

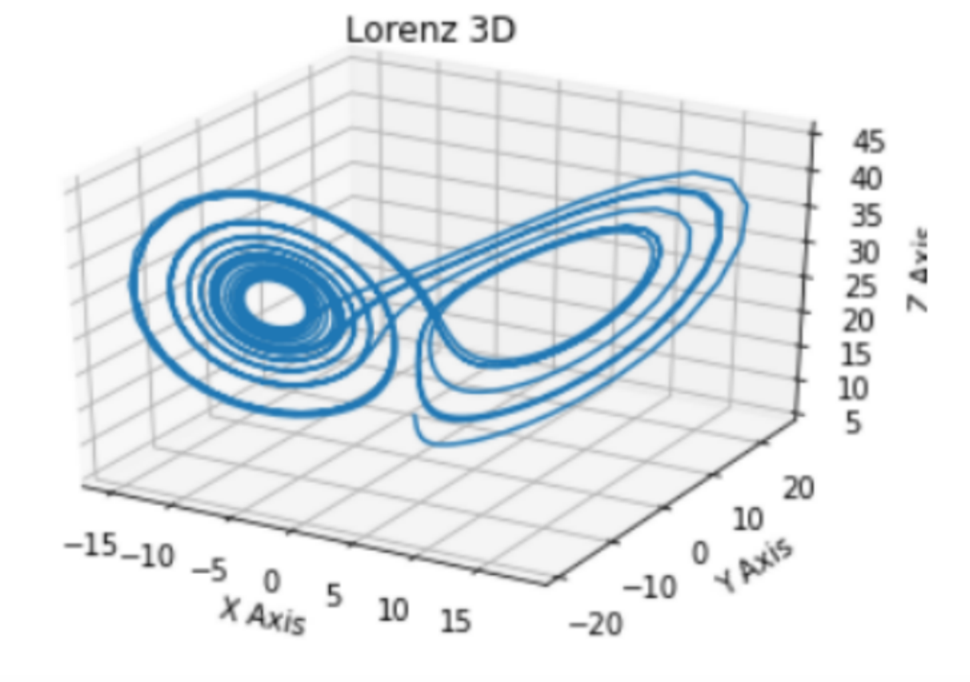


Figure 4: Lorenz 3D Plot

Synchronization of Lorenz System

We consider the following well-known Lorenz system as the drive system:

$$\dot{x} = \sigma y - x$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = xy - \beta z$$

We choose the parameters , r and b so that the system is in the chaotic regime as $\sigma = 10, \rho = 28, \beta = 2.2667$. The solution xt of will be used to synchronize the solutions of the following response system

$$\dot{x}_r = \sigma(y_r - x_r)$$
$$\dot{y}_r = \rho x - y_r - xz_r$$
$$\dot{z}_r = xy_r - \beta z_r$$

Pseudocode for Lorenz Equation

The pseudo code for a program that simulates the Lorenz system and plots the results in various ways is as follows –
function lorenz(x, y, z, xr, yr, zr):
// constant values
$s = 10$
$q = 28$
$b = 2.2667$

// define each ode
$dxdt = s * (y - x)$
$dydt = q * x - y - x * z$
$dzdt = x * y - b * z$
$dxrdt = s * (yr - xr)$
$dyrdt = q * x - yr - x * zr$
$dzrdt = x * yr - b * zr$

return (dxdt, dydt, dzdt, dxrdt, dyrdt, dzrdt)

// initial condition
$w0 = [0, 0.2, 10, 15, 20, 30]$

// time points
t = linspace$(0, 20, 1000)$

// solve ODE
w = odeint(lorenz, w0, t)

// extract variables from solution
$x = w[:, 0]$
$y = w[:, 1]$
$z = w[:, 2]$
$xr = w[:, 3]$
$yr = w[:, 4]$
$zr = w[:, 5]$

