MA202 PROJECT

DIGITAL IMAGE ENCRYPTION USING LORENZ EQUATION AND CHAOTIC SYSTEMS

Haikoo Khandor	Kanishk Singhal	Ksheer Agrawal	Madhav Kanda	Medhansh Singh
20110071	20110091	20110098	20110104	20110111

"Order in Chaos"

"You must have chaos within you to give birth to a dancing star." - Friedrich Nietzsche

Contents

Acknowledgement

We would like to thank all the three professors of the MA202 course for giving us this opportunity. Throughout the process, we learned the qualities of cooperation, leadership and understanding. We gained vital experience in learning new topics like image encryption and the methods used to encrypt and decrypt it. We would also like to thank Prof. Uddipta Ghosh for providing valuable guidance and support throughout the project. We would also like to express gratitude to all the professors of the MA 202 course and the institute for helping us develop our creative thinking.

Problem Statement

With the onset of the information age, the demand for new encryption techniques is ever increasing. Encryption is important to protect sensitive important data from getting into the wrong hands. Encryption is used in technologies such as net banking, credit cards, emails etc. In a similar fashion, images are encrypted.

The requirement for image encryption is :-

- The ability to get the pixels of the original image.
- Create a strong encrypted image such that it cannot be decrypted easily.
- Perfection in the decrypted image.

Chaos theory is an interdisciplinary scientific theory and branch of Mathematics that focuses on underlying patterns and deterministic laws which are highly sensitive to initial conditions in dynamical systems. The Lorenz system is one such set of equations which has chaotic solutions for certain parameters and initial conditions. Runge-Kutta method is an iterative numerical method, used to evaluate Ordinary Differential Equations. The Runge-Kutta will be used to evaluate values in the Lorenz Equation.

The idea is to use Lorenz equation and RK-method to encrypt an image that fulfills the above mentioned criteria.

Notations

1. $\frac{db}{da} = Change\ of\ b\ with\ respect\ to\ a.$
2. x, y, z coordinates in the 3d cartesian system, t is time.
3. σ - sigma, ρ - rho and β - Beta, the three parameters used in the Lorenz system of equations.

Introduction

Encryption and Decryption

Encryption is a method of safeguarding digital data by using mathematical procedures using a key which is used to decode the data. This process uses an algorithm that renders the original data unreadable. The original text known as the plaintext is converted into ciphertext, which is the result of encryption. When an authorised user needs to access the data, they can use a binary key to decrypt it. This will convert ciphertext to plaintext, allowing the authorised user to view the original data. When an authorised user needs to access the data, they can use a binary key to decrypt it. This will convert ciphertext to plaintext, allowing the authorised user to view the original data.

Why Chaos and Lorenz equations?

In order to give justice to the core idea of encryption, it becomes necessary to use a system or a method in place which becomes difficult to predict by both the people who encrypted the information and the ones whom we do not want to decrypt the information. There is also one more crucial aspect to this. We must have an initial set of values/conditions/keys such that they when used gives the exact solution to the encryption i.e. decrypts the file/image. Hence we use chaos to solve this problem. Taking into the example of a Lorentz system of equations, even a slight variation in the initial value conditions can lead to a large deviation in the output values as time goes on. In other words, it becomes difficult or impossible to encrypt the file even if the exact values are not known. If any person tries to approximate or guess the initial values, the decryption they get may be far from the original image thus preventing them from obtaining the original image. This is synonymous to the Butterfly effect. It depicts that a small change in the deterministic initial value can lead to a larger change in the later stage.

Chaos has a lot of applications in today's world especially where you want to protect key information from hackers, phishing attacks etc. It incorporates randomness in the model so it becomes very difficult to predict the outcome. This simple yet complex algorithm finds use in many application test cases in the world today.

Lorenz equation is a set of three ordinary differential equations:

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

The Lorentz attractor is another such system of equations. It takes in three parameters which when varied makes the solutions chaotic i.e. vary greatly and becomes difficult to predict.

The parameters are:

1. Sigma
2. Beta
3. Rho

We will now use Lorentz attractor for image encryption. The Lorentz attractor is visualized by solving and plotting the solutions of the Lorenz system by choosing certain values for the parameter.

The Lorenz system is non-linear, aperiodic and deterministic provided we have the exact initial conditions. The shape of the graph and the number of attraction points i.e. the point around which the graph appears to circle around or revolve around differ for different values of Sigma, Rho and Beta. The values used for sigma, rho and beta for our demonstration are:-

$\sigma = 10$

$\rho = 28$

$\beta = 8/3$

The choice of sigma, beta and rho is not random and is chosen to form the Lorenz attractor. The Lorenz system has chaotic solution at the given values. For small values of rho, the system is stable and evolves to one of the fixed points, but for higher values the trajectory becomes complex.

