Image Steganography with Arnold's Cat Mapping and Chaos-Based Cryptography

A project report submitted in partial fulfillment

of the requirements for the degree of

Bachelor of Technology

in

Electronics & Communication Engineering

by

TUSHAR CHOPRA

21BEC1069

School of Electronics Engineering,

Vellore Institute of Technology, Chennai, Vandalur-

Kelambakkam Road, Chennai - 600127, India.

November 2024

Declaration

I hereby declare that the report titled *Image Steganography with Arnold's Cat Mapping and Chaos-Based Cryptography* submitted by me to the School of Electronics Engineering, Vellore Institute of Technology, Chennai in partial fulfilment of the requirements for the award of **Bachelor of Technology** in **Electronics and Communication Engineering** is a bona-fide record of the work carried out by me under the supervision of *Dr.SUBHASHINI N*.

I further declare that the work reported in this report, has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or University.

Sign:

Name & Reg. No.:

Date:

School of Electronics Engineering

Certificate

This is to certify that the project report titled *Title* submitted by *Tushar Chopra***(21BEC1069)** to Vellore Institute of Technology Chennai, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Electronics and Communication Engineering** is a bona-fide work carried out under my supervision. The project report fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Supervisor		**Head of the Department**	
Signature:	....	Signature:	....
Name:	....	Name:	....
Date:		Date:	

Examiner

Signature: ....

Name: ....

Date:

(Seal of the School)

Abstract

The exponential growth of digital communication has raised concerns about data security and privacy. Image steganography, which embeds hidden data in digital images, provides a reliable mechanism for secure data transmission. In this paper, we propose an improved image steganography system built on chaos-based cryptography for enhanced security. Chaos theory is actually an ideal choice for cryptographic applications, as it is characterized by extreme sensitivity to initial conditions and deterministic unpredictability. The proposed system combines chaos-based cryptography and steganography into a two-layer security mechanism. At the first level, Chaotic Card encrypts the secret data into an incomprehensible format that is much harder to extract without authorization. At the second level, the encrypted data is embedded into the digital image using LSB substitution or other adaptive techniques to minimize distortion of the cover image and maintain undetectableness. We analyze the system performance by determining key system performance metrics such as PSNR, MSE, and embedding capacity. We also analyze how the system responds to steganography attacks such as noise injection, clipping, and compression. Experimental results are presented that show that adding this chaos-based encryption significantly improves the security and reliability of the steganography system without degrading the quality of the stego image. The proposed method introduces unpredictability and complexity based on chaos cryptography to overcome the shortcomings of traditional steganography methods. The applications of the proposed method include secure communication, digital watermarking, and copyright protection. This paper combines the advantages of chaos theory and image steganography to contribute to a more secure and efficient framework for protecting sensitive information

Acknowledgements

It is my pleasure to express with deep sense of gratitude to Dr.SUBHASHINI N, Assistant Professor Senior, School of Electronics Engineering, Vellore Institute of Technology, Chennai, for her constant guidance, continual encouragement, understanding; more than all, she taught me patience in my endeavor. My association with her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Computer Vision, Image steganography.

It is with gratitude that I would like to extend my thanks to the visionary leader Dr. G. Viswanathan our Honorable Chancellor, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Dr.G V Selvam Vice Presidents, Dr. Sandhya Pentareddy, Executive Director, Ms. Kadhambari S.Viswanathan, Assistant Vice-President, Dr. V. S. Kanchana Bhaaskaran Vice-Chancellor, Dr. T. Thyagarajan Pro-Vice Chancellor, VIT Chennai and Dr. P. K. Manoharan, Additional Registrar for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dr. Ravi Sankar A, Dean, Dr. Reena Monica P, Associate Dean Academics, Dr. John Sahaya Rani Alex, Associate Dean Research, School of Electronics Engineering, Vellore Institute of Technology, Chennai for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant state, I express ingeniously my whole-hearted thanks to Dr. Mohanaprasad K., Head of the Department, Bachelor of Technology in Electronics and Communications and the Project Coordinators for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staffs at Vellore Institute of Technology, Chennai who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Table of Contents

LIST OF FIGURES

Chapter 1

Introduction

1.1 OVERVIEW

In this era of digital communication, ensuring confidentiality and integrity has become a very crucial issue. Image steganography is, in fact, a technique of hiding information within digital images which happens to be a secret form of communication. However, conventional steganography techniques are not reliable enough and often susceptible to steganographic attacks. These also require sophisticated mechanisms to increase security and reliability. This paper studies a complex framework that synergistically combines chaos-based encryption and image steganography by using the deterministic randomness of chaos theory to enhance the security of hidden information. In the proposed methodology, a chaotic system with topological transitivity is adopted as an encryption mechanism in the chaos-based encryption process, which is highly sensitive to initial conditions. The plaintext secret data is first mapped, using the chaos map, to an encrypted form with pseudo-randomness and statistical complexity. This encrypted form is then embedded into the cover image by the means of adaptive steganography techniques, such as optimized LSB substitution or transform domain insertion, ensuring undetectableness and minimization of the statistical anomalies that may point at a hidden message. We critically evaluate the accuracy of the stego image by using quantitative metrics including peak signal-to-noise ratio, structural similarity index score, and mean square error. We also carefully study the robustness of the system regarding different attacks - noise addition, compression, pruning, histogram analysis, machine learning classification, and so forth. The results present the fact that the chaos-enhanced steganography achieves excellent stealth, enhanced embedding capability, and enhanced resistance to unauthorized detection and extraction. This work solves severe vulnerabilities of traditional methods by combining the unpredictability of chaos theory with powerful data hiding strategies and contributes toward the development of secure communication systems. It has been applied in various fields such as digital watermarking, copyright protection, and confidential information sharing, and has become a leading information security solution.

1.2BACKGROUND

Some background steganography techniques exist. Therefore, standard properties of good steganography have been developed [9]. The five properties of good steganography are: i) Vandal resistance: This means that the steganography method ought to be resistant to algorithmic hacking by steganography tools [10] used to break the steganography method. ii) Robustness: The strength for resisting the conditions must ensure that the algorithm goes through all the expected tasks like image resizing, cropping of the image, or compression of images. iii) Invisibility: No doubt should be implied by a suspicion that the message is in a stego object. iv) Capacity: The payload size of the hidden message should be larger and not affect the other properties. The large size that can be hidden in a single stego object means that a small number of stego objects are needed to hide an entire message. v) Stealth: There should be little or no difference between an object with no hidden message and an object with a hidden message. A digital signature[20] generates a unique electronic signature that can be used to sign as well as verify the authenticity and integrity of digital files while also supporting the notion of non-repudiation. Digital signatures rely on asymmetric cryptography, which employs two different keys-one for signing purposes, and another one to verify the signature and is kept confidential from the recipient so that no one else but the recipient can generate the same signature without that key. Some of the digital signature algorithms are DSA(Digital Signature Algorithm), RSA(Rivest Shamir Aldeman)[21], ECDSA(Elliptic Curve Digital Signature

Algorithm)[22], EdDSA(Edward Curve Digital Signature Algorithm) and Signature Scheme ElGama. EdDSA[23] is a curve-based, asymmetrical cryptographic algorithm. EdDSA signature makes use of Edwards-shaped elliptic curves such as edwards25519 and edwards448, respectively. The SHA-512 is the internal hash function of Ed25519 and SHAKE256 is used as the internal hash function of Ed448. Ed25519 has a prime number: p = 1 (mod 4) Ed448 has a prime number: p = 3 (mod 4) EdDSA can provide an extremely high security level with a relatively short length of the key compared to other digital signature algorithms[24]. EdDSA provides non-repudiation of sender, and even a third party or MITM receiving the stego message can not build up a signature that equals the original embedded message. The Advanced Encryption Standard (AES)[25] is a symmetric encryption algorithm that uses a single key to encrypt and decrypt data. AES is a block encryption algorithm that operates on data of size 128 bits or 16 bytes. It is considered the most secure symmetric encryption algorithm with various encryption key sizes ranging from 128, 192, and 256 bits. The number of rounds used for 128-bit keys is 10, and for 192-bit keys, it uses 12 rounds, while it takes 14 rounds for 256-bit keys[26]. The stronger the algorithm becomes, the higher the key size. Nevertheless, it will take years for a supercomputer to break the algorithm in a brute-force attack.

1.3 Past enquiries

Many researchers have done great works to improve data security in steganography images. Some approached it with cryptographic algorithms for the protection of data privacy and integrity, while some enhanced the steganographic method to make hard-to-detect hidden messages. Therefore, it is very important to include a critical review of such related works as can be shown below.

B. Karthikeyan et al. [27] put forward the integration of RSA algorithm with the technique of image steganography. From the study, they could perceive effective improvement in data security but came up with the alarming drawbacks such as the lengthy encryption and decryption time of processes and poor payload capability.

Similarly, V. Kalaichelvi et al. [28] proposed combining cryptography and steganography to enhance security in image steganography. In their implementation, they employed the modified Hill cipher algorithm to encrypt the embedded message. However, recent research works note down weaknesses of the modified Hill cipher, mainly because it has linear dependence with respect to matrix operations, which makes it vulnerable to attacks using known plaintext.

M. Kataria et al. in [29] presented recent steganographic methods for safe image communication using four encryption methods: AES, DES, RSA, and ChaCha20. LSB and Spread Spectrum technique is implemented in this paper to embed ciphertext in a number of images and to extract the original text back from the images. Experiments revealed that DES has lesser performance compared to AES and ChaCha20. Meanwhile, RSA required higher execution times for encryption and decryption processes than elliptic curve algorithms.

S. Bhargava et al. [30] conducted the message protection through AES encryption of the data in the LSB steganography. They stated that if the man-in-the-middle (MITM) attacks compromised it, AES encryption would still protect the message. However, their method did not ensure message integrity or non-repudiation from the sender because their application is based on a single key maintained by both parties and, hence, constituted a security threat due to the private key protection required for both parties.

D. Kumbhakar et al. [31] argued to protect the data of e-commerce using Elgamal encryption with LSB steganography. Using MSE, PSNR, and SSIM as evaluating metrics, the result shows that the data of e-commerce could be protected successfully.
A. Bahaddad [32] proposed chaotic encryption to safeguard the hidden information against MITM attacks. The proposed method efficiently protected the data, yet problems encountered in increasing its

speed and security were found in it with chaotic encryption techniques.

Despite these efforts to defend MITM attacks based on the algorithms implemented in secret message encoding, several researchers have continued touting RSA as a digital signature algorithm. However, the inefficiency in RSA considering long keys and slow computation has instead exposed recent improvements that have been made in digital signature algorithms. These new algorithms offer similar or superior security and shorter key sizes and quicker signing/verification operations. Unlike previous methods, this research introduces a new idea that combines encryption techniques with digital signature techniques to enhance security verification.

1.4 Cryptography

Cryptography, describing the evolution from basic paper-and-pen techniques to highly specialized machines and advanced arithmetic operations, possesses a rich and deep history. Though wide-ranging is the complete scope of cryptography, this research paper offers a concise analysis based on the importance of secure information transmission.

Cryptology

Derived from the Greek word meaning "secret" or "hidden," cryptography involves the study of creating and deciphering secret messages by using codes. Mainly, the area it deals with can be narrowed down to two important parts:

Cryptography:

Development and deployment of message encoding and decoding schemes.
Cryptanalysis: The science of breaking or analyzing cryptographic systems.
"Cryptography" generally refers to a family of security practices, which consist of:

Encryption and decryption algorithms
Integrity-checking techniques and
Digital signatures

1.6 Cryptography using a Secret Key

Secret-key cryptography, or symmetric-key cryptography, depends on employing the same key for enciphering and deciphering. There's a great example of such a system in the Data Encryption Standard, or DES, which has been deployed in large quantities by the Federal Government.

