

STEALTH AND SECURITY: EXPLORING MULTIMEDIA STEGANOGRAPHY

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

RISHAV AGRAWAL (Reg. No - 40111062)
BHARTI SINGH (Reg. No - 40120016)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Category-1 University by UGC

**Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119**

APRIL – 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Rishav Agrawal(40111062)** and **Bharti Singh(40120016)** who have done the Project work as a team entitled "**STEALTH AND SECURITY: EXPLORING MULTIMEDIA STEGANOGRAPHY**" under my supervision from November 2023 to April 2024.

[Signature]
12/4/24

Internal Guide

Dr. A. SIVASANGARI, M.E., PhD

[Signature]

Head of the Department

Dr. L. LAKSHMANAN, M.E., PhD

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)
Jeppiaar Nagar, Rajiv Gandhi Salai,
Chennai - 600 119.

Submitted for Viva voce Examination held on

15/04/2024

[Signature]
13 May 15/4/24
Internal Examiner

[Signature]
15/4/24
External Examiner

DECLARATION

I, Rishav Agrawal(Reg.No - 40111062), hereby declare that the Project Report entitled "STEALTH AND SECURITY: EXPLORING MULTIMEDIA STEGANOGRAPHY" done by me under the guidance of Dr. A. Sivasangari, M.E., PhD, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 15/04/2024

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., PhD, Dean, School of Computing, Dr. L. Lakshmanan M.E., PhD**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A. Sivasangari, M.E., PhD** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

According to the latest estimates, 328.77 million terabytes of data are created each day. Protecting and securing the transmission of this data is the key to the creation of a better digital world. Data is the pollution problem of the information age, and protecting privacy is the environmental challenge. In the present decade, images have become a valuable source of information. They play an essential role in every field whether it be medical, agriculture, defense or education. Over the past few years, steganography has received a lot of attention for passing military sensitive details. The information is passed generally via image steganography. With the increase in amount to data to hide, other multimedia formats are gaining popularity too. Secured communication on sensitive information between two parties need to consider information security as necessity. Steganography and cryptography can be used to enhance the information security. Cryptography discloses the availability of information, whereas steganography hides the availability of information so that no other than the two communicating parties can recognize it. With advancement in internet applications the information mainly consists of text, audio, image, and video type of information. This project provides a detailed analysis and methodology for Multimedia based steganography of information. Also, we tend to discuss the available methods and techniques for multimedia steganography and their performances.

Secured communication on sensitive information between two parties need to consider information security as necessity. Steganography and cryptography can be used to enhance the information security. Cryptography discloses the availability of information, whereas steganography hides the availability of information so that no other than the two communicating parties can recognize it. With advancement in internet applications the information mainly consists of text, audio, image, and video type of information. This project provides a detailed analysis and methodology for Multimedia based steganography of information. Also, we tend to discuss the available methods and techniques for multimedia steganography and their performances.

CONTENT

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	vi
1	INTRODUCTION	7-8
2	LITERATURE SURVEY	9-12
	2.1 Inferences from Literature Survey	11-12
	2.2 Open problems in Existing System	12
3	REQUIREMENT ANALYSIS	13-14
	3.1 Feasibility Studies/Risk Analysis of the Project	13
	3.2 Software Requirements Specification Document	14
4	DESCRIPTION OF PROPOSED SYSTEM	15-26
	4.1 Selected Methodology or process model	17-20
	4.2 Architecture / Overall Design of Proposed System	21
	4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System	22
	4.4 Results & Discussion	23-26
5	IMPLEMENTATION DETAILS	27-30
	5.1 Installing Spring Boot Application	28-29
	5.2 Deploying Spring Boot Application	29-30
6	RESULTS AND DISCUSSIONS	31
7	CONCLUSION	32
	7.1 Future Work	32
	REFERENCES	33
	APPENDIX	
	A SOURCE CODE	35-55
	B SCREENSHOTS	55-59
	C RESEARCH PAPER	60-68
	D CERTIFICATE	69

LIST OF FIGURES

FIGURE NO	FIGURE NAME	Page No.
1.1	Evolution of Steganography	8
4.1	Image Steganography	16
4.2	Audio Steganography	16
4.3	Video Steganography	17
4.1.1	Chaotic mapping evaluation criteria equations	19
4.1.3	Flowchart of Encryption Process	19
4.1.4	Flowchart of Decryption Process	20
4.2	System Architecture	21
4.3.1	Comparative Study of Various types of Chaotic Mapping Methods	23
4.4.1	Welcome Page of Application when started	24
4.4.2	Human	25
4.4.3	Animal	25
4.4.4	Embedded	25
4.4.5	Encryption Webpage	25
4.4.6	Decryption Webpage	26
4.4.7	Spring Application Started	26
5.1	Deployment Code	28
5.1.1	Application Configuration	29
5.2.1	Directory Structure of Project	30
5.2.2	Image Steganography Implementation	31
5.2.3	Audio Steganography Implementation	31
5.2.4	Video Steganography Implementation	31
Table 1	Study of Various types of Chaotic Mapping Methods	23