Runge Kutta Methods

The Runge Kutta methods, or also known as RK methods are iterative methods to compute solutions to ordinary differential equations (ODEs). They are used to solve Initial value problems (IVPs). Runge Kutta methods produce the results by numerically iterating over the function itself rather than computing their derivatives. This property marks the popularity of this method.

The building concept of the Runge Kutta method comes from the Mean value theorem of integrals. It begins with firstly dividing the integral into N steps or subintervals like so,
[x_i , x_{i+1}] , where i ϵ [0,1,2,....,n-1]

After dividing the intervals, iterative methods are used to calculate next value from previous value with the help of Mean value theorem.

Let the Initial Value Problem be,

$$\{ y' = f(x,y),\ a<=x<=b$$
$$\{y(a) = y0$$

Using Mean Value Theorem,

$$y_{i+1} - y_i = \int_{i}^{i+1} f(x, y(x))$$

$$y_{i+1} = y_i + h\, f(\alpha\, , y(\alpha))\ \text{, where } h = x_{i+1} - x$$

If we approximate values of f by linear combination of $x, y(x)$ at different values, we will get various computation formulas for Runge kutta method. The most commonly used method is the fourth order Runge kutta method, commonly known as "RK4" or the "classical Runge Kutta method".

$$\{y_{n+1} = y_n + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$
$$\{K_1 = hf(x_n, y_n)$$
$$\{K_2 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}K_1)$$
$$\{K_3 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}K_2)$$
$$\{K_4 = hf(x_n + h, y_n + K_3)$$

This method needs to compute K_1, K_2, K_3, K_4, i.e. four values to compute the next iterative result, hence this method is called fourth order Runge kutta method.

Methodology

1. We define an algorithm to encrypt and decrypt an image of size `n = hxw`.
2. We employ the classic "RK-4" Runge Kutta method on the lorenz system to create the encrypted image and its key.
3. The Lorentz system is a set of three ordinary differential equations. We have set the parameters of the lorentz system such that the system exhibits chaotic behaviour. Even if the initial conditions are similar (**not same**) the keys generated would be chaotic.
4. The input parameter of the function `RungeKutta4(n):` is taken as the image size.
5. The time factor `dt` is set small to sustain the legitimacy of the RK-4 method. RK provides a much more accurate solution than the euler method. For a step size of `dt = 0.1`, the local truncation error will be of the error dt^5.
6. A key of size `n = hxw` is initialised with zeroes.
7. The RK method evaluates the value of the z variable in the lorentz system as the key array of size `(n)`.
8. We convert the values of the key array into a number between 0 & 255 in order `O(n)`
9. We created the encrypted image by performing the XOR operation on the input image and the key.
10. We perform XOR operation again on the encrypted image to get the original image back.

Algorithm

```
# Function returning the dimensions of the image as n = h X w
def ProvideDimension(image):
    return image.shape[0], image.shape[1]

# Defining the lorentz equations and parameters
def LorenzEquations(x, y, z):
    sigma = 10.0
    beta = 8.0/3.0
    rho = 28.0

    dx = (sigma*(y - x))
    dy = (rho*x - y - x*z)
    dz = (-1*(beta*z) + x*y)
    return dx, dy, dz

# Defining the Runge Kutta Method
def RungeKutta(x0, y0, z0, n):

    # Initializing array x, y & z of size n+1 with zeroes
    x = np.zeros(n+1)
    y = np.zeros(n+1)
    z = np.zeros(n+1)

    x[0] = x0
    y[0] = y0
    z[0] = z0

    # Defining the increment dt involved in the numerical algorithm
    T = 25
    dt = T/float(n)
```

```
        # Evaluating array x, y & z values for t = k*dt.

        for k in range(n):
                k1, l1, m1 = LorenzEquations(x[k], y[k], z[k])
                k2, l2, m2 = LorenzEquations(x[k] + 0.5*k1*dt, y[k] +
        0.5*l1*dt, z[k] + 0.5*m1*dt)
                k3, l3, m3 = LorenzEquations(x[k] + 0.5*k2*dt, y[k] +
        0.5*l2*dt, z[k] + 0.5*m2*dt)
                k4, l4, m4 = LorenzEquations(x[k] + k3*dt, y[k] + l3*dt,
        z[k] + m3*dt)

          # For the 3 differential equations and 3 initial conditions, the
          RK4 method uses the below recursion formula to evaluate.
          x[k+1] = x[k] + (dt*(k1 + 2*k2 + 2*k3 + k4) / 6)
          y[k+1] = y[k] + (dt*(l1 + 2*l2 + 2*l3 + l4) / 6)
          z[k+1] = z[k] + (dt*(m1 + 2*m2 + 2*m3 + m4) / 6)
return z
```