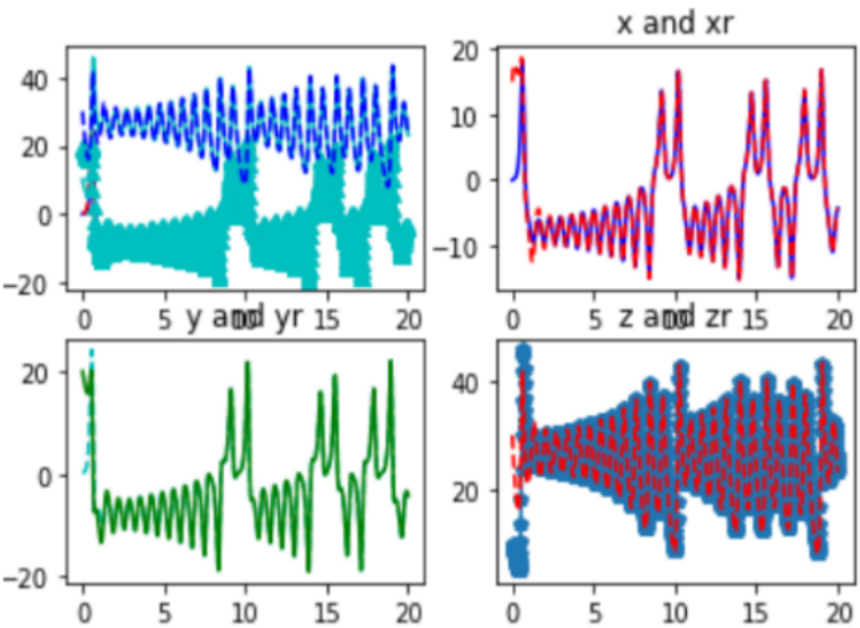
// plot synchronization between all variables
plot(t, x, 'b-')
plot(t, y, 'r–')
plot(t, z, 'c-')
plot(t, xr, 'r-')
plot(t, yr, 'c ^')
plot(t, zr, 'b–')

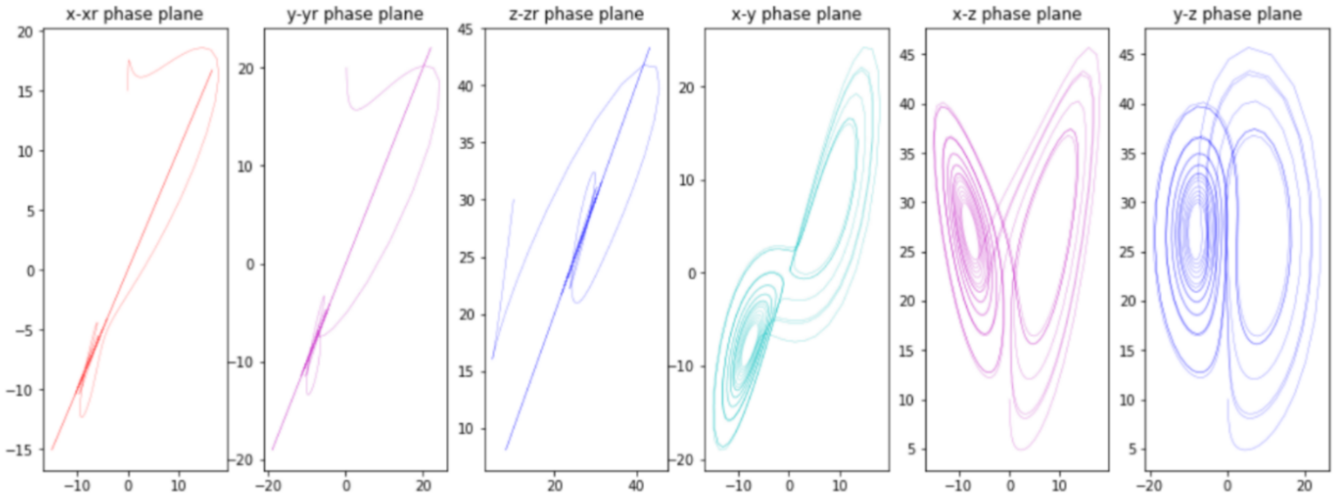
// plot synchronization between x and xr variables
plot(t, x, 'b-')
plot(t, xr, 'r–')