CHAPTER 1

INTRODUCTION

Steganography is the practice of concealing one piece of information within another in such a way that it is difficult to detect or decipher. It is often used to hide the existence of a secret message within an innocuous carrier medium, such as text, images, audio, or video files. The word "steganography" is derived from the Greek words "steganos" meaning "covered" and "graphia" meaning "writing" essentially describing the act of hiding information within another form of data.

Multimedia steganography, specifically, focuses on concealing information within multimedia files like images, audio, and video. This branch of steganography has gained significant attention due to the proliferation of digital multimedia content on the internet. Here's a brief introduction to steganography and multimedia steganography.

Image Steganography: In image steganography, secret data is embedded within an image file. Common techniques include modifying the least significant bits of the pixel values or altering the frequency components of the image in a way that is imperceptible to the human eye. **Audio Steganography:** In audio steganography, secret data is hidden within an audio file, often inaudibly. Techniques may involve modifying the amplitude or frequency components of audio signals to encode the information. **Video Steganography** hides data within video files. It can be more challenging than other forms, as it involves both spatial and temporal data. Techniques often rely on modifying frames or altering specific video characteristics. **Network steganography** is the youngest branch of information hiding. It is a fast-developing field recent years have resulted in multiple new information hiding methods, which can be exploited in various types of networks. The exploitation of protocols belonging to the Open Systems Interconnection reference model is the essence of network steganography. This family of methods may utilize one or more protocols simultaneously or the relationships between them – relying on the modification of their intrinsic properties for the embedding of steganograms.

Lately, another steganographic method TranSteg intended for a broad class of multimedia and real-time applications, like IP telephony, has been proposed. TranSteg bases on the general idea of transcoding of the voice data from a higher bit rate codec, and thus greater voice payload size, to a lower bit rate codec with smaller voice payload size. This occurs with the least possible degradation in voice quality. In other words, compression of the overt data is utilized to make space for the steganogram in the payload field. The achieved steganographic bandwidth is as high as 32 kbit/s.

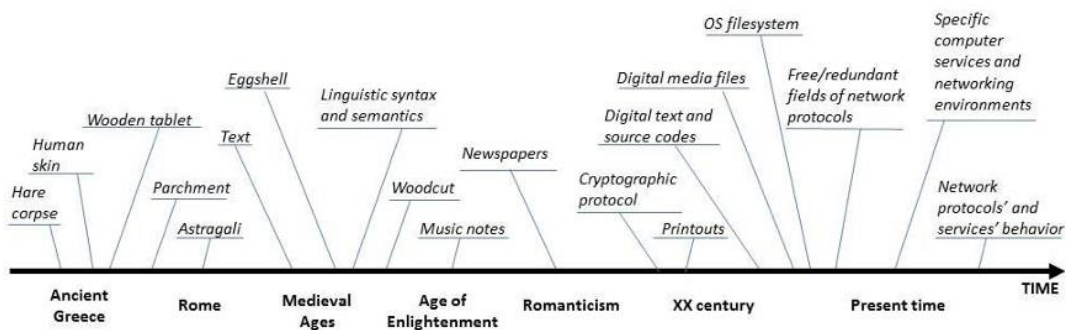


Fig 1.1 Evolution of Steganography

In conclusion, steganography, especially multimedia steganography, plays a crucial role in information security, privacy, and copyright protection. It allows for the covert transmission of data while maintaining the appearance of the carrier medium, making it an essential tool in various fields, including cybersecurity and digital

CHAPTER 2

LITERATURE SURVEY

This problem statement has been extensively studied over the past 5 years by researchers and automotive companies in a bid to create a solution, and all their solutions vary from analyzing various patterns of distractive habits to analyzing health vitals of the driver.

The work of Dr. Alluri Harika et al [1] introduced a steganography image imaging application which is based on the Advanced Encryption Standard and random bit technique. The AES technique is used to encrypt the secret message and the random bit technique is used to hide the encrypted message in the image.

Multimedia steganography involves various techniques, including Least Significant Bit embedding, frequency domain methods e.g., Discrete Cosine Transform, and spread spectrum techniques. Several papers delve into the analysis and enhancement of these techniques, such as "Steganography Using Reversible Texture Synthesis" by Thien et al [2].