Code

```
import matplotlib.image as img

import matplotlib.pyplot as plt

import numpy as np

from PIL import Image


# Function for showing the image

def ShowImage(image,heading):

    plt.imshow(image)

    plt.title(heading)

    plt.show()



# Function providing the Dimensions of the image

def ProvideDimension(image):

    return image.shape[0], image.shape[1]



# Function for defining the Lorenz Equations

def LorenzEquations(x, y, z):

    sigma = 10.0

    beta = 8.0/3.0

    rho = 28.0

    dx = (sigma*(y - x))

    dy = (rho*x - y - x*z)

    dz = (-1*(beta*z) + x*y)

    return dx, dy, dz



# Defining the Runge kutta Method

def Rungekutta(x0, y0, z0, n):


    x = np.zeros(n+1)

    y = np.zeros(n+1)

    z = np.zeros(n+1)


    x[0] = x0

    y[0] = y0

    z[0] = z0


    T = 25

    dt = T/float(n)


    for k in range(n):

        k1, l1, m1 = LorenzEquations(x[k], y[k], z[k])

         k2, l2, m2 = LorenzEquations(x[k] + 0.5*k1*dt, y[k] + 0.5*l1*dt,
z[k] + 0.5*m1*dt)
```

```
         k3, l3, m3 = LorenzEquations(x[k] + 0.5*k2*dt, y[k] + 0.5*l2*dt, 
z[k] + 0.5*m2*dt)

          k4, l4, m4 = LorenzEquations(x[k] + k3*dt, y[k] + l3*dt, z[k] + 
m3*dt)


        x[k+1] = x[k] + (dt*(k1 + 2*k2 + 2*k3 + k4) / 6)

        y[k+1] = y[k] + (dt*(l1 + 2*l2 + 2*l3 + l4) / 6)

        z[k+1] = z[k] + (dt*(m1 + 2*m2 + 2*m3 + m4) / 6)

    return z



# Taking the image as input

picture = img.imread('GroupPhoto.jpg')

ShowImage(picture,"Actual Image")


# Taking the dimesions of the image using the function

h, w = ProvideDimension(picture)



# Using the key for each pixel of the image

keys = Rungekutta(0.08342, 0.0696969, 0.442056, h*w)


# Doing Encryption using the key and the given image

Encryption = np.zeros(shape=[h, w, 3], dtype=np.uint8)

Index = 0
```

```
for i in range(h):

    for j in range(w):

        keyIndex = (int((keys[Index]*pow(10, 5)) % 256))

        Encryption[i, j] = picture[i, j] ^ keyIndex

        Index += 1

ShowImage(Encryption,"Encrypted Image")


# Saving the Encrypted image

im = Image.fromarray(Encryption)

im.save('Encrypted.jpg')




# Proper Decryption using actual key

Decryption = np.zeros(shape=[h, w, 3], dtype=np.uint8)

Index = 0

for i in range(h):

    for j in range(w):

        keyIndex = (int((keys[Index]*pow(10, 5)) % 256))

        Decryption[i, j] = Encryption[i, j] ^ keyIndex

        Index += 1

ShowImage(Decryption,'Decrypted Image with Actual key')




# Improper Decryption using different key

keys1 = Rungekutta(0.1, 0.2, 0.3, h*w)
```

```
Decryption = np.zeros(shape=[h, w, 3], dtype=np.uint8)

Index = 0

for i in range(h):

    for j in range(w):

        keyIndex = (int((keys1[Index]*pow(10, 5)) % 256))

        Decryption[i, j] = Encryption[i, j] ^ keyIndex

        Index += 1

ShowImage(Decryption,"Decrypted Image with incorrect key")
```

The GitHub Repository can be found in: https://github.com/Madhav-Kanda/ImageEncryption.git

Security Analysis

The Security of an encryption method is the most important part. This is because in the modern world the availability of high power driven supercomputers with enormous computing power has made decryption of ciphers very easy. To protect the data from vulnerabilities, it is important to store data in a secure manner. The image encryption method uses Lorenz equation to generate chaotic keys which are deterministic but slight change in initial values will produce a completely different set of encryption keys.

The high security of this image encryption method can easily be realized by manipulating the key values slightly. This produces a new set of keys without any similarity to the previous set. When decryption is done using these new sets of keys, no useful outputs are produced except noise.