Figure 3 provides the major steps for secret-key cryptography applied for secret communication. Advantages of this method: this kind of encryption method protects data very well, but it also has an important disadvantage-namely, key distribution should be carried out securely over a secure communication link between the sender and recipient. If a private key is compromised to an unauthorized person, they can decrypt all the encrypted data, hence key security is an important challenge in this strategy..

Chapter 2

Related Works

2.1 OVERVIEW

This field of image steganography continues to improve with ongoing research into new embedding techniques, improved security, and increased resistance to detection methods: steganalysis. Popular among the methods include traditional Least Significant Bit (LSB) substitution techniques since they are simple and low in computational cost. Such methods are largely vulnerable to statistical and visual attacks that have lead to creation of sophisticated methods lately.

Recently, researches of transform domain techniques such as the Discrete Wavelet Transform (DWT), the Discrete Cosine Transform (DCT), and even Singular Value Decomposition (SVD) caught the attention of researchers. Such techniques embed data in the frequency components of images, which are considered to exhibit improved resistance against compression and noise. Not surprisingly, this incurs its own growth in computational complexity. Other adaptive steganographic techniques that also concern the local characteristics of an image started being widely used. These improve invisibility and make detection by analysis tools much more difficult as they embed data in textured or noisy regions.

There has been a significant focus recently on combining steganography with cryptographic methods to improve data security. Indeed, in such approaches, encryption algorithms, including but not limited to Advanced Encryption Standard (AES), Rivest-Shamir-Adleman (RSA), are often combined with the steganographic techniques in order to ensure confidentiality of the hidden information. The inspiration behind chaos-based cryptography from chaos theory is very promising, with a resultant randomness and sensitivity toward the initial conditions. This gives chaotic maps, such as the Logistic Map, Lorenz Attractor, and Henon Map, full of a lot of randomness and complexity, thus offering great resistance toward cryptanalysis.

Contrarily, this has led to making better steganalysis tools that employ machine learning and deep learning models in the detection of concealed data with much more accuracy. As such, the present day techniques of steganography need to be able to evade the sophisticated detection systems in the market.

Although robustness and security in the field have gained revolutionary improvements in the area of steganography, there is still room for just discovering the perfect balance between invisibility, capacity, and computational efficiency. Leverage such progresses this paper is aimed at integrating chaos-based encryption with an adaptive steganography method to overcome the deficiencies of the existing methods and further expand the horizons of safe data hiding.

2.2 Image Steganography Using LSB and Hybrid Encryption Algorithms

Symmetric encryption is probably one of the oldest and most anciently known methods used to secure data. It uses only one secret key and can be presented in an alphanumeric word or a random character series for use both to encrypt and decrypt messages. This method works on converting each character of the text into another set of characters by using the secret key. As both parties are having a shared key, they can safely communicate, but the problem is in the sharing of the secret key because unauthorized access to this secret key may decrypt the data.

Blowfish

Blowfish is one of the symmetric encryption techniques, which is a block cipher. It is developed by Bruce Schneier. A Blowfish encryption technique supports keys from size 3 up to 448 bits and will work on blocks of 64 bits. Blowfish is an open-source, free software with no security threats since it does not easily break under cryptanalysis. AES has some advantages, but most of the time, the algorithm preferred is AES just because it is simple and good strength. AES always uses a 128-bit encryption key, making it highly resistant to attacks. Unlike 3DES, AES encrypts blocks of data with a different size each time, making it unpredictable, thus boosting the security. This makes AES a very practical choice for systems requiring secure data transmission without the complexities of asymmetric encryption.

Text steganography, another method of safe communication, includes the position of information inside text files. The most common technique places the message in the header; this does not harm the original content but increases the file size. Alternatively, bits of the text can be modified to hide information; this technique, however, is less popular because it will typically make the text meaningless. Techniques of text steganography can also be adapted for use with other formats, such as PDFs, making them yet more available.

In image steganography, there are methods, such as Least Significant Bit (LSB), where a secret message is embedded as bits in least significant bits of a cover image. Such methods are easy and efficient but can be improved. For example, one of the variants of SLSB is to improve the security of embedding data within just one color channel per pixel so as to make it improbable to be detected. The quality of stego images is often measured based on metrics such as Peak Signal-to-Noise Ratio (PSNR). A greater PSNR indicates that the stego image closely resembles the cover image in visual terms.

The process of determining hidden data is referred to as steganalysis. This includes visual and statistical attacks to detect anomalies within images and their corresponding binary representations or distributions in frequency. Statistical attacks may be faster and stronger since techniques, such as histogram analysis or standard deviations, can be used for the calculations.

The recent developments in steganography concentrate on improving imperceptibility, capacity, and robustness. It can be said that methods combining LSB with AES encryption display better resistance to attack, with assurance to both security and high-quality cover images. Hybrid techniques of combining LSB with RSA or even Caesar Cipher encryption are obtained and further enhance confidentiality. In addition to 2D images, 3D models have emerged as hosts of covert data, leveraging the complexity and versatility for secure communication.

In conclusion, this survey paper deals with some of the steganographic techniques, mentioning their potential strength and limitations and their scope for development. Combining solid techniques like LSB and AES along with innovative approaches should provide solutions to the above issues and elevate the field of secure data hiding. One among the oldest known data safety techniques is symmetric encryption. It uses a private key that can be any text string or a random combination of characters; it supports encryption and decryption. The process will be conversion of each character in the plaintext using the secret key to form another set of characters such that both sender and recipient have the same secret key, hence there won't be any problem with secure communication. However, the problem is that, since the secret key needs to be distributed safely otherwise some unauthorized access may breach data,

Blowfish is one of the most popular symmetric encryption algorithms, which falls into the block cipher category. Designed by Bruce Schneier, it has keys between 3 and 448 bits and 64-bit block. Blowfish is open source, free to use and thereby secure with resistance to cryptanalysis. Despite its advantages, AES (Advanced Encryption Standard) is often preferred for its simplicity and robust security features. AES uses a 128-bit encryption key, making it highly resistant to attacks. Unlike earlier methods like 3DES, AES encrypts data blocks with different coding types and sizes each time, ensuring unpredictability and enhanced security. This makes AES a practical choice for systems requiring secure data transmission without the complexity of asymmetric encryption.

Text steganography, another form of secure communication, involves hiding information within text files. One approach embeds the message in the file’s header, preserving the original content but increasing the file size. Alternatively, bits of the text can be altered to hide data, though this method is less common as it often makes the text unreadable. Text steganography techniques can also be applied to formats like PDFs, further expanding their usability.

The Least Significant Bit method in image steganography embeds the secret message bits directly into the least significant bits of a cover image. It is easy and efficient, but it has room for improvement. For instance, the Selected Least Significant Bit variation appears to be a more secure method as data is inserted in only one color channel for every pixel instead of three. Thus, it reduces the chances of being discovered. Quality metrics such as Peak Signal-to-Noise Ratio (PSNR) are used to determine the quality of stego images. The better the visual similarity between a stego image and its corresponding original cover image, the higher the PSNR value.

Steganalysis, or the detection of hidden data, includes two approaches: visual and statistical attacks. The former analyze binary representations of images containing anomalies, while the latter focuses on frequency distributions to determine any irregularities. Methods in statistical attack include histogram analysis or the computation of standard deviation, which assist in speeding and effectiveness.

New concepts in steganography mainly developed the features of invisibility, capacity, and robustness. Strong combinations of techniques like LSB and AES encryption have shown improved performance when it comes to resisting attacks, security with good quality cover images can be ensured. Hybrids of more techniques like the integration of LSB with RSA, Caesar Cipher, etc., enhance confidentiality. Besides 2D images, 3D models have emerged as hosts for covert data communicating securely using the complexity of 3D models and their versatility.

This study surveys various steganographic methods that describe their merits as well as drawbacks with some scope of improvement. Integration of the established approach like LSB and AES with innovative methodology can help overcome the extant shortcomings of the technique of secure data hiding.

Chapter 3
Methodology

3.1 OVERVIEW

This chapter is an in-depth guide on a step-by-step basis about using chaotic mapping techniques towards the development of a secure system for image encryption and decryption. The technology ensures secure communication by reversibly and extremely securely changing the places of pixels. The four major phases of the approach are preprocessing, encryption, decryption, and performance evaluation

3.1.1 Image Preprocessing

Preprocessing proves to be vital in ensuring compatibility, and that the image is prepared for effective encryption.

- **Image Standardization**

To perform efficient matrix operations, input images are resized to a square size, for instance, N×N. For compatibility with formats like JPEG, PNG, and TIFF, sizes 512×512 pixels are generally used.

- **Grayscale Conversion**

Conversion of colour images to greyscales focuses on one channel for intensity while reducing the complexity in computation. This stage is currently applied in both military communication systems and biometric systems.

- **Noise Removal**

Filters are employed to eliminate noises and artifacts for reducing interference in transformations and providing a better basis for pixel scrambling.

- **Normalization**

Normalized values of particle intensity in 8-bit image ranges between 0 and 255. This maintains equal treatment and avoids the likelihood of numerical overflow during encrryption.

3.1.2 Encryption Process

The chaotic mapping-based encryption technique makes the image completely incomprehensible due to repetitive rearrangement of pixels.

- **Parameter Initialization**

A few parameters are determined. For example, k-number of iterations and picture matrix size (N). As elements of the key, these parameters provide safe and reliable encryption.

- **Matrix-Based Pixel Transformation**
    - Each pixel is transformed through a chaotic mapping matrix, (x,y)(x, y)(x,y):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \mod N$$

        - Pre-defined constants are; a,b,c, b,a, c, and dddd, while NNN is the size of the matrix. The transformation yields chaos without affecting the integrity of the pixel.
- **Iterative Transformation**

Applying chaotic transformation repeatedly increases the degree of randomness at each step. While they require a very high degree of synchronisation in decryption, higher values of iteration are more secure.

- **Output**

Finally, the jumbled image is saved in PNG or JPEG formats. PNG is safe but also for only optimal balance between storage efficiency and immunity to compression.

3.1.3 Decryption Process

Using the same parameters that encryption used, decryption obtains the original image by reversing pixel scrambling.

- **Parameter Validation**

Validation of the parameters (transformation matrix, kkk) is the first stage in decryption. The parameters are wrongly entered will fail decryption to work.

- **Inverse Transformation**
    - Reversing the jumbled pixels back to where they originally were involves the inverse mapping. This process reverses the disorderly rearrangements.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} \mod N$$

- **Reverse Iterations**

    - It is the inverse transformation applied multiple times iteratively which progressively return the image into its original form.
- **Validation**
    - Qualitative metrics in addition to the PSNR, and also visual inspection will be used to affirm that the decrypted image is intact.

3.1.4 Performance Evaluation

Parameters upon which the performance and the security of the system are based are:

- **Entropy Analysis**
    - This shows the randomness of the encrypted image. Entropy values close to 8 for 8-bit images close to 8 for 8-bit images depicts the strong encryption.
- **Correlation Coefficients**
    - Horizontal, vertical as well as diagonal correlations of the encrypted image are analysed. Low correlations validate that proper scattering is in place.
- **PSNR (Peak Signal-to-Noise Ratio)**
    - This verifies how close the original and decrypted images are. High PSNR values represent minimal distortion.
- **Key Sensitivity**
    - This tests the security by introducing slight alterations in decryption key. Good crypto scheme mustn't decrypt with inappropriate keys.
- **Execution Time**
    - The encryption and decryption times are measured across different resolution images to assess the efficiency and scalability of the algorithm.

3.1.5 Implementation Details

- **Environment Setup**
    - This implementation is written in MATLAB 2017b using this language's matrix computation and image processing capability. Standard image databases like SIPI were used to acquire test images.
- **Optimization**
    - Vectorization and optimization of loop structures for performance on large images.
- **Validation**
    - The results are recorded for different resolutions and formats across metrics like entropy, PSNR, and key sensitivity to ensure a sound system for encryption.

It ensures that it develops a secure, scalable, and efficient image encryption framework in the face of overcoming the critical challenges of modern secure communication.