Research in this area focuses on improving the security of steganography and developing more robust detection methods. "Current Challenges in Steganography for Digital Images" by Naveed et al. [3] is an example of a paper discussing the challenges and security issues in image steganography.

Studies aim to increase the information hiding capacity while maintaining perceptual quality. Papers like "Adaptive Steganography with Minimization of the Impact on the Carrier Image" by Dumitrescu et al. [4] explore ways to achieve this balance.

Steganography isn't limited to images; it extends to audio and video. Research papers such as "Audio Steganography: A Comprehensive Review" by Alzahrani and Almutairi [5] cover methods and challenges in audio steganography.

Deep learning techniques have also been applied to improve steganography and steganalysis. "Towards Unsupervised Deep Steganography" by Xu et al. [6] discusses using convolutional neural networks for steganography

Steganalysis is the art of detecting hidden data. Papers like "Steganalysis of Multimedia: A Comprehensive Review" by Hussain et al. [7] provide insights into steganalysis methods and challenges.

Some papers explore practical applications of multimedia steganography, such as "A Survey on Steganography Techniques for Digital Images" by Singh and Khanna [8], which discusses the use of steganography in medical imaging.

Hardware acceleration, particularly using GPUs, has been a focus in recent research. "Parallel Steganography and Steganalysis on Graphics Hardware" by Bianchi and Piva [9] is an example of work in this area.

Steganalysis is the art of detecting hidden data. Papers like "Steganalysis of Multimedia: A Comprehensive Review" by Hussain et al. [8] provide insights into steganalysis methods and challenges.

Some papers explore practical applications of multimedia steganography, such as "A Survey on Steganography Techniques for Digital Images" by Singh and Khanna [9], which discusses the use of steganography in medical imaging.

Research in this area focuses on improving the security of steganography and developing more robust detection methods. "Current Challenges in Steganography for Digital Images" by Naveed et al. [10] is an example of a paper discussing the challenges and security issues in image steganography.

Studies aim to increase the information hiding capacity while maintaining perceptual quality. Papers like "Adaptive Steganography with Minimization of the Impact on the Carrier Image" by Dumitrescu et al. [11] explore ways to achieve this balance.

2.1 INFERENCES FROM LITREATURE SURVEY

Inferences drawn from a literature survey of multimedia steganography can provide insights into the current state of research and highlight trends and challenges within the field. Here are some common inferences that can be made based on a literature survey of multimedia steganography:

Diverse and Evolving Techniques: Multimedia steganography encompasses a wide range of techniques, from traditional LSB-based methods to advanced frequency domain and deep learning approaches. Researchers continue to develop and refine these techniques to achieve better hiding capacity and imperceptibility. **Security and Detection Remain Critical** Ensuring the security of hidden information and developing robust steganalysis methods are ongoing challenges. Researchers are working on both sides of the spectrum, devising more secure embedding techniques and more effective detection methods. **Balancing Capacity and Quality** Striking a balance between increasing data hiding capacity and maintaining the perceptual quality of multimedia files is a significant concern. Research is focused on methods that hide larger amounts of data while minimizing the visual or auditory impact on the carrier file. **Extension to Audio and Video** Steganography is not limited to images; it extends to audio and video data. Research has increasingly explored techniques and challenges specific to audio and video steganography. **Emergence of Deep Learning:** Deep learning techniques have made their way into multimedia steganography, with convolutional neural networks and other models being applied to enhance both steganography and steganalysis.

Steganalysis Advances: Steganalysis, the process of detecting hidden data, is a growing area of research. Researchers are developing more sophisticated methods to detect hidden content. **Applications Beyond Privacy** Steganography has applications beyond privacy and security. Some studies explore the use of steganography in medical imaging, copyright protection, and data hiding for various practical purposes. **The use of hardware acceleration, particularly GPUs, is gaining traction in multimedia steganography research.** This can significantly speed up the embedding and detection processes. **Ethical and Legal Considerations** Researchers and scholars are increasingly addressing the ethical and legal aspects of

steganography. Discussions on responsible usage and potential legal implications are growing in importance. Challenges and Future Directions A recurring theme in the literature is the identification of challenges and the proposal of future research directions. Researchers are continuously looking for ways to improve the field and address its limitations.