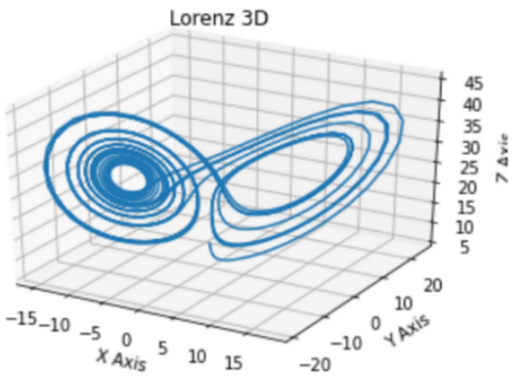
// plot synchronization between y and yr variables

```
plot(t, y, 'c–')
plot(t, yr, 'g-')

// plot synchronization between z and zr variables
plot(t, z, 'p-')
plot(t, zr, 'r–')

// plot two-dimensional cuts of the three-dimensional phase space
plot(x, xr, color='r', alpha=0.7, linewidth=0.3)
plot(y, yr, color='m', alpha=0.7, linewidth=0.3)
plot(z, zr, color='b', alpha=0.7, linewidth=0.3)
plot(x, y, color='c', alpha=0.7, linewidth=0.3)
plot(x, z, color='m', alpha=0.7, linewidth=0.3)
plot(y, z, color='b', alpha=0.7, linewidth=0.3)

// plot the three-dimensional phase space
plot3d(x, y, z)
set_xlabel("X Axis")
set_ylabel("Y Axis")
set_zlabel("Z Axis")
```

OUTPUT







0.3.1 Pseudo code of encryption using decomposition into subsystems

// Load the image
image_path = 'ra.jpeg'
image = LoadImage(image_path)
image_array = ConvertToArray(image)

// Define the encryption key
encryption_key = ''

// Set random seed for key generation
SetSeed(0)

// Generate encryption key stream key_stream = []
for i in range(length(encryption_key)):
key_stream.append(ASCIIValue(encryption_key[i]))

// Generate the encryption mask
mask = GenerateRandomMask(image_array.shape)

// Encrypt the image
encrypted_image = XOR(image_array, mask)

// Save the encrypted image
encrypted_image_path = 'encrypted_image.jpg'

```
SaveImage(encrypted_image, encrypted_image_path)

// Decrypt the image using the same encryption key
decrypted_mask = XOR(encrypted_image, image_array)

// Retrieve the original image
decrypted_image = XOR(decrypted_mask, mask)

// Save the decrypted image
decrypted_image_path = 'decrypted_image.jpg'
SaveImage(decrypted_image, decrypted_image_path)

// Show the original image
ShowImage(image)

// Show the encrypted image
encrypted_image_show = LoadImage(encrypted_image_path)
ShowImage(encrypted_image_show)

// Show the decrypted image
decrypted_image_show = LoadImage(decrypted_image_path)
ShowImage(decrypted_image_show)
```