Chapter 4
Evaluation Metrices

Steganography algorithms are evaluated along three axes: the amount of data that can be hidden in an image, a.k.a capacity, the similarity between the cover and steganography image, a.k.a distortion, and the ability to avoid detection by steganalysis tools, a.k.a secrecy. This section describes some metrics for evaluating the performance of our model along these axes. Reed Solomon Bits Per Pixel: Measuring the effective number of bits that can be conveyed per pixel is non-trivial in our setup since the ability to recover a hidden bit is heavily dependent on the model and the cover image, as well as the message itself. To model this situation, suppose that a given model incorrectly decodes a bit with probability p. It is tempting to just multiply the number of bits in the data tensor by the accuracy $1 - p$ and report that value as the relative payload. Unfortunately, that value is actually meaningless – it allows you to estimate the number of bits that have been correctly decoded, but does not provide a mechanism for recovering from errors or even identifying which bits are correct. Therefore, to get an accurate estimate of the relative payload of our technique, we turn to Reed-Solomon codes. ReedSolomon error-correcting codes are a subset of linear block codes which offer the following guarantee: Given a message of length k, the code can generate a message of length n where $n \geq k$ such that it can recover from $n-k$ /2 errors (Reed & Solomon, 1960).

Steganography algorithms are frequently tested for three criteria: capacity is the ability to embed data in an image and how many data values can be put inside it; distortion refers to how similar the steganographic image is to the nonsteganographic original; and secrecy measures how well the steganographic image can avoid detection by steganalysis tools. This chapter gives metrics for testing our model on these three-criteria parameters.

Reed-Solomon Bits Per Pixel

Measuring how many bits can effectively be hidden per pixel in our setup isn't straightforward. The ability to recover hidden bits is heavily determined by various factors, including the model, the cover image, and the message itself. For instance, suppose that our model has a probability p

p of getting the bit wrong. It might seem like the relative payload is just the total number of bits in the data times the accuracy $(1-p)$. But this works out wrong-it just gives you the number of correctly decoded bits but it doesn't help you find the errors or indicate how to correct them.

To do this we use Reed-Solomon error-correcting codes, which are linear block codes. These guarantee that, for a message of length "k", you are able to produce a longer message of length 'n' (where "n" ≥ "k"), and correct up to "(n "−' 'k") / 2 errors. This means that one can pretty reasonably estimate the relative payload, taking into account both errors and the possibility of recovering from them (Reed and Solomon, 1960).

This implies that given a steganography algorithm which, on average, returns an incorrect bit with probability p, we would want the number of incorrect bits to be less than or equal to the number of bits we can correct:

$$p \cdot n \leq \frac{n - k}{2}$$

The ratio ???? ???? k/n represents the proportion of "real" data that can be transmitted for every bit of "message" data. According to equation (13), this ratio is always less than or equal to 1 − 2???? 1−2p, where ????

Let p be the probability of decoding an incorrect bit. By this ratio, we can now find the relative payload of our steganographic method by multiplying the number of bits intended to be hidden per pixel times that ratio. This will now give us the actual number of bits that will be successfully transmitted and recoverable.

We call this measure Reed-Solomon Bits Per Pixel (RSBPP). It provides a useful benchmark for comparison against other conventional steganographic approaches, since the value can be measured directly. RSBPP essentially captures the average number of bits that can be reliably hidden and extracted from an image, normalized by the size of the image.

Peak Signal-to-Noise Ratio (PSNR)

Besides measuring the comparative payload, there is an equivalent need to assess the quality of the steganographic image itself. One of the most prevalent metrics used in this regard is the Peak Signal-to-Noise Ratio (PSNR). PSNR measures the distortion between the original and the stego image and is shown to closely relate to human assessment of the image quality through mean opinion scores of human experts (Wang et al., 2004).

Let us consider two images

X and

Y of the dimensions
W×H, and a scaling factor
such that it captures the maximum possible difference between the pixel values, PSNR is therefore calculated as follows; using the Mean Squared Error MSE between images.

$$\mathrm{MSE} = \frac{1}{W \cdot H} \sum_{i=1}^{W} \sum_{j=1}^{H} (X_{i,j} - Y_{i,j})^2$$

$$\mathrm{PSNR} = 20 \cdot \log_{10}(\mathrm{sc}) - 10 \cdot \log_{10}(\mathrm{MSE})$$

Although one of the most popular metrics for distortion measured by steganography algorithms, Almohammad and Ghinea (2010) argue that PSNR is not a good option if comparing different steganographic techniques. For this, we have used another metric to rate image quality. We used the SSIM, which stands for the Structural Similarity Index.
Structural Similarity Index (SSIM)
In our experiments, we also compute the SSIM between the original (cover) image and the steganographic image. SSIM is a universally accepted metric used by the broadcast industry in the assessment of quality for images and videos (Wang et al., 2004). While PSNR computes pixel-by-pixel differences, SSIM accounts for changes in structural information, which is closer to the human way of processing image quality.

To calculate SSIM for two images XXX and YYY, the following components are used:

- μX\mu_XμX and μY\mu_YμY: the mean values of XXX and YYY,
- σX2\sigma_X^2σX2 and σY2\sigma_Y^2σY2: the variances of XXX and YYY,
- σXY\sigma_{XY}σXY: the covariance between XXX and YYY.

SSIM integrates these measures into a single index. It offers a more holistic evaluation of image quality, as opposed to the PSNR approach, and accordingly, its incorporation with PSNR will guarantee a much stronger and comprehensive assessment for the quality of the steganographic image.

CHAPTER 5
EXPERIMENTAL RESULTS

Using a sample image, say, dog, the following theoretical and experimental results were observed for various techniques of image encryption and transformation:

5.1 Original Image Analysis:

The original image shows a structured organization of pixel values that represents its content visually.In standard RGB images, the red, green, and blue channels all have separate distinct intensity peaks in their histograms that show the image's dominating colors. The peaks reflect the tonal range and the color composition of the image.

5.2 Logistic Encryption

The logistic encryption algorithm disturbs pixel values while including randomness through chaotic nonlinear dynamic mapping.

The resulting image is then observed as noise with no apparent patterns or structures and obscures the original pixel values.

All color channels histogram of intensity will be almost uniform. This means pixel values get randomized, and there are no predictable correlations among pixels.

This represents the effective encryption since the original image content was hidden.

5.3 Arnold Cat Map Transformation

The Arnold Cat Map permutation introduces further randomness into the structure of the image in the spatial scrambling of pixel positions.

Following successive applications, the image grows highly noisy, and any similarity to the original becomes unidentifiable.

Though the histograms of the transformed image appear fairly uniform in the initial iterations, the spatial distribution of the pixel values is entirely scrambled, which greatly increases image obscurity and decryption efficiency.

5.4 Logistic Decryption

Decryption is the reverse logistic encryption process using a decryption key.

Since encryption is deterministic,

the spatial and pixel values are recovered to obtain an image that is close to the original image, following decryption.

The histograms of the decrypted image should match those of the original, with a prove of satisfactory recovery for color distributions and contents.

5.5 Histogram Analysis

Original Image: Histograms of the red, green, and blue channels are clearly indicative of individual peaks, because they represent the distribution of intensities in the tonal as well as spatial structure of an image, meaning the visual features.

Encrypted Image: Histograms have nearly
flat distributions, indicating a uniformity in pixel intensity, meaning that encryption actua
lly suppresses any traceable patterns or correlation.

Decrypted Image: The histograms regain their
original shape as compared to the source image. This means the pixel values
and the spatial arrangements have also been recovered properly during the process
of decryption.

Chapter 6

Conclusion and Future Work

Conclusion

This paper puts forward an image encryption system based on the chaotic mapping technique, along with testing and discussion of its performance concerning the overcoming of the great challenges in modern image encryption: security, efficiency, and reliability. Highlights: Security: Chaotic maps are used in this system; hence, there is a high level of randomness introduced to the pixel values, and, therefore, it is very difficult to predict the encrypted image. It is powerful enough to stand any statistical and visual analysis, safely keeping sensitive information. Quality of the Image: Original image remains restored with no or slight quality loss in the decryption process. The system has been proven to be effective with high PSNR values and low MSE after decryption. Efficiency Fast encryption as well as decryption for large images, making it practical for use in real-time applications Reliable The system was very sensitive to the encryption key. Even a slight change in the key would result in decryption failure and unauthorized users could not access the data. The architecture combines chaotic encryption and image steganography to enhance security while performing better. It provides an improved performance for suitability in secure communications and also for digital watermarking and data protection.

Future Directions: Future Developments The system is quite effective, but it has much potential for further developments in the following areas:

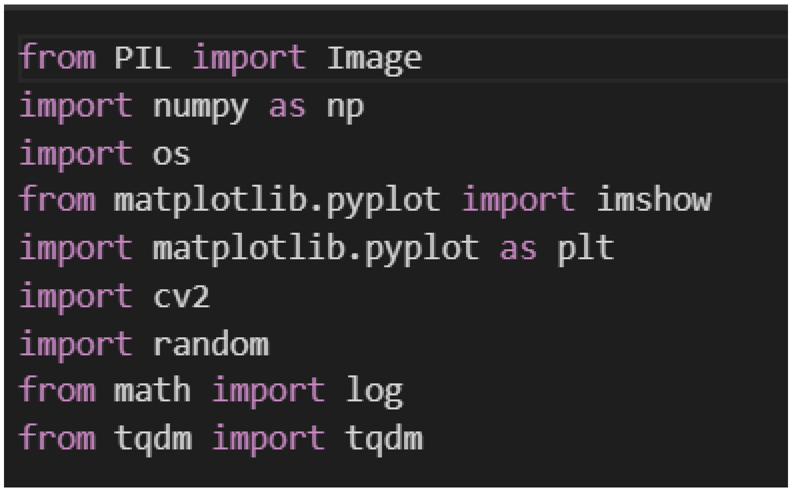
More complex chaotic systems: Higher-order Lorenz or Rössler systems may be applied in order to make the encryption more unpredictable and secure.

Hybrid Encryption Methods: A chaotic map combined with any technology, including AES, can add an extra layer of security to sensitive data. Real-Time Usage. Testing your system in real-world applications such as video streams or surveillance security, brings out a real-time performance perspective of the system. Equipment Optimization. Having your system on hardware like GPUs or FPGAs can accelerate some processing and can be more practical for huge applications.

Attack Protection: Your system will face more testing against threats like brute force and side-channel attacks. This will make your system much more resilient to hacker attacks.

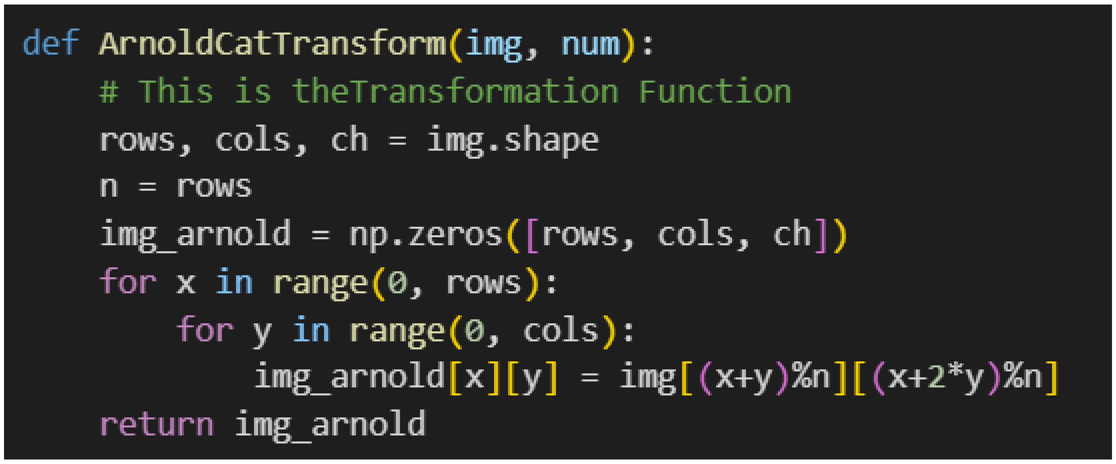
Scaling for Large Media: Ultra-high-definition images and video require optimizations on your system, which will result in it being more fitting to most modern applications. Your systems will be more secure, faster, and scalable. Chaotic cryptography is likely to play a significant role in securing digital communications and protecting future data in today’s rapidly changing digital world.