In conclusion, a literature survey of multimedia steganography reveals a dynamic and evolving field that incorporates various techniques, security concerns, and emerging trends. It highlights the importance of balancing data hiding capacity with perceptual quality, the ongoing battle between steganography and steganalysis, and the exploration of new application areas. Moreover, it underscores the ethical and legal considerations that researchers and practitioners need to address as steganography continues to advance.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

Capacity vs. Security Trade-off: Balancing the amount of hidden information with the level of security against detection is a fundamental challenge in steganography. Increasing capacity often leads to a higher chance of detection. Robustness to Attacks: Steganography techniques should be able to withstand various forms of attacks, such as compression, filtering, noise addition, and other common image processing operations. Security Against Advanced Statistical Attacks: Steganalysis techniques continue to improve, utilizing advanced statistical and machine learning methods. Designing steganographic algorithms that are robust against these sophisticated attacks remains an ongoing challenge.

Payload Integrity and Reliability: Ensuring that the hidden information is reliably extracted without corruption, even in the presence of minor distortions or lossy compression, is a critical concern. Multimedia Type Agnosticism: Many steganography techniques are designed for specific types of media. Developing methods that can work seamlessly across different types of media is an open problem. Adaptive and Dynamic Embedding Strategies Designing steganographic techniques that adaptively adjust the embedding process based on the content and context of the cover media is a challenging problem.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

Risk analysis in steganography involves identifying, assessing, and mitigating potential risks and vulnerabilities associated with the use of steganographic techniques. These risks can have significant implications for both security and privacy. Here's a breakdown of risk analysis in steganography:

Detection Risk is the primary risk in steganography is the potential detection of hidden information by adversaries or steganalysis techniques. Detection risks could lead to the exposure of sensitive data or the compromise of covert communication.

Misuse Risk there is a risk that steganography techniques can be misused for malicious purposes, such as concealing malware, distributing illegal content, or bypassing security controls.

Data Loss Risk when data is embedded within multimedia files, there's a risk of data loss or corruption during the embedding and extraction processes. If the embedded data is critical, its loss could have significant consequences.

Inadequate Security Risk If steganographic methods are not implemented securely, there is a risk that the hidden data can be easily extracted or compromised. Weak security measures can lead to unauthorized access to concealed information.

Privacy Risk In cases where steganography is used to protect privacy, there's a risk that the hidden data may be exposed, leading to privacy breaches.

Likelihood Evaluate the likelihood of each identified risk occurring. Factors that contribute to likelihood include the sophistication of potential adversaries, the strength of the steganographic method, and the prevalence of steganalysis techniques.

Impact Assess the potential impact of each risk. Consider the consequences of detection, misuse, data loss, or privacy breaches.

Vulnerability Analysis Identify vulnerabilities within the steganographic system itself. Vulnerabilities may include weak encryption, inadequate key management, or poor implementation of steganographic algorithms.

Steganographic Technique Selection Choose steganographic techniques that strike an appropriate balance between capacity and imperceptibility.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

Multimedia steganography applications, which involve hiding data within audio, image, or video files, may have slightly higher hardware requirements compared to general steganography software, as multimedia files can be larger and require more processing power. Here are the typical hardware requirements for running multimedia steganography applications.

Processor : A multi-core processor with a clock speed of at least 2 GHz is recommended. More powerful CPUs may be necessary for processing large multimedia files or using advanced steganographic techniques.

Memory : 8GB or more of RAM is recommended for working with multimedia files and steganography applications. Larger files or complex steganography methods may benefit from more RAM.

Storage: Adequate storage space for multimedia files is necessary. SSDs are preferred for faster read and write operations. Free space should accommodate the original multimedia files, the steganography software, and any output files.

Graphics Processing Unit: While most multimedia steganography software doesn't heavily rely on GPUs, certain advanced techniques, such as deep learning-based steganography, may take advantage of GPU acceleration.

Operating System: Ensure that the multimedia steganography software you plan to use is compatible with your operating system e.g., Windows, macOS, or Linux.

Display: A standard computer monitor is sufficient for interacting with multimedia steganography applications.

Internet Connection: An internet connection may be required for downloading and updating multimedia steganography software. It is good to know that the specific requirements or the can vary depending on the multimedia steganography application.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

The proposed system in this project signifies a groundbreaking departure from conventional methods for multimedia steganography, introducing an innovative and data that harnesses advanced technology and encryption/decryption methodologies.

Text Steganography: The data is concealed within the text. The key knowledge of this system is concealed behind the n th character in a phrase of the substance of the document. There are several techniques in a text document to conceal data. They are as follows Format Based Method and Random and Statistical Method.