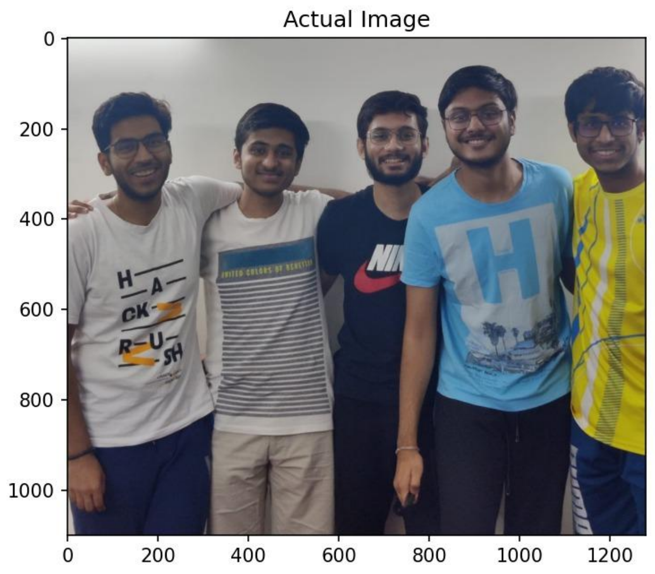
This shows the need of exact same initial values for decryption and high security level of Image encryption using chaotic keys.

Future Work

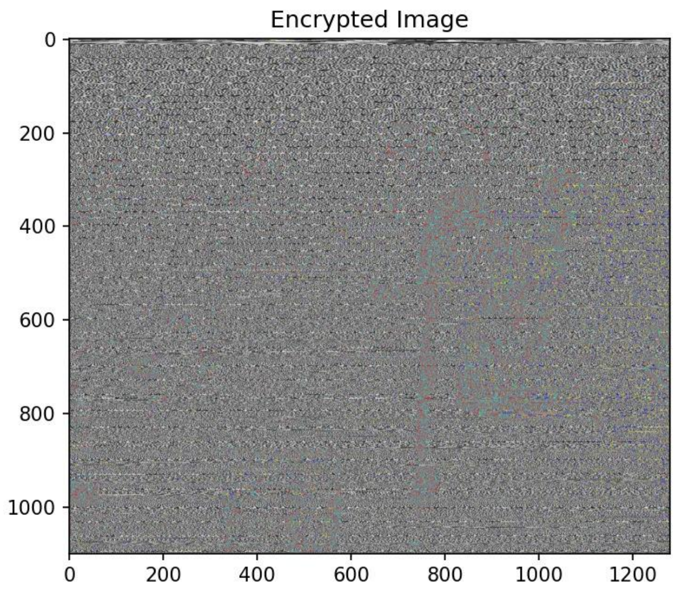
1. We can implement a python based GUI (Graphical User interface), to encrypt or decrypt any image.
2. Currently the algorithm only supports .jpg format image encryption and decryption but we can also scale it to encrypt text and other image formats.
3. Increasing level of encryption and security by shuffling of image. This is done by a method called Confusion and diffusion.
4. Explore other Numerical methods to increase the efficiency of encryption.

Observations

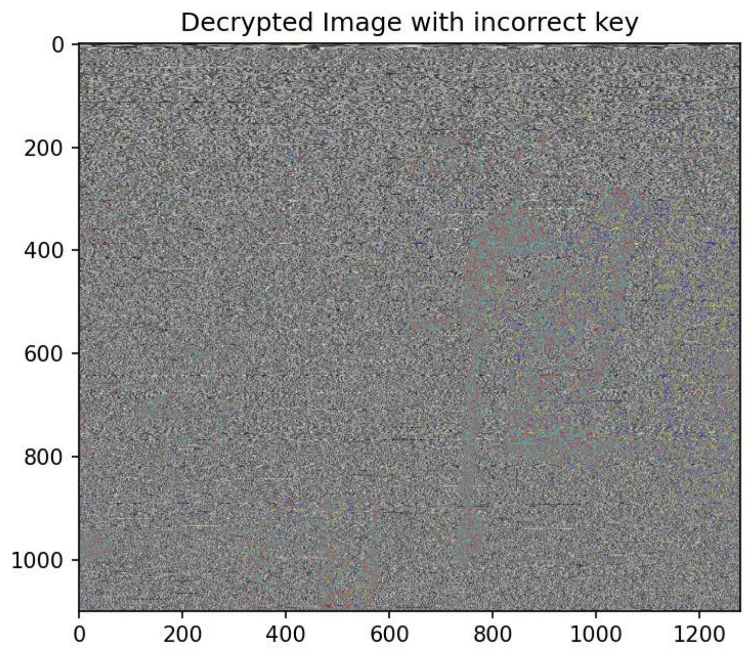
The actual image is taken in jpg format and read by python with help of the library in matplotlib.