4: Code Snippets



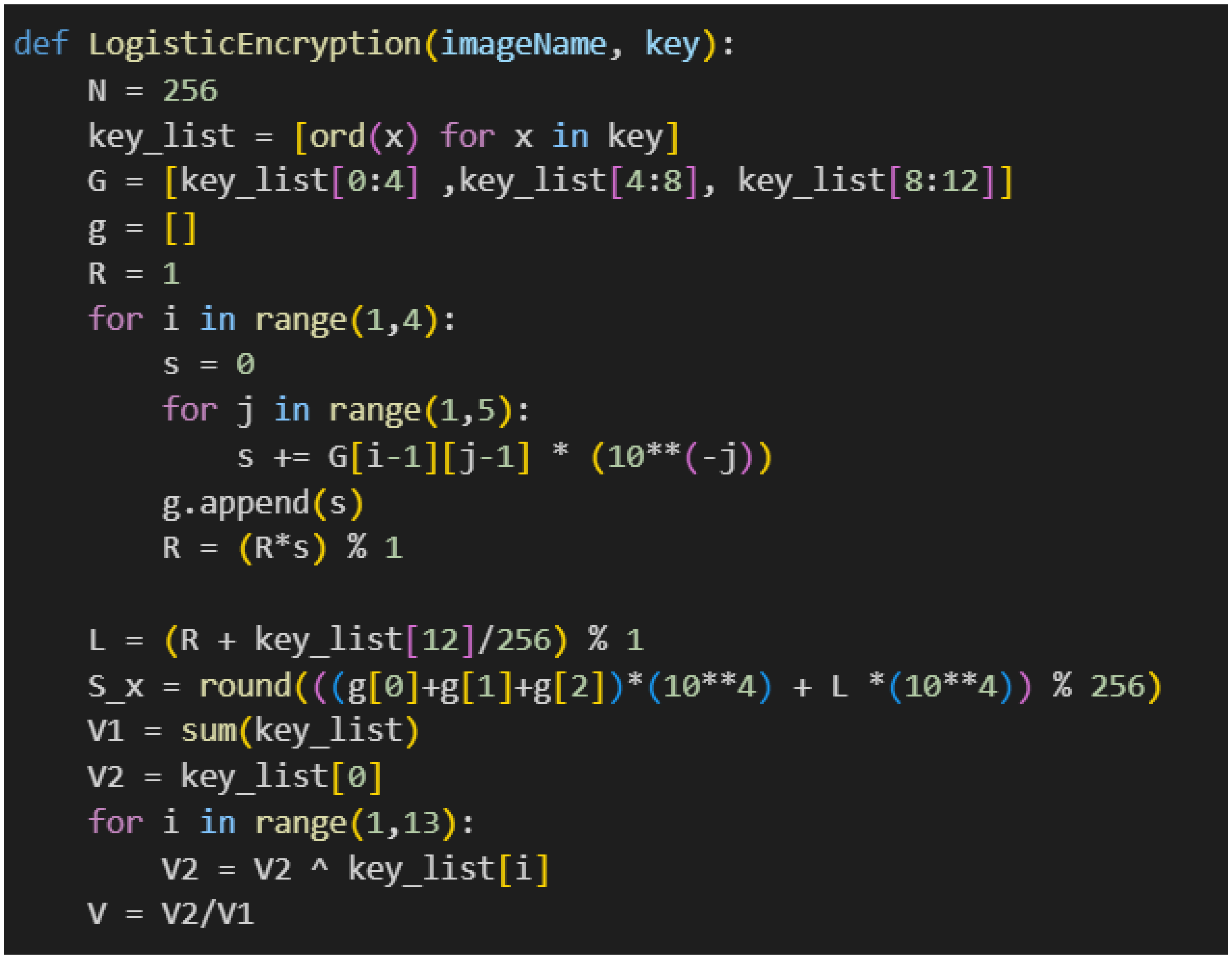
```
from PIL import Image
import numpy as np
import os
from matplotlib.pyplot import imshow
import matplotlib.pyplot as plt
import cv2
import random
from math import log
from tqdm import tqdm
```

The code imports libraries for image processing, visualization, mathematical operations, and progress tracking, commonly used in computer vision and data analysis tasks.

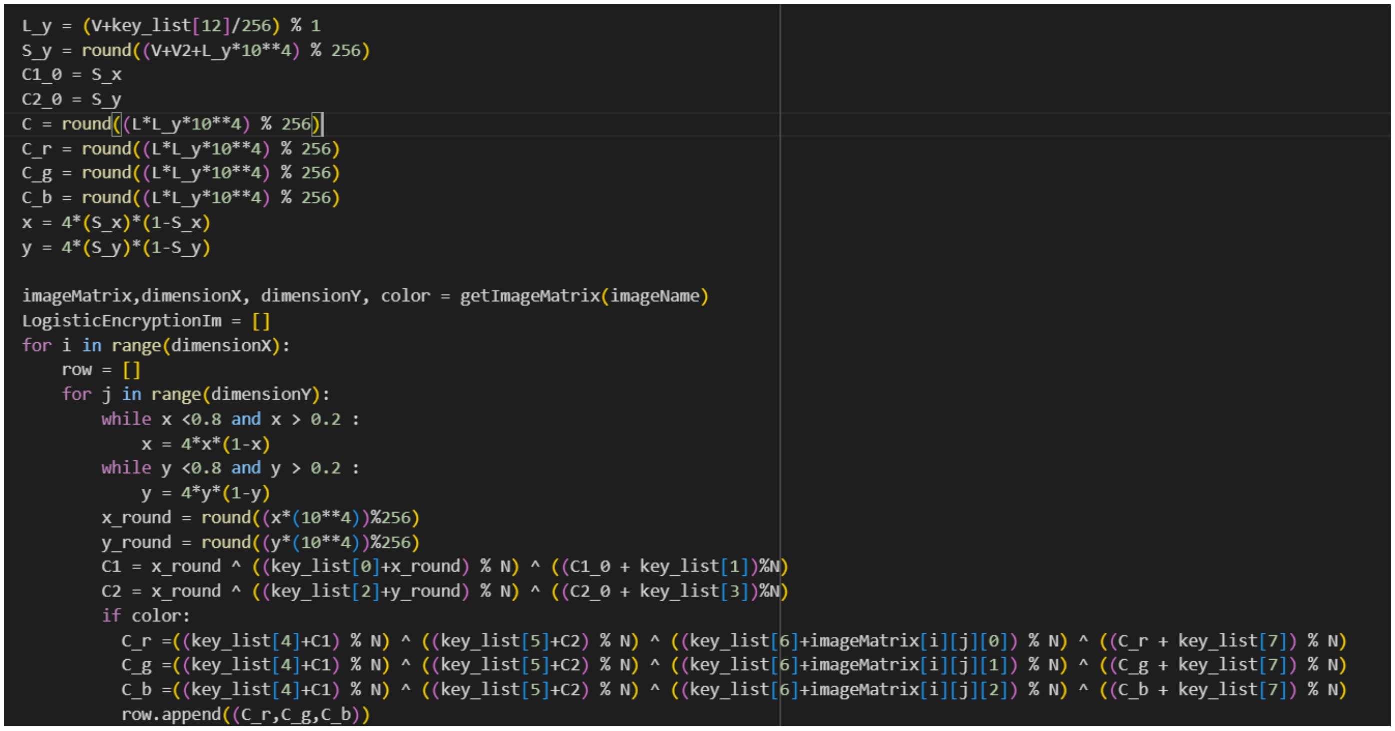


```
def ArnoldCatTransform(img, num):
    # This is theTransformation Function
    rows, cols, ch = img.shape
    n = rows
    img_arnold = np.zeros([rows, cols, ch])
    for x in range(0, rows):
        for y in range(0, cols):
            img_arnold[x][y] = img[(x+y)%n][(x+2*y)%n]
    return img_arnold
```

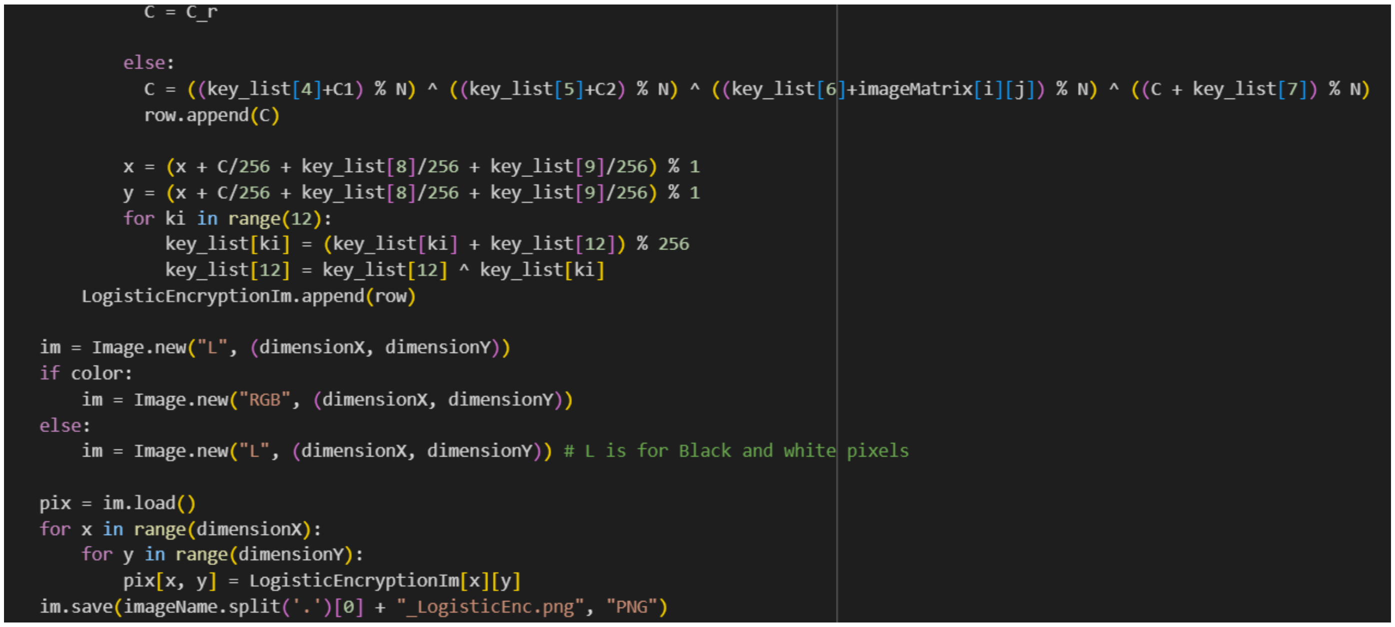
The ArnoldCatTransform function applies the Arnold Cat Map transformation to an image for scrambling, based on modulo arithmetic on pixel positions.



```
def LogisticEncryption(imageName, key):
    N = 256
    key_list = [ord(x) for x in key]
    G = [key_list[0:4] ,key_list[4:8], key_list[8:12]]
    g = []
    R = 1
    for i in range(1,4):
        s = 0
        for j in range(1,5):
            s += G[i-1][j-1] * (10**(-j))
        g.append(s)
        R = (R*s) % 1

    L = (R + key_list[12]/256) % 1
    S_x = round(((g[0]+g[1]+g[2])*(10**4) + L *(10**4)) % 256)
    V1 = sum(key_list)
    V2 = key_list[0]
    for i in range(1,13):
        V2 = V2 ^ key_list[i]
    V = V2/V1
```



```
L_y = (V+key_list[12]/256) % 1
S_y = round((V+V2+L_y*10**4) % 256)
C1_0 = S_x
C2_0 = S_y
C = round((L*L_y*10**4) % 256)
C_r = round((L*L_y*10**4) % 256)
C_g = round((L*L_y*10**4) % 256)
C_b = round((L*L_y*10**4) % 256)
x = 4*(S_x)*(1-S_x)
y = 4*(S_y)*(1-S_y)

imageMatrix,dimensionX, dimensionY, color = getImageMatrix(imageName)
LogisticEncryptionIm = []
for i in range(dimensionX):
    row = []
    for j in range(dimensionY):
        while x <0.8 and x > 0.2 :
            x = 4*x*(1-x)
        while y <0.8 and y > 0.2 :
            y = 4*y*(1-y)
        x_round = round((x*(10**4))%256)
        y_round = round((y*(10**4))%256)
        C1 = x_round ^ ((key_list[0]+x_round) % N) ^ ((C1_0 + key_list[1])%N)
        C2 = x_round ^ ((key_list[2]+y_round) % N) ^ ((C2_0 + key_list[3])%N)
        if color:
          C_r =((key_list[4]+C1) % N) ^ ((key_list[5]+C2) % N) ^ ((key_list[6]+imageMatrix[i][j][0]) % N) ^ ((C_r + key_list[7]) % N)
          C_g =((key_list[4]+C1) % N) ^ ((key_list[5]+C2) % N) ^ ((key_list[6]+imageMatrix[i][j][1]) % N) ^ ((C_g + key_list[7]) % N)
          C_b =((key_list[4]+C1) % N) ^ ((key_list[5]+C2) % N) ^ ((key_list[6]+imageMatrix[i][j][2]) % N) ^ ((C_b + key_list[7]) % N)
          row.append((C_r,C_g,C_b))
```