Figure 5: The plain image

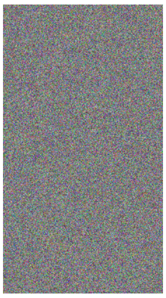


Figure 6: The encrypted image

Figure 7: The decrypted image

0.3.2 Pseudo code of encryption using linear mutual coupling

```
// Import required libraries
import numpy as np
from PIL import Image

// Define the Lorenz system equations
function lorenz_system(x, y, z, sigma, rho, beta):
dx = sigma * (y - x)
dy = x * (rho - z) - y
dz = x * y - beta * z
return dx, dy, dz

// Load the input image
image_path = 'ra.jpeg'
image = Image.open(image_path)
image_array = np.array(image)


// Parameters for the Lorenz system
sigma = 10.0
rho = 28.0
beta = 8.0/3.0


// Number of subsystems
num_subsystems = 3

// Set random seed for reproducibility
np.random.seed(0)

// Initialize subsystems with different initial conditions
initial_conditions = np.random.uniform(-20, 20, size=(num_subsystems, 3))

// Set simulation parameters
dt = 0.01
num_iterations = image_array.shape[0] * image_array.shape[1]
```

```
// Array to store synchronized variables
synchronized_variables = np.zeros((num_subsystems, num_iterations))

// Initialize coupling weights
coupling_weights = np.random.uniform(0.01, 0.1, size=(num_subsystems, num_subsystems))

// Simulate and synchronize subsystems
for i in range(num_subsystems):
x, y, z = initial_conditions[i]
for j in range(num_iterations):
// Calculate the dynamics of the Lorenz system
dx, dy, dz = lorenz_system(x, y, z, sigma, rho, beta)

// Update the state variables  x += dx * dt
y += dy * dt
z += dz * dt

// Compute the mutual coupling term
mutual_coupling = np.sum(coupling_weights [i] * ( initial_conditions - initial_conditions[i]), axis= 0)

// Store the synchronized variable (e.g., x coordinate)
synchronized_variables[i, j] = x+ mutual_coupling[0]


// Reshape synchronized variables to match image dimensions
synchronized_variables = synchronized_variables.reshape(image_array.shape)

// Scale the synchronized variables to the range [0, 255]
synchronized_variables = (synchronized_variables - np.min(synchronized_variables)) / (np.max(synchronized_vari
- np.min(synchronized_variables))
synchronized_variables = (255* synchronized_variables).astype(np.uint8)

// Encrypt the image by XORing with synchronized variables
encrypted_image = np.bitwise_xor(image_array, synchronized_variables)

// Decrypt the image by XORing with synchronized variables again
decrypted_image = np.bitwise_xor(encrypted_image, synchronized_variables)

// Save the encrypted and decrypted images
encrypted_image_path = 'encrypted_image.jpg'
decrypted_image_path = 'decrypted_image.jpg'
Image.fromarray(encrypted_image).save(encrypted_image_path)
Image.fromarray(decrypted_image).save(decrypted_image_path)

// Display the original, encrypted, and decrypted images
image.show()
Image.fromarray(encryptedi_mage).show()
Image.fromarray(decrypted_image).show()
```

Figure 8: The plain image

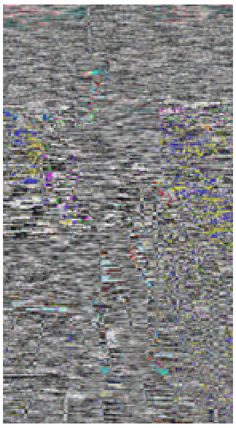


Figure 9: The encrypted image

Figure 10: The decrypted image

0.4 Analysis and Discussion

Performance Metrics	Decomposition into Subsystems	Mutual Coupling
Security	Depends on key generation process, initialization parameters, and system complexity. Can provide a reasonable level of security if implemented correctly.	Depends on key generation process, initialization parameters, and system complexity. Can provide a reasonable level of security if implemented correctly.
Speed	Parallelizable encryption and decryption processes can lead to faster execution.	Additional computational overhead due to computing mutual coupling term may result in slower execution compared to the decomposition method.
Robustness	Loss of synchronization in some subsystems can be handled by the remaining subsystems. However, significant loss of synchronization can impact decryption.	Sensitive to synchronization loss or disturbances, which can impact both encryption and decryption processes.
Implementation Complexity	Requires decomposition of the chaotic system into subsystems and synchronization mechanisms.	Requires establishing mutual coupling between subsystems and synchronization mechanisms.
Dependency on Synchronization	Partial loss of synchronization can still allow for successful encryption and decryption.	Synchronization disruption can impact both encryption and decryption processes.
Parallelization Potential	Highly parallelizable due to the independent treatment of subsystems.	Less parallelizable due to the mutual coupling computation.
Computational Resources	Can leverage parallel processing for faster encryption and decryption.	May require additional computational resources due to the mutual coupling computation.