A new approach of text Steganography *Fig 4.1* using ASCII values, this project aims to maximize the usage capacity in text Steganography. A new technique has been proposed to encrypt the text or content inside the images by using pixel values and ASCII values. Merging and dividing procedure is the heart of this method and gives the result. In this technique they give more importance to pixel values. Unlike other techniques, this model can embed or encrypt all 255 ASCII characters, this approach accessible only for lossless text formats. The proposed algorithm clients can hide more measures of information without altering the size of the cover picture. It implies changes reflected are almost insignificant. This project's limitation is to improve the efficiency of the steganographic algorithm for better performance. We suggest a modern methodology of format-based content Steganography, by utilizing the approach of two content or steganography approaches that are line-shifting and character or word-shifting strategies with a copy protection method. Mask the details or material in binary or digitized type rather than character format by utilizing the above methods and less than one bit is hidden in of line of cover text, so this approach has a strong hidden ability and less distortion in the cover text is provided by this process. Image Steganography conceals details by selecting the cover media or object as an image. In image Steganography, pixels qualities are considered. In digital Steganography to encode information, images are commonly used as a cover source because there are several bits present in digital image representation.

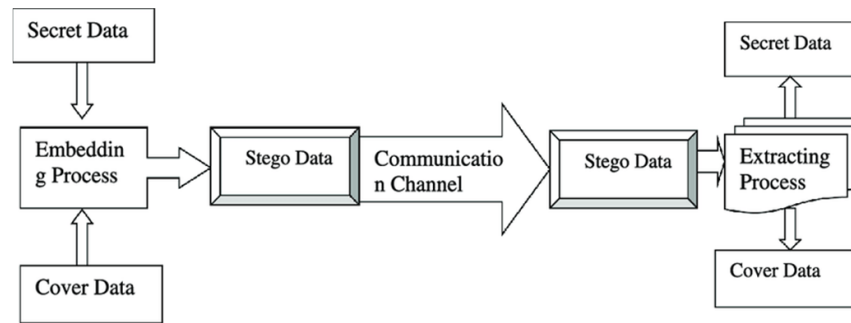


Fig 4.1: Image Steganography

Audio Steganography: In this strategy, the secret content *Fig 4.2* is embedded in digitized or binary audio. This results in small changes to the data, giving solutions to audio steganography which is available in the binary classification of the audio file. This includes the embedding of audio files with data. This approach hides the sound files in mp3, au, and wav, using specific audio Steganography methods. There are multiple techniques of Audio Steganography. These techniques are as follows: Spread Spectrum, Low Bit Encoding and Phase Coding.

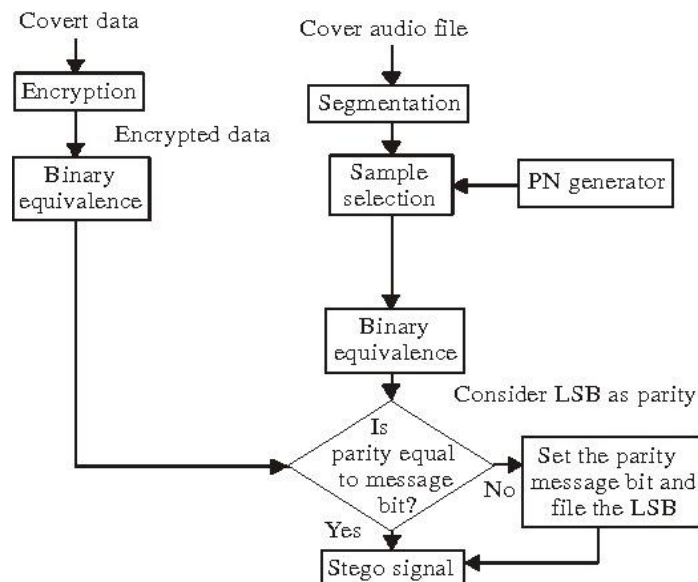


Fig 4.2: Audio Steganography

Video Steganography: This is a method of concealing content in binary or digital video format. Video consists of a set of images, which is usually used as the cover file for embedding the contents. Typically, by using i.e., discrete cosine transform algorithm, modify the values 9.667 to 10 that are used to mask the material in each entity of the images in the video *Fig 4.3* that the human eye cannot identify or locate. In video Steganography, the formats used are MP4 and MPE are used. In all of these

approaches the main motive is to conceal a secret message in another cover item. This may not be able to guess the presence of information inside cover media. It is necessary to apply the proper decoding method to retrieve the information.