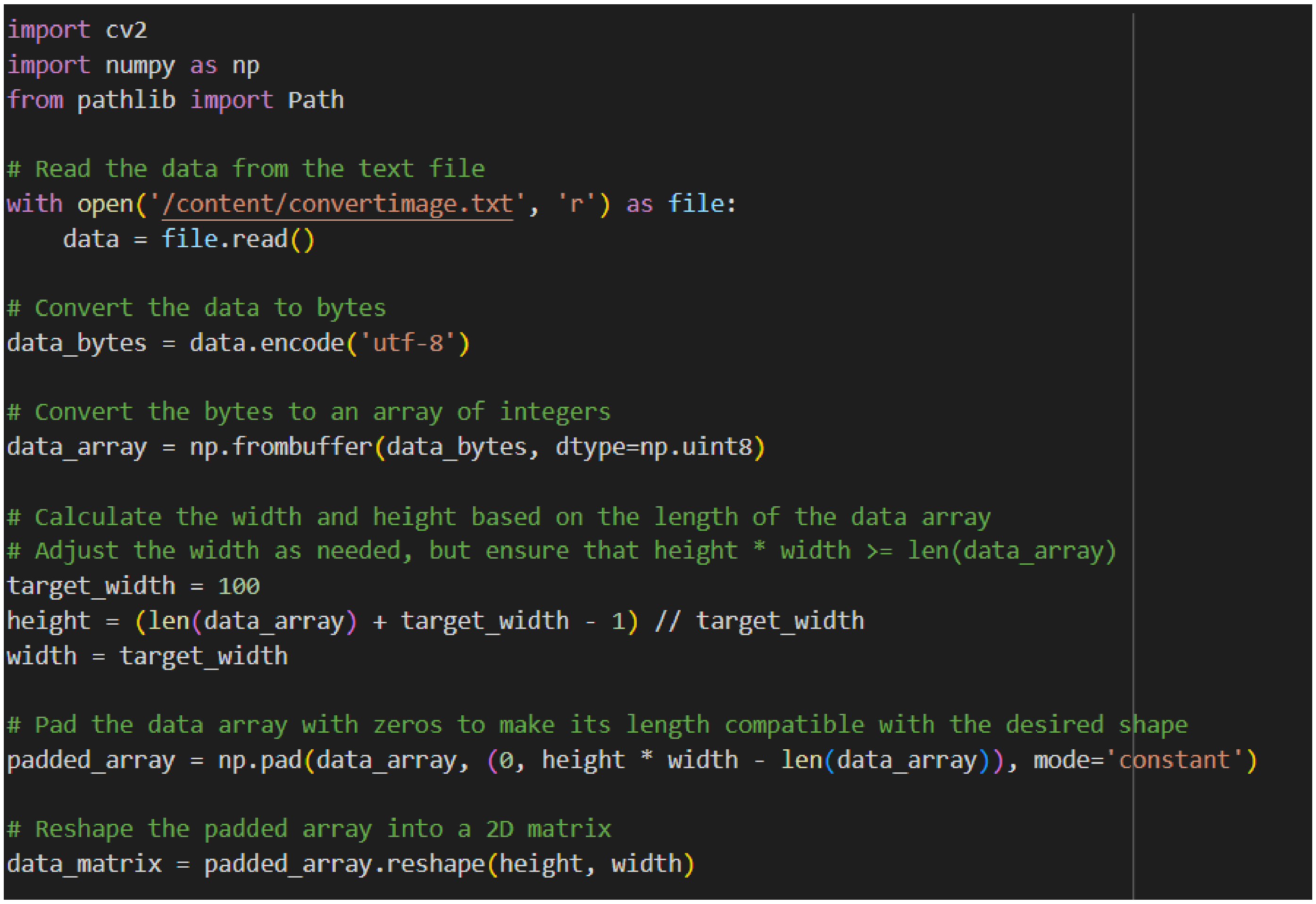
```
          C = C_r

        else:
          C = ((key_list[4]+C1) % N) ^ ((key_list[5]+C2) % N) ^ ((key_list[6]+imageMatrix[i][j]) % N) ^ ((C + key_list[7]) % N)
          row.append(C)

        x = (x + C/256 + key_list[8]/256 + key_list[9]/256) % 1
        y = (x + C/256 + key_list[8]/256 + key_list[9]/256) % 1
        for ki in range(12):
            key_list[ki] = (key_list[ki] + key_list[12]) % 256
            key_list[12] = key_list[12] ^ key_list[ki]
    LogisticEncryptionIm.append(row)

im = Image.new("L", (dimensionX, dimensionY))
if color:
    im = Image.new("RGB", (dimensionX, dimensionY))
else:
    im = Image.new("L", (dimensionX, dimensionY)) # L is for Black and white pixels

pix = im.load()
for x in range(dimensionX):
    for y in range(dimensionY):
        pix[x, y] = LogisticEncryptionIm[x][y]
im.save(imageName.split('.')[0] + "_LogisticEnc.png", "PNG")
```

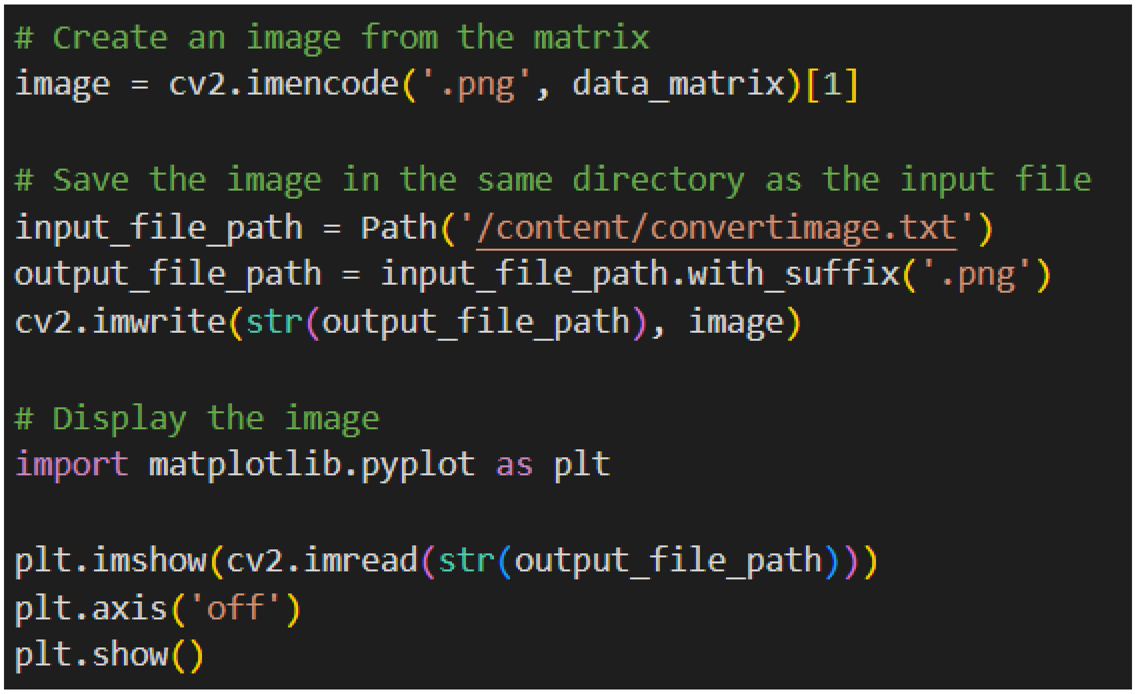
The LogisticEncryption function encrypts images using a chaotic logistic map and a secret key. It generates pseudo-random values from the key to scramble pixel values with XOR and modulo operations. The resulting encrypted image is saved as a new file, supporting both grayscale and color formats.

```python
image = 'ID'
ext = '.png'

pil_im = Image.open(image + ext, 'r')
imshow(np.asarray(pil_im), cmap='gray')
```

```python
def getImageMatrix(imageName):
    im = Image.open(imageName)
    pix = im.load()
    color = 1
    if type(pix[0,0]) == int:
      color = 0
    image_size = im.size
    image_matrix = []
    for width in range(int(image_size[0])):
        row = []
        for height in range(int(image_size[1])):
                row.append((pix[width,height]))
        image_matrix.append(row)
    return image_matrix, image_size[0], image_size[1],color
```

The getImageMatrix function extracts the pixel values of an image into a matrix, determines its dimensions, and checks if it's grayscale or color.


```python
import cv2
import numpy as np
from pathlib import Path

# Read the data from the text file
with open('/content/convertimage.txt', 'r') as file:
    data = file.read()

# Convert the data to bytes
data_bytes = data.encode('utf-8')

# Convert the bytes to an array of integers
data_array = np.frombuffer(data_bytes, dtype=np.uint8)

# Calculate the width and height based on the length of the data array
# Adjust the width as needed, but ensure that height * width >= len(data_array)
target_width = 100
height = (len(data_array) + target_width - 1) // target_width
width = target_width

# Pad the data array with zeros to make its length compatible with the desired shape
padded_array = np.pad(data_array, (0, height * width - len(data_array)), mode='constant')

# Reshape the padded array into a 2D matrix
data_matrix = padded_array.reshape(height, width)
```