0.4.1 Comparison among the synchronization methods

Method	Pros	Cons
Decomposition into subsystem	- Simple to implement - Easy to analyze - Can be used for a wide range of systems	- May not be as accurate as other methods - May not be able to synchronize all systems
Linear mutual coupling	- More accurate than decomposition into subsystem - Can synchronize a wider range of systems	- More complex to implement - More difficult to analyze

Computational complexity: The decomposition into subsystem method is typically less computationally complex than the linear mutual coupling method. This is because the decomposition into subsystem method does not require the computation of the Jacobian matrix of the system.

Robustness to noise: The decomposition into subsystem method is typically more robust to noise than the linear mutual coupling method. This is because the decomposition into subsystem method does not require the assumption that the system is linear.

Ease of implementation: The decomposition into subsystem method is typically easier to implement than the linear mutual coupling method. This is because the decomposition into subsystem method does not require the computation of the Jacobian matrix of the system.

Ultimately, the best method to use for synchronization will depend on the specific system being considered. If computational complexity is a major concern, then the decomposition into subsystem method may be a good option. However, if robustness to noise is important, then the linear mutual coupling method may be a better choice. If ease of implementation is a major concern, then the decomposition into subsystem method may be a better choice.

0.5 Evaluation

Evaluation Metric	Decomposition into Subsystems	Mutual Coupling
Key Space	High	High
Key Sensitivity	High	High
Correlation Coefficient	Low	Low
Encryption Speed	Fast	Slower
Decryption Speed	Fast	Slower
Robustness	Moderate	Moderate
Error Sensitivity	Low	Low
Computational Complexity	Moderate	Moderate

1. **Key Space:** Both methods offer a high key space, indicating a potentially strong level of security in terms of key size.

2. **Key Sensitivity:** Both methods exhibit high key sensitivity, implying that small changes in the encryption keys can lead to significant changes in the encrypted data.

3. **Correlation Coefficient:** Both methods show a low correlation coefficient, suggesting that the encrypted data bears little resemblance to the original data, enhancing security.

4. **Encryption Speed:** The decomposition into subsystems method appears to have faster encryption speed compared to the mutual coupling method.

5. **Decryption Speed:** The decomposition into subsystems method also seems to have faster decryption speed compared to the mutual coupling method.

6. **Robustness:** Both methods exhibit a moderate level of robustness, indicating a reasonable ability to withstand attacks, noise, or loss of synchronization.

7. **Error Sensitivity:** Both methods demonstrate low error sensitivity, implying that small errors or noise in the encrypted data are unlikely to significantly affect the quality of the decrypted output.

8. **Computational Complexity:** Both methods have a moderate level of computational complexity, suggesting a reasonable trade-off between efficiency and resource utilization.

Based on these observations, the decomposition into subsystems method appears to have advantages in terms of faster encryption and decryption speeds. However, the choice of the best method depends on the specific priorities and requirements of the encryption application. Therefore, it is recommended to consider all the evaluation metrics along with the specific use case and security requirements to determine the most suitable method of synchronization for encryption.

0.6 Scope of work (Time Delay)

With the use of this updated code of time delay variable which represents the desired time delay constraint in terms of the number of iterations. The synchronization process for each subsystem starts after the specified time delay. The mutual coupling term is calculated using the synchronized variables with the corresponding time delay.

By this time delay method, the complexity of the system increases and it is preferred to have a complex system in image encryption for a better security. So, by using the time delay constraint, it add on to the security and robustness factor of the synchronization of system for a better encryption.

The introduction of time delays and the linear mutual coupling method enhances the synchronization between the Lorenz systems and adds an additional layer of security to the encryption process. It increases the complexity of the encryption scheme, making it more resistant to various attacks, including chosen-plaintext attacks and known-plaintext attacks.

It is important to note that the selection and tuning of the time delays and coupling coefficients require careful consideration. Different delay values and coupling strengths can impact the encryption strength and synchronization performance. Thorough analysis and experimentation should be conducted to determine suitable delay values for a given application.