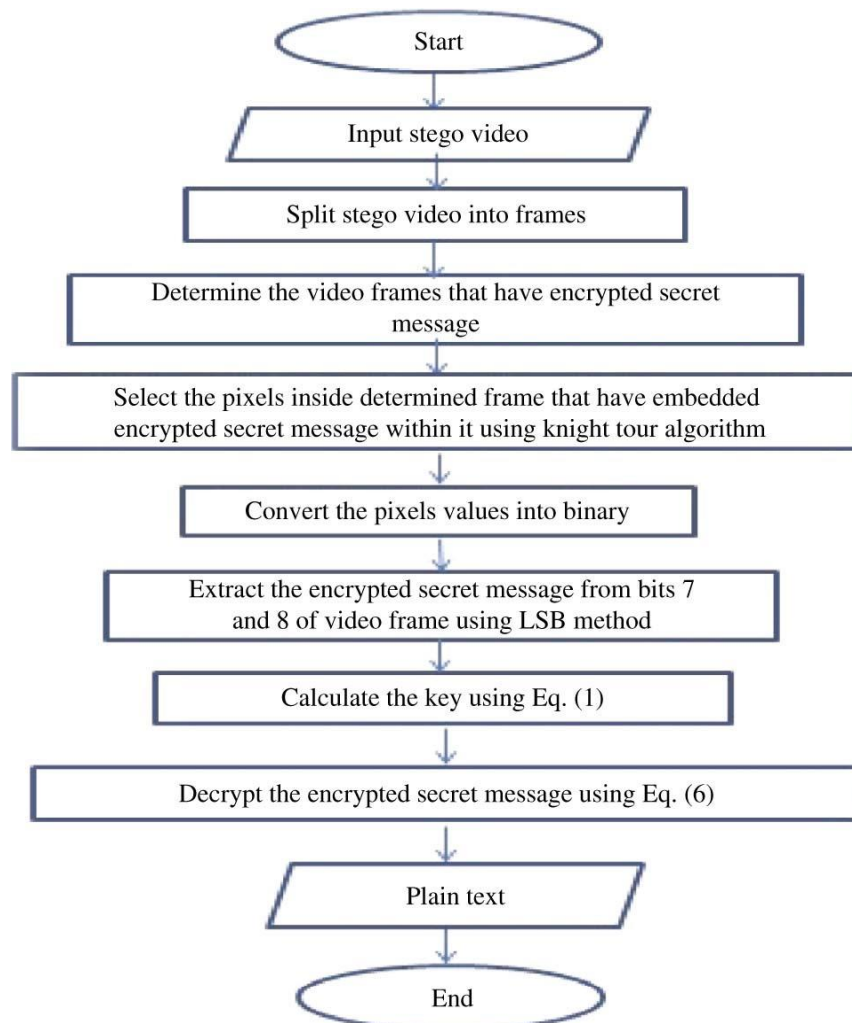


Fig 4.3: Video Steganography

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

Investigated Chaotic Maps: Chaotic systems are very sensitive to initial conditions also chaotic systems has positive Lyapunov exponent which leads orbital instability. Moreover, ergodicity property of chaotic systems is also used as a leveraging factor for hiding information or generating unpredictable random sequences. The some of the well-known chaotic mapping methods used in this study are:

- Logistics Map formula is given at Eq. (1)
- Tent Map formula is given at Eq. (2)
- Quadratic Map formula is given at Eq. (3)
- Bernoulli Map formula is given at Eq. (4)
- Sine Map formula is given at Eq. (5)
- Chebyshev Maps formula is given at Eq.(6)

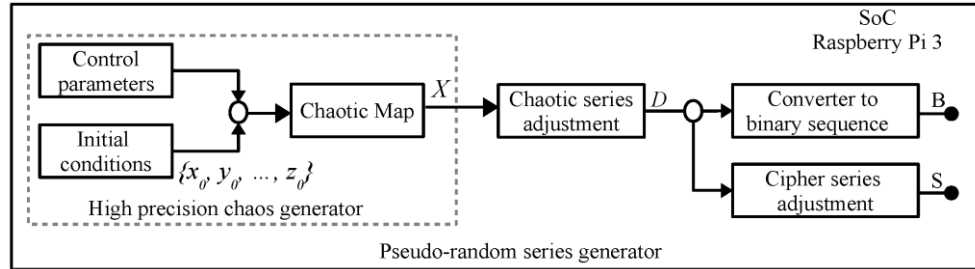


Fig 4.1.1: Chaotic mapping method

As it could be observed from given chaotic map equations are all effected by the previous states. Therefore, those maps are sensitive to initial conditions also guessing the next outcome of the map is difficult unless the initial conditions are known.

Evaluation Criteria: The authors of the study have determined the invisibility, load capacity and robustness criteria against steganalysis and attacks. These their metric is used to assess the steganographic systems. In their work at statistical indeterminacy is also accepted as requirement.

The Imperceptibility or in other word invisibility of hidden secret message is the cardinal and fundamental requirement. When the changes are noticed in the picture, the steganographic algorithm becomes void. Generally, is measured by the quality of the stego image one important metric.