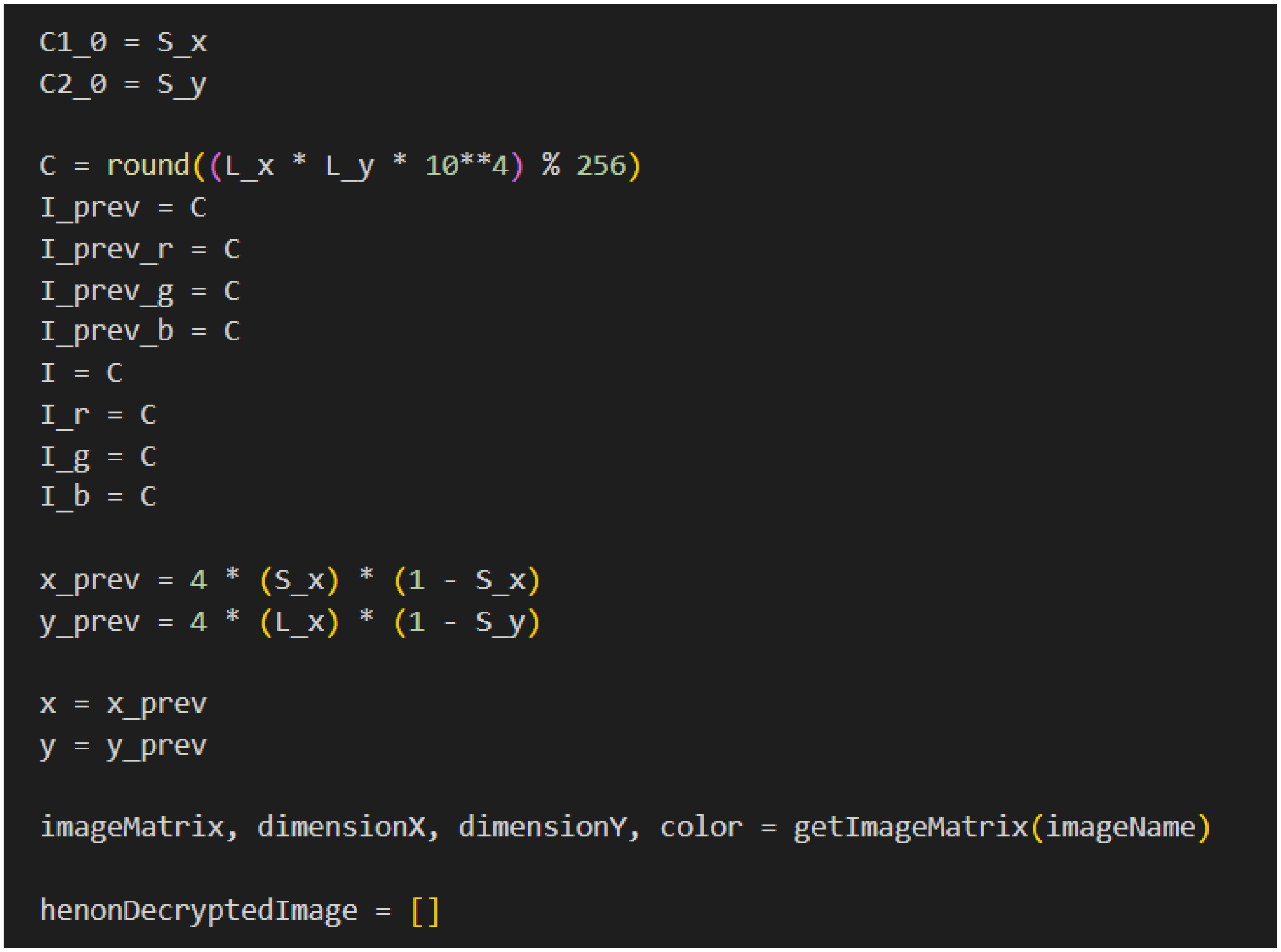

```python
# Create an image from the matrix
image = cv2.imencode('.png', data_matrix)[1]

# Save the image in the same directory as the input file
input_file_path = Path('/content/convertimage.txt')
output_file_path = input_file_path.with_suffix('.png')
cv2.imwrite(str(output_file_path), image)

# Display the image
import matplotlib.pyplot as plt

plt.imshow(cv2.imread(str(output_file_path)))
plt.axis('off')
plt.show()
```

The script converts text into a visual representation by encoding the content into a pixel matrix, reshaping it to a target size, and saving it as an image. It also displays the resulting image using Matplotlib.

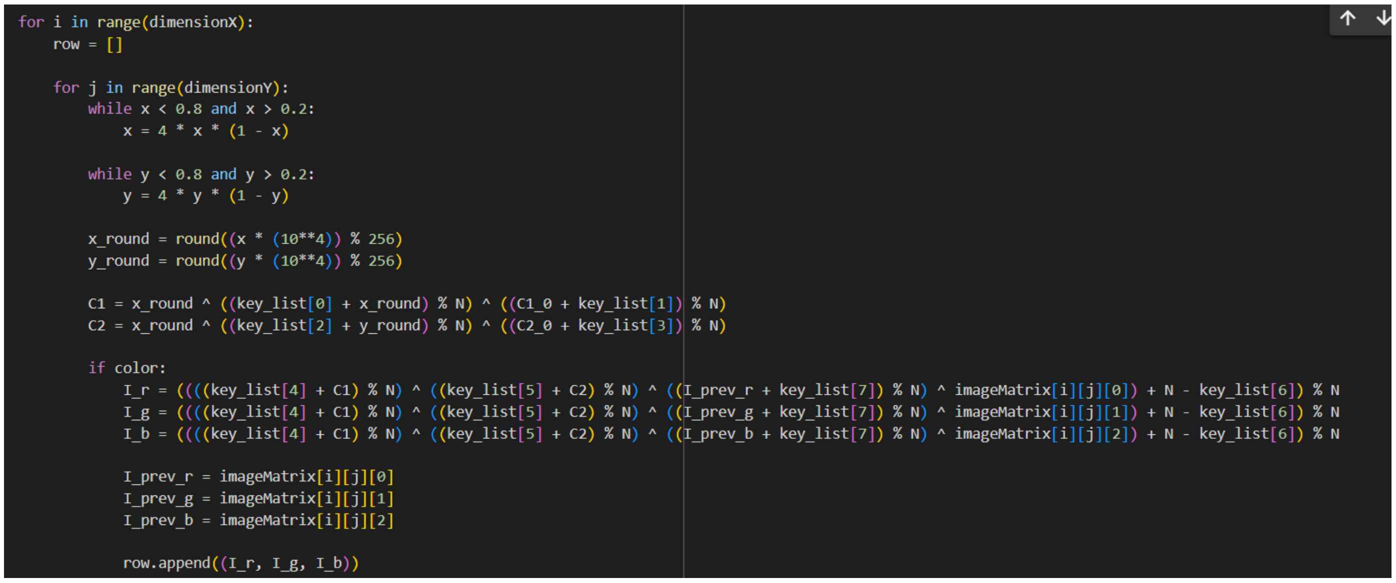
```
LogisticEncryption("ID.png", "abcdefghijklm")
im = Image.open("ID_LogisticEnc.png", 'r')
imshow(np.asarray(im), cmap='gray')
```

This code encrypts the image ID.png using the LogisticEncryption function with the key "abcdefghijklm", then opens and displays the encrypted image as a grayscale image.

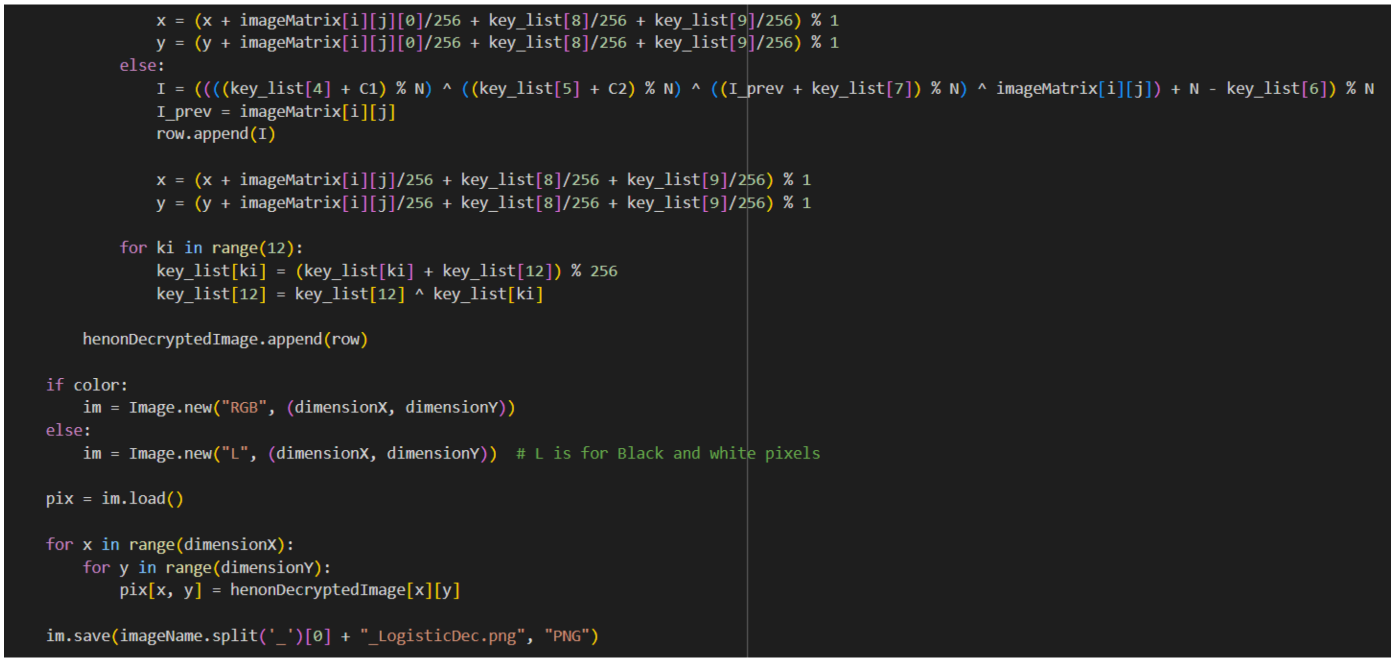
```
def LogisticDecryption(imageName, key):
    N = 256
    key_list = [ord(x) for x in key]

    G = [key_list[0:4], key_list[4:8], key_list[8:12]]
    g = []
    R = 1

    for i in range(1, 4):
        s = 0
        for j in range(1, 5):
            s += G[i-1][j-1] * (10**(-j))
        g.append(s)
        R = (R * s) % 1

    L_x = (R + key_list[12]/256) % 1
    S_x = round(((g[0] + g[1] + g[2]) * (10**4) + L_x * (10**4)) % 256)

    V1 = sum(key_list)
    V2 = key_list[0]

    for i in range(1, 13):
        V2 = V2 ^ key_list[i]
    V = V2 / V1

    L_y = (V + key_list[12]/256) % 1
    S_y = round((V + V2 + L_y * 10**4) % 256)
```



```
C1_0 = S_x
C2_0 = S_y

C = round((L_x * L_y * 10**4) % 256)
I_prev = C
I_prev_r = C
I_prev_g = C
I_prev_b = C
I = C
I_r = C
I_g = C
I_b = C

x_prev = 4 * (S_x) * (1 - S_x)
y_prev = 4 * (L_x) * (1 - S_y)

x = x_prev
y = y_prev

imageMatrix, dimensionX, dimensionY, color = getImageMatrix(imageName)

henonDecryptedImage = []
```

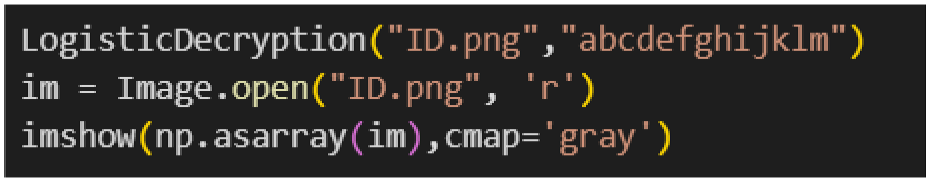


```
for i in range(dimensionX):
    row = []

    for j in range(dimensionY):
        while x < 0.8 and x > 0.2:
            x = 4 * x * (1 - x)

        while y < 0.8 and y > 0.2:
            y = 4 * y * (1 - y)

        x_round = round((x * (10**4)) % 256)
        y_round = round((y * (10**4)) % 256)

        C1 = x_round ^ ((key_list[0] + x_round) % N) ^ ((C1_0 + key_list[1]) % N)
        C2 = x_round ^ ((key_list[2] + y_round) % N) ^ ((C2_0 + key_list[3]) % N)

        if color:
            I_r = ((((key_list[4] + C1) % N) ^ ((key_list[5] + C2) % N) ^ ((I_prev_r + key_list[7]) % N) ^ imageMatrix[i][j][0]) + N - key_list[6]) % N
            I_g = ((((key_list[4] + C1) % N) ^ ((key_list[5] + C2) % N) ^ ((I_prev_g + key_list[7]) % N) ^ imageMatrix[i][j][1]) + N - key_list[6]) % N
            I_b = ((((key_list[4] + C1) % N) ^ ((key_list[5] + C2) % N) ^ ((I_prev_b + key_list[7]) % N) ^ imageMatrix[i][j][2]) + N - key_list[6]) % N

            I_prev_r = imageMatrix[i][j][0]
            I_prev_g = imageMatrix[i][j][1]
            I_prev_b = imageMatrix[i][j][2]

            row.append((I_r, I_g, I_b))
```