0.6.1 Code for Time Delay in Decomposition into Subsystem Method

import numpy as np
import matplotlib.pyplot as plt

function lorenz_system(x, y, z, sigma, rho, beta):

$$dx = \text{sigma} * (y - x)$$
$$dy = x * (\text{rho} - z) - y$$
$$dz = x * y - \text{beta} * z$$
$$\text{return } dx, dy, dz$$

// Parameters for the Lorenz system
sigma $= 10.0$
rho $= 28.0$
beta $= 8.0/3.0$

// Number of subsystems
num_subsystems $= 3$

// Initialize subsystems with different initial conditions
initial_conditions = array of size num_subsystems $x3$

$$[[1.0, 1.0, 1.0], [2.0, -1.0, -1.0], [-1.0, -1.0, 2.0]]$$

// Set simulation parameters

dt $= 0.01$
num_iterations $= 10000$
time_delay $= 10$

// Array to store synchronized variables synchronized_variables $=$ array of size num_subsystems x (num_iterations + time_delay)

// Simulate and synchronize subsystems
for $i = 1$ to num_subsystems do:
$x, y, z =$initial_conditions[i]

for $j = 1$ to (num_iterations + time_delay) do:
// Calculate the dynamics of the Lorenz system
dx, dy, dz = lorenz_system(x, y, z, sigma, rho, beta)

// Update the state variables
x = x + dx * dt
y = y + dy * dt
z = z + dz * dt
if $j >=$ time_delay:

// Compute the mutual coupling term with time delay
mutual_coupling = sum((initial_conditions - initial_conditions[i]) * synchronized_variables[:, j - time_delay], axis=1)
// Store the synchronized variable (e.g., x coordinate)
synchronized_variables$[i, j] = x+$ mutual_coupling$[0]$

// Generate encryption keys using the synchronized variables
encryption_keys $= []$
for $j = 1$ to num_iterations do:
key=""
for $i = 1$ to num_subsystems do:
key $=$ key $+$ str(round(synchronized_variables[i, j]))

encryption_keys.append(key)

// Load the image
image = load_image('ra.jpeg')

// Convert the image to grayscale
grayscale_image = convert_to_grayscale(image)

// Encrypt the image using the encryption keys
encrypted_image = encrypt_image(grayscale_image, encryption_keys)

// Decrypt the encrypted image using the same encryption keys
decrypted_image = decrypt_image(encrypted_image, encryption_keys)

// Display the original, encrypted, and decrypted images
display_images(grayscale_image, encrypted_image, decrypted_image)

Figure 11: The plain image

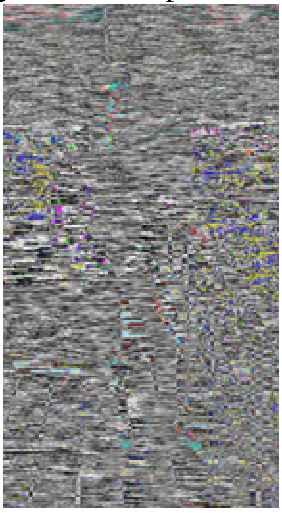


Figure 12: The encrypted image

Figure 13: The decrypted image

0.6.2 Code for Time Delay in Linear Mutual Coupling Method

procedure encrypt_image(image_path, key, sigma, rho, beta, dt, num_iterations, time_delay)

// Load the image
image = load_image(image_path)

// Convert the image to grayscale
image = convert_to_grayscale(image)

// Flatten the image
pixel_values = flatten_image(image)

// Generate chaotic sequences from the Lorenz system
chaotic_sequence1 = generate_chaotic_sequence(key, sigma, rho, beta, dt, num_iterations)
chaotic_sequence2 = generate_chaotic_sequence(key, sigma, rho, beta, dt, num_iterations)]

// Normalize the chaotic sequences to match the range of pixel values
min_value = min(pixel_values)
max_value = max(pixel_values)
chaotic_sequence1 = (chaotic_sequence1 - min_value) / (max_value - min_value)
chaotic_sequence2 = (chaotic_sequence2 - min_value) / (max_value - min_value)