In order to evaluate aforementioned metrics and also assess quality of the stego image Entropy, Correlation analysis, mean squared error and Peak Signal Noise Ratio are calculated. Equations form (7) to (10) are the given evaluation metrics:

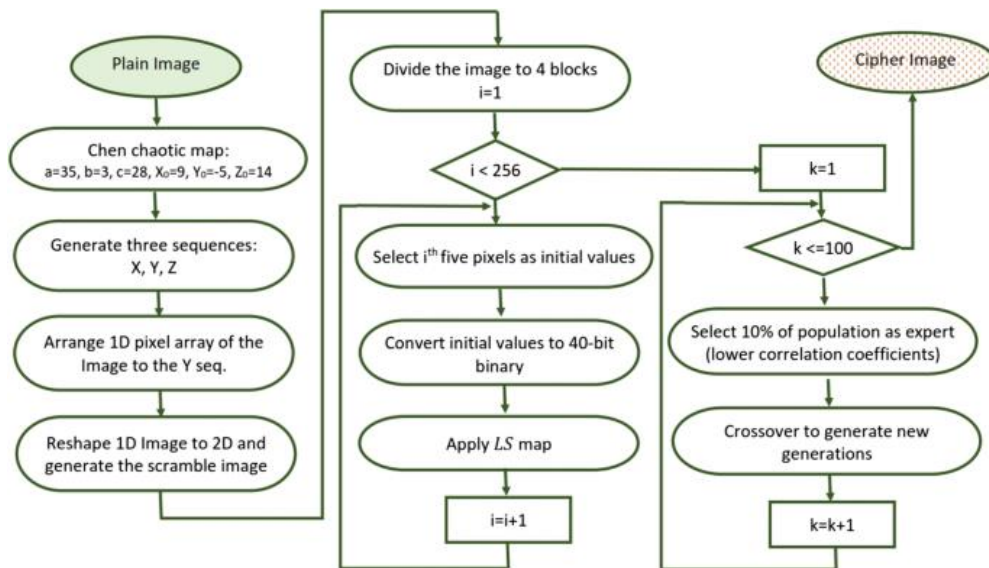


Fig 4.1.2: Chaotic mapping evaluation criteria

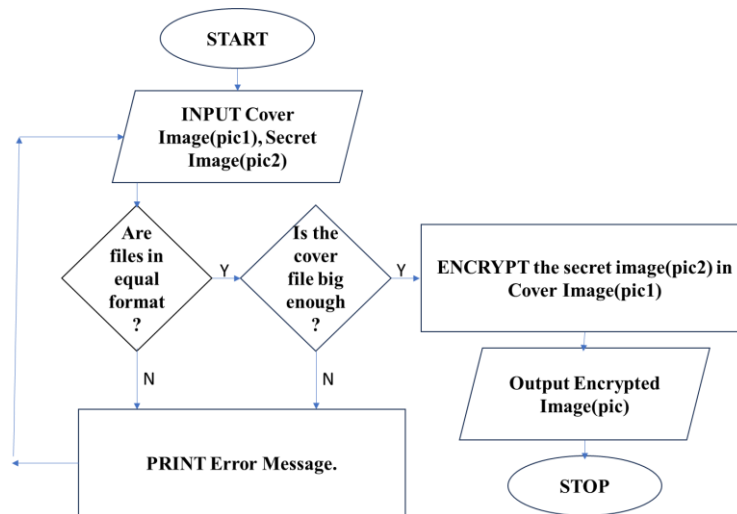


Fig 4.1.3 Flowchart of Encryption Process

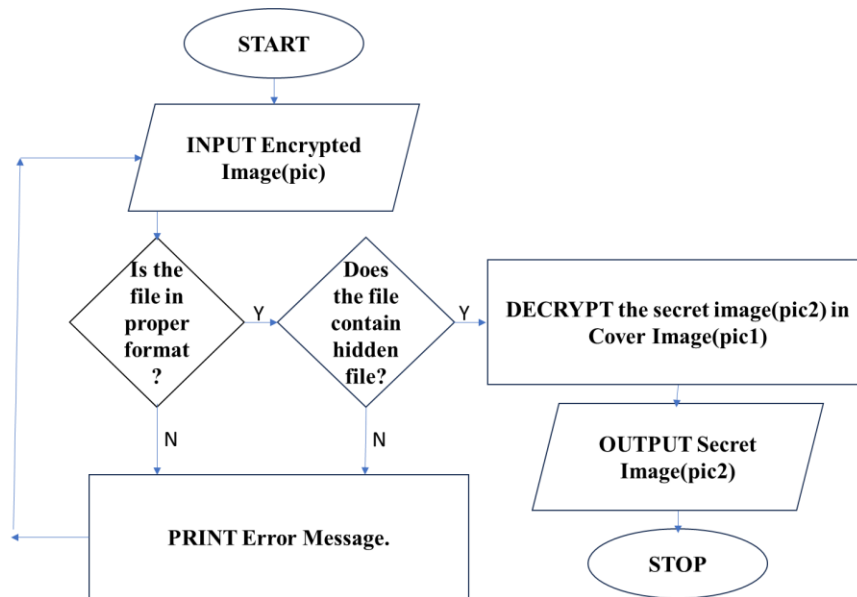


Fig 4.1.4 Flowchart of Decryption Process

Proposed Model and Evaluations: In the conventional LSB steganography, just LSB bit of the carrier image is changed with the secret message bit. At first Secret message is preprocessed as a bit vector.

Algorithm 1

Algorithm for Embedding Message

Inputs: Cover image C, Message M, Keys Arrays: Coefficients and initial starting point of Chaotic Maps Obtain Gzip of the Message M -> ZM

LIST ZM: Message Possible Locations

LIST : Cover Image Possible Locations TEMP:

Outputs: Stego Image

- 1- For j:0 to ZM
- 2- location=Generate Random Locations in the LIST ZM using Chaotic Map
- 3- Assign LIST ZM[location] message to TEMP
- 4- For i:0 to 7
- 5- Generate Random locations in the LIST using Chaotic Map
- 6- Remove the selected location from the LIST
- 7- Take mod 3 of Chaotic Location to Find the embedding channel