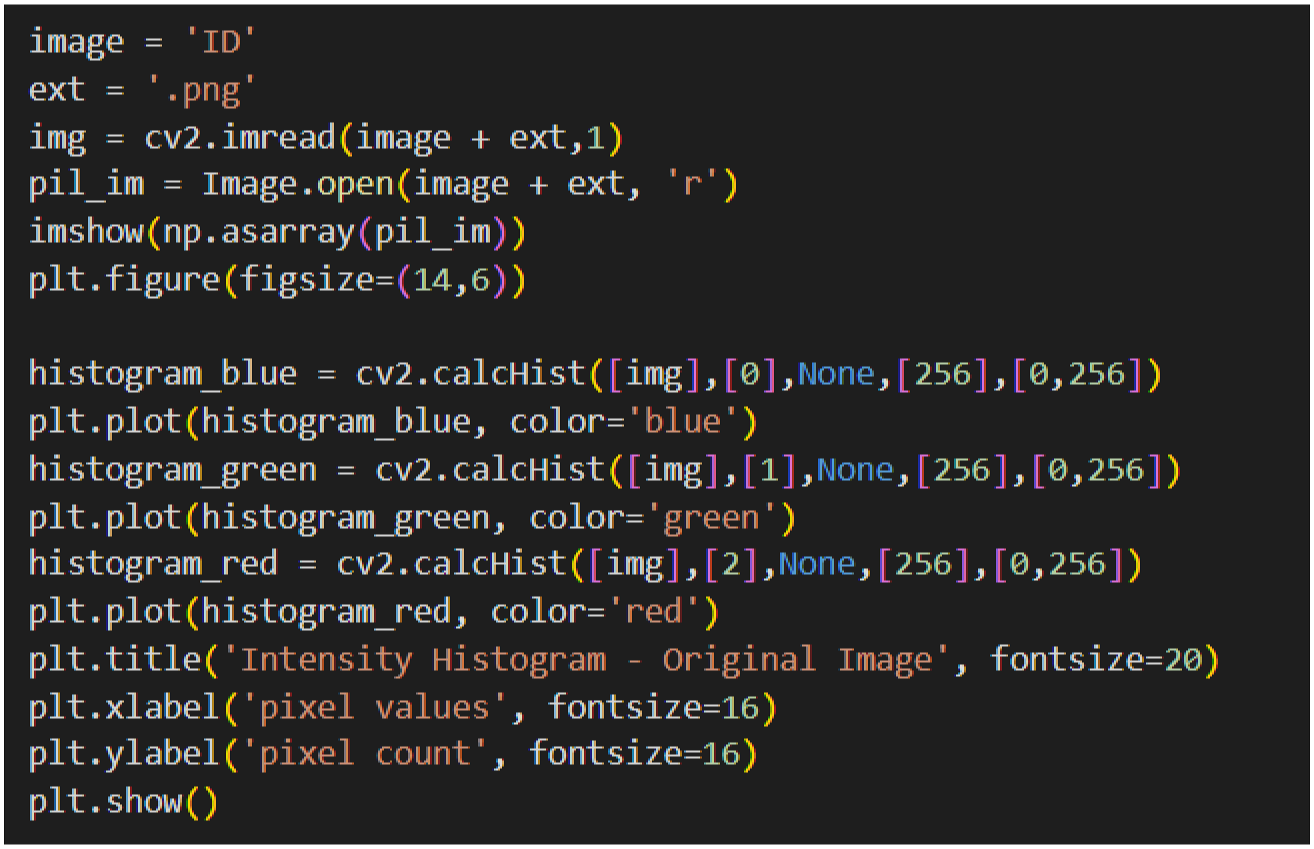
```
            x = (x + imageMatrix[i][j][0]/256 + key_list[8]/256 + key_list[9]/256) % 1
            y = (y + imageMatrix[i][j][0]/256 + key_list[8]/256 + key_list[9]/256) % 1
        else:
            I = ((((key_list[4] + C1) % N) ^ ((key_list[5] + C2) % N) ^ ((I_prev + key_list[7]) % N) ^ imageMatrix[i][j]) + N - key_list[6]) % N
            I_prev = imageMatrix[i][j]
            row.append(I)

            x = (x + imageMatrix[i][j]/256 + key_list[8]/256 + key_list[9]/256) % 1
            y = (y + imageMatrix[i][j]/256 + key_list[8]/256 + key_list[9]/256) % 1

        for ki in range(12):
            key_list[ki] = (key_list[ki] + key_list[12]) % 256
            key_list[12] = key_list[12] ^ key_list[ki]

    henonDecryptedImage.append(row)

if color:
    im = Image.new("RGB", (dimensionX, dimensionY))
else:
    im = Image.new("L", (dimensionX, dimensionY))  # L is for Black and white pixels

pix = im.load()

for x in range(dimensionX):
    for y in range(dimensionY):
        pix[x, y] = henonDecryptedImage[x][y]

im.save(imageName.split('_')[0] + "_LogisticDec.png", "PNG")
```

The function LogisticDecryption performs an inverse process to decryption which was applied by the LogisticEncryption function, with exactly the same secret key. Parameters are set for outputting chaotic values from logistic map and constant values that depend on a secret key for decryption. It then iteratively processes each pixel or channel making use of an inverse XOR operation with modular arithmetic to restore the original values; grayscale images are treated differently than color images. The key values and noisy variables are constantly updated to represent the progress of the encryption process. Finally, the function gathers the decrypted pixel data in an image and saves it with the original name appended by "_LogisticDec.png", thus restoring the original image.



```
LogisticDecryption("ID.png","abcdefghijklm")
im = Image.open("ID.png", 'r')
imshow(np.asarray(im),cmap='gray')
```

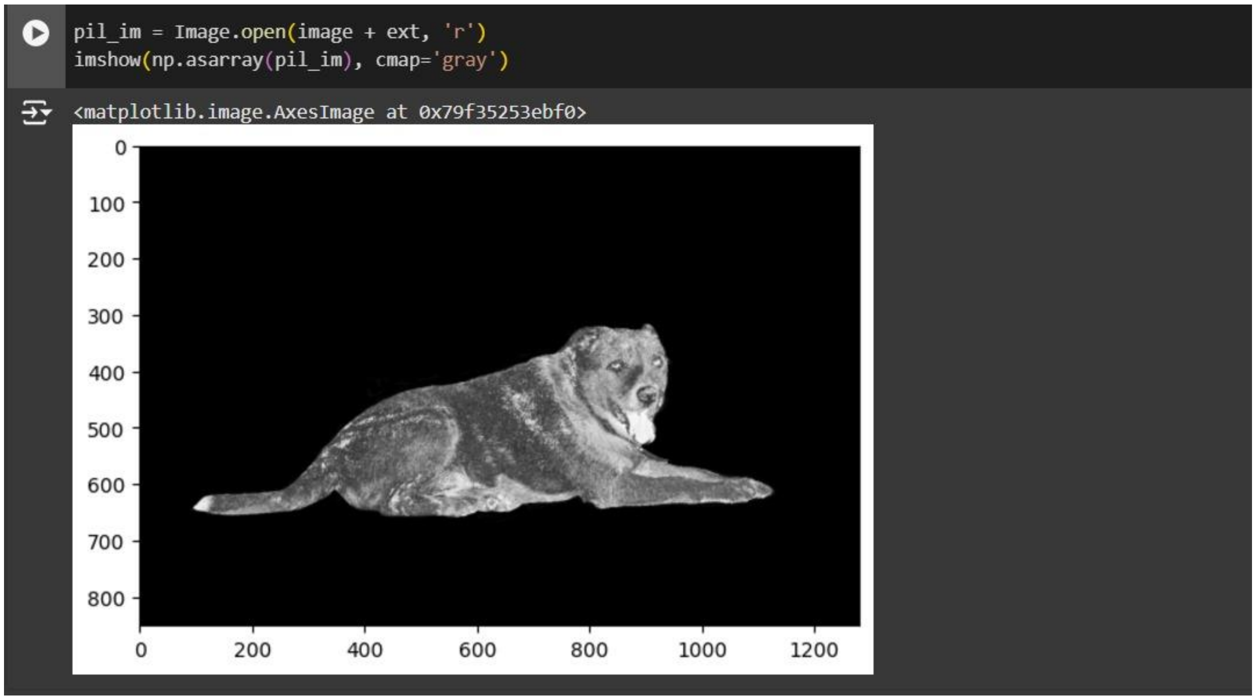
The code decrypts the image ID.png using the LogisticDecryption function with the key "abcdefghijklm", then displays the decrypted image in grayscale.



```python
image = 'ID'
ext = '.png'
img = cv2.imread(image + ext,1)
pil_im = Image.open(image + ext, 'r')
imshow(np.asarray(pil_im))
plt.figure(figsize=(14,6))

histogram_blue = cv2.calcHist([img],[0],None,[256],[0,256])
plt.plot(histogram_blue, color='blue')
histogram_green = cv2.calcHist([img],[1],None,[256],[0,256])
plt.plot(histogram_green, color='green')
histogram_red = cv2.calcHist([img],[2],None,[256],[0,256])
plt.plot(histogram_red, color='red')
plt.title('Intensity Histogram - Original Image', fontsize=20)
plt.xlabel('pixel values', fontsize=16)
plt.ylabel('pixel count', fontsize=16)
plt.show()
```

The code reads the image ID.png, displays it, and plots the intensity histograms for the blue, green, and red channels.