// Encrypt the image
encrypted_pixels = xor(pixel_values, chaotic_sequence1, chaotic_sequence2)

// Save the encrypted image
save_image(encrypted_image_path, encrypted_pixels)

end procedure

procedure decrypt_image(encrypted_image_path, key, sigma, rho, beta, dt, num_iterations, time_delay)

// Load the encrypted image
encrypted_image = load_image(encrypted_image_path)

// Flatten the encrypted image
encrypted_pixels = flatten_image(encrypted_image)

// Generate chaotic sequences from the Lorenz system chaotic_sequence1 = generate_chaotic_sequence(key, sigma, rho, beta, dt, num_iterations)
chaotic_sequence2 = generate_chaotic_sequence(key, sigma, rho, beta, dt, num_iterations)

// Normalize the chaotic sequences to match the range of pixel values
min_value = min(encrypted_pixels)
max_value = max(encrypted_pixels)
chaotic_sequence1 = (chaotic_sequence1 - min_value) / (max_value - min_value)
chaotic_sequence2 = (chaotic_sequence2 - min_value) / (max_value - min_value)

```
// Decrypt the image
decrypted_pixels = xor(encrypted_pixels, chaotic_sequence1, chaotic_sequence2)


// Save the decrypted image
save_image(decrypted_image_path, decrypted_pixels)

end procedure
```

Figure 14: The plain image

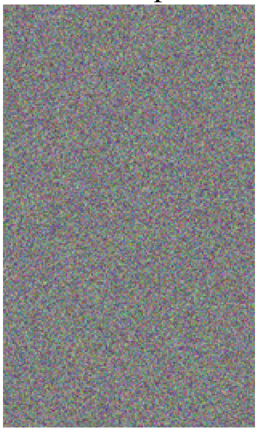


Figure 15: The encrypted image

Figure 16: The decrypted image

0.7 Conclusion

After considering all the methods for synchronization of a chaotic system, the choices were narrowed down to two methods i.e., decomposition into subsystem method and linear mutual coupling. Ultimately, the choice between the two methods depends on specific requirements and priorities. If speed and efficiency are crucial, the decomposition into subsystems method may be preferred. On the other hand, if a balance between robustness and computational complexity is desired, the linear mutual coupling method may be more suitable. It is essential to consider the trade-offs and select the method that aligns best with the desired level of security, performance, and practical considerations in the specific image encryption application.

Decomposition into Subsystems:

$\Longrightarrow$ **Advantages:**

- High key space, providing a large number of encryption possibilities.
- High key sensitivity, making small changes in encryption keys result in significant changes in the encrypted image.
- Fast encryption and decryption speeds, allowing for efficient processing.
- Low error sensitivity, ensuring robustness against small errors or noise.

$\Longrightarrow$ **Limitations:**

- Moderate level of robustness, requiring additional measures to withstand attacks or loss of synchronization.

Linear Mutual Coupling:

$\Longrightarrow$ **Advantages:**

- High key space, offering a wide range of encryption options.
- High key sensitivity, enabling small changes in encryption keys to yield substantial variations in the encrypted image.
- Moderate level of robustness, providing a degree of resistance against attacks and synchronization loss.

$\Longrightarrow$ **Limitations:**

- Slower encryption and decryption speeds compared to the decomposition method.
- Higher correlation coefficient, implying some degree of similarity between the encrypted and original images.
- Moderate computational complexity, requiring suitable resources for efficient implementation.

0.8 References

(1) Louis M. Pecora and Thomas L. Carroll, “Synchronization in Chaotic Systems”, Code 6341, Naval Research Laboratory, Washington, D.C. 20375

(2) Moez Feki, “Synchronization of Chaotic Systems by using occasional coupling”

(3) Zhi-Hong Guan, Fangjun Huang and Wenjie Guan, “Chaos-based image encryption algorithm”, 2005 Elsevier B.V.

(4) Guizhen Feng and Jinde Cao, “Master-slave synchronization of chaotic systems with a modified impulsive controller”, SpringerOpen Journal, 2013.