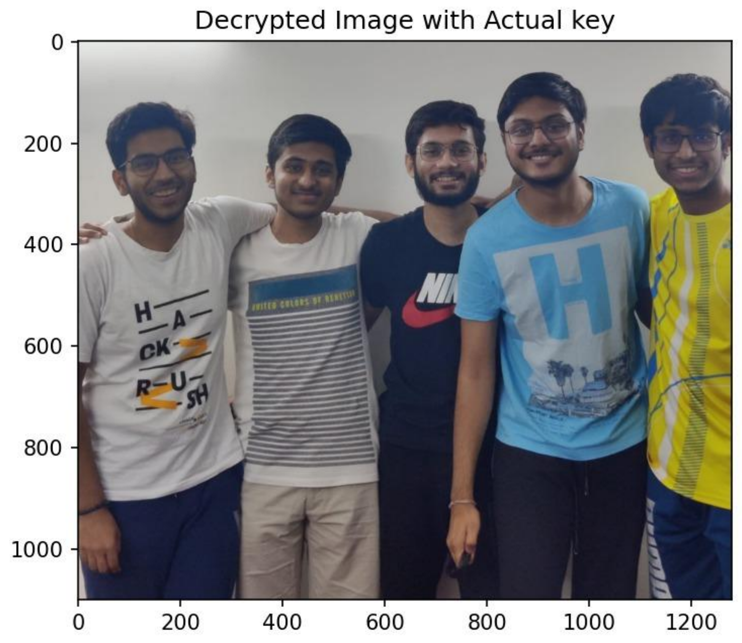
The original image is encrypted by keys (0.08342, 0.0696969, 0.442056).



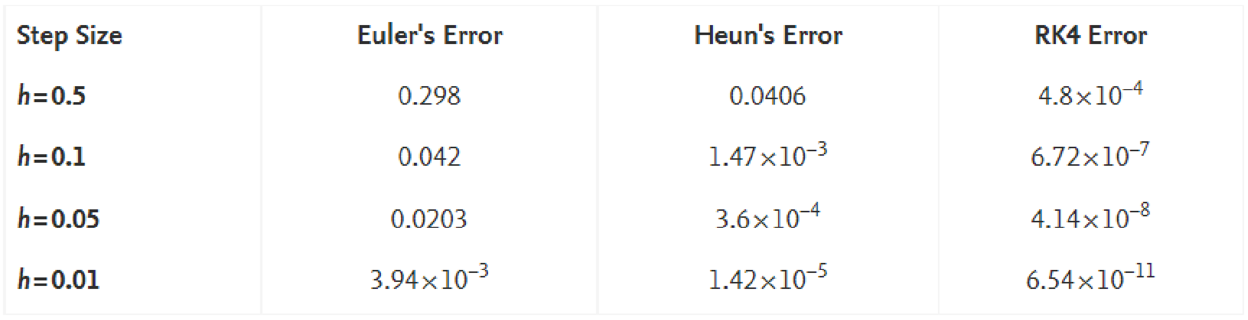
Decrypting an image with wrong key results in image with noise.



This image is decrypted with the original key. This provides the actual image without any noise.



The step size is important as its value reduces the error while calculating from the RK4 method.



Step Size	**Euler's Error**	**Heun's Error**	**RK4 Error**
$h=0.5$	0.298	0.0406	4.8×10^{-4}
$h=0.1$	0.042	1.47×10^{-3}	6.72×10^{-7}
$h=0.05$	0.0203	3.6×10^{-4}	4.14×10^{-8}
$h=0.01$	3.94×10^{-3}	1.42×10^{-5}	6.54×10^{-11}

Table 1. Global Error committed by RK4 Method. src

Conclusion

Numerical methods can be used for various real-life applications. One such application is the use of encryption. The combination of Chaotic system of equations like the Lorenz equation and a numerical method like the Runge-Kutta method can be used to deploy a robust encryption system. While Chaotic systems make the system unpredictable, the numerical method makes deterministic keys that can only be generated using the same initial values - hence making it a dependable algorithm.

References

1. https://www.sciencedirect.com/science/article/pii/S004579061500364X
2. https://hipc.org/hipc2012/documents/SRSPapers/Paper%2046.pdf
3. https://ieeexplore.ieee.org/document/7399173
4. https://www.researchgate.net/publication/319275465_Image_encryption_algorithm_based_on_Lorenz_chaotic_map_with_dynamic_secret_keys