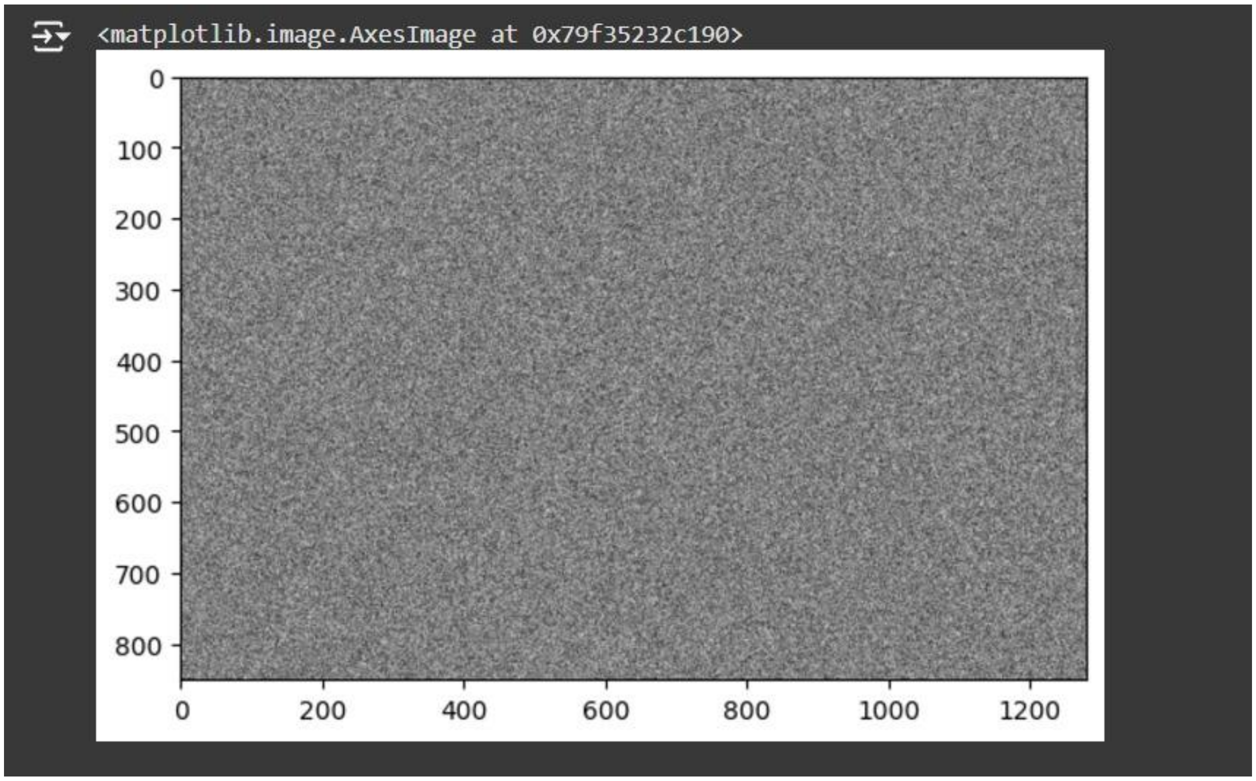
5. OUTPUT



```
pil_im = Image.open(image + ext, 'r')
imshow(np.asarray(pil_im), cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x79f35253ebf0>
```

Figure 1: compressed image before encryption



```
<matplotlib.image.AxesImage at 0x79f35232c190>
```

Figure 2: encrypted image

Logistic Encryption:

The image of the dog is first read and encrypted using a chaotic logistic map that is controlled by a

secret key. This encryption process involves shuffling pixel values through XOR operations and random variables, thus creating an encrypted image that mimics random noise. The image data is treated as a matrix of pixel values, which are encoded in bytes and converted to a specific format for storage. In this method, the image data can be represented in different visual formats.

Arnold's Cat Map:

For further hiding the image, the Arnold's Cat Map transformation is applied. In the process, the pixel positions are rearranged, where more dramatic visual changes to the image take place.

Image Decryption:

Apply the Logistics Decryption with the same key, and the encrypted dog image will be returned to its original form by reversing all chaotic transformations, restoring the previous values of pixels and spatial placement.

Visualization

Generate a pixel intensity histogram for the dog image which illustrates the distribution of pixel values among the red, green, and blue channels.

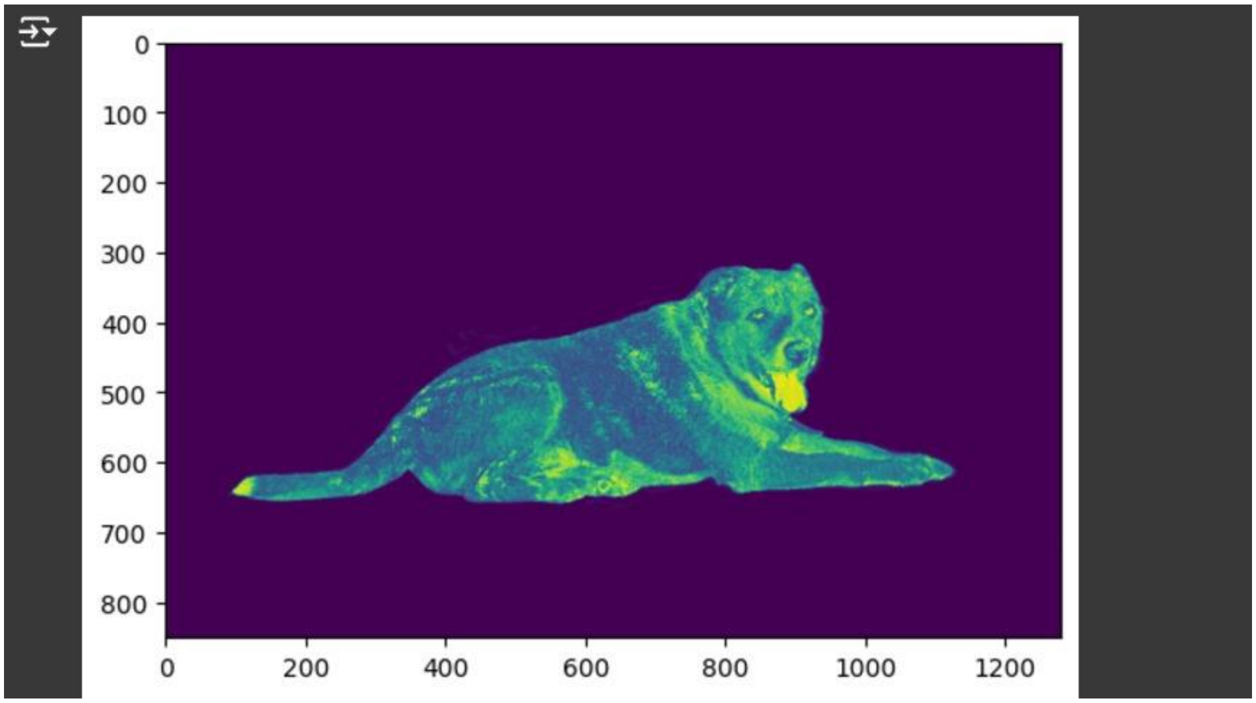


Figure 3: decrypted image

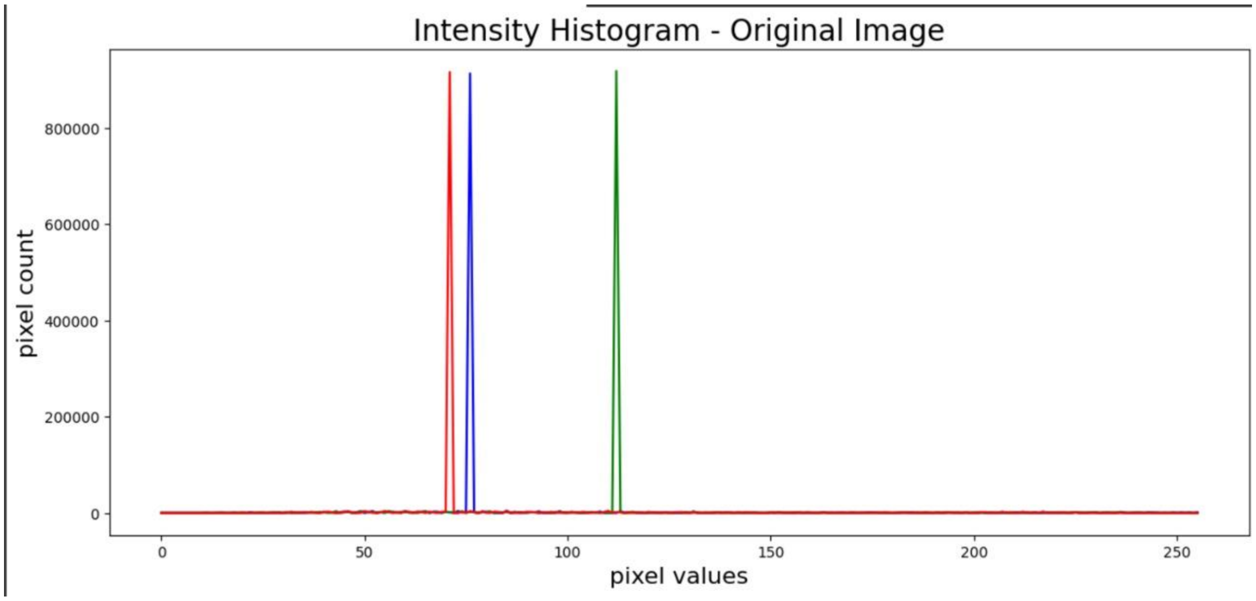


Figure 4: intensity histogram

6. REFERENCES

Patel, H., & Dave, P. (2012). Steganography technique based on DCT coefficients. International Journal of Engineering Research and Applications, 2(1), 713–717.

Mostafa, R., Ali, A. F., & EI Taweal, G. (2015). Hybrid curvelet transform and least significant bit for image steganography. In 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS) (pp. 300–305). IEEE. https://doi.org/10.1109/IntelCIS.2015.7397238

Sifuzzaman, M., Islam, M. R., & Ali, M. Z. (2009). Application of wavelet transform and its advantages compared to Fourier transform. Journal of Physical Science, 13, 121–134.

Cheung, G., Magli, E., Tanaka, Y., & Ng, M. K. (2018). Graph spectral image processing. Proceedings of the IEEE, 106(5), 907–930. https://doi.org/10.1109/JPROC.2018.2799702

Zhang, X. (2016). Design of orthogonal graph wavelet filter banks. In IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society (pp. 889–894). IEEE. https://doi.org/10.1109/IECON.2016.7793133

Narang, S. K., & Ortega, A. (2012). Perfect reconstruction two-channel wavelet filter banks for graph-structured data. IEEE Transactions on Signal Processing, 60(6), 2786–2799. https://doi.org/10.1109/TSP.2012.2188718

Hammond, D. K., Vandergheynst, P., & Gribonval, R. (2011). Wavelets on graphs via spectral graph theory. Applied and Computational Harmonic Analysis, 30(2), 129–150. https://doi.org/10.1016/j.acha.2010.04.005

Hamidi, H., Amirani, M. C., & Arashloo, S. R. (2015). Local selected features of dual tree complex wavelet transform. [Publication details missing].

Majeed, M. A., Sulaiman, R., Shukur, Z., & Hasan, M. K. (2021). A review on text steganography techniques. Mathematics, 9, Article 2829. https://doi.org/10.3390/math9212829

Bilgaiyan, S., Ahmad, R., & Sagnika, S. (2023). Adaptive image steganography using rotating color channels and inverted LSB substitution. SN Computer Science, 4, Article 565. https://doi.org/10.2174/2352096516666230428104626

ALRikabi, H. T. S., & Hazim, H. T. (2021). Enhanced data security of communication system using combined encryption and steganography. iJIM, 15(5), 145. https://doi.org/10.3991/ijim.v15i05.19191

Attanayake, A. M. S. P., & Yapage, N. (2023). Application of Arnold's Cat Map and Bülban Map for image encryption in chaotic cryptography. https://doi.org/10.21203/rs.3.rs-3363736/v1

Degang Yang, Xiaofeng Liao, Yong Wang, Huaqian Yang, and Pengcheng Wei. A novel chaotic block cryptosystem based on iterating map with output-feedback. Chaos, Solitons & Fractals, 41(1):505–510, 2009.

Huaqian Yang, Kwok-Wo Wong, Xiaofeng Liao, Wei Zhang, and Pengcheng Wei. A fast image encryption and authentication scheme based on chaotic maps. Communications in Nonlinear Science and Numerical Simulation, 15:3507–3517, nov 2010.

Je Sen Teh, Moatsum Alawida, and You Cheng Sii. Implementation and practical problems of chaos-based cryptography revisited. Journal of Information Security and Applications, 50(November), 2020.

Goce Jakimoski and Ljupco Kocarev. Kocarev, L.: Chaos and Cryptography: Block Encryption Ciphers Based on Chaotic Maps. IEEE Transactions on Circuits and Systems I 48, 163-169. Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, 48:163–169, mar 2001.

Temadher Alassiry Al-Maadeed, Iqtadar Hussain, Amir Anees, and Muhammad Tahir Mustafa. A image encryption algorithm based on chaotic Lorenz system and novel primitive polynomial S-boxes. Multimedia Tools and Applications, 80(16):24801–24822, 2021.