- 8- EMBED The LSB of the selected channel of the selected pixel with the TEMP[i] bit
- 9- End For
- 10- Remove the selected ZM[location] from the LISTZM
- 11- End For
- 12- Return Stego Image

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

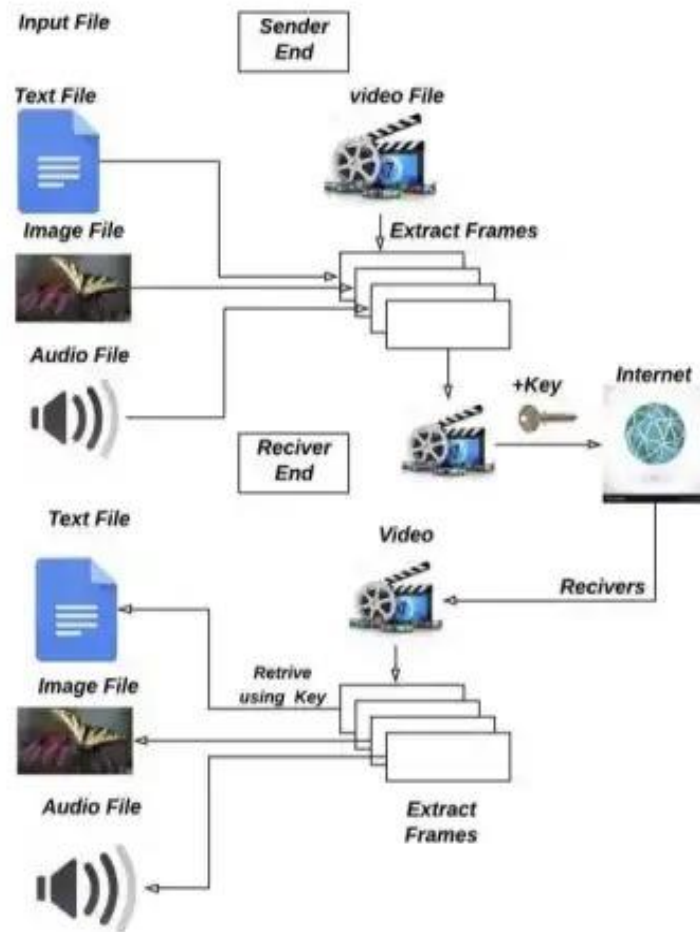


Fig 4.2: System Architecture

The system architecture *Fig 4.2* of a multimedia steganography application is a well-structured framework composed of several interconnected components, each with specific roles and responsibilities. The steganographic engine lies at the core, implementing various techniques for embedding and extracting data within multimedia files, ensuring that the original content remains visually or audibly intact. The multimedia file handler reads and writes these files, and the optional encryption and error correction components may be utilized to secure and maintain data integrity. Metadata management preserves metadata consistency, and format compatibility components ensure compatibility with various file formats. User preferences and settings, along with an output viewer/player, offer customization options and the ability to review the modified multimedia content.

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

The software for implementing and testing the proposed model/system for encryption /decryption using above methodology plays a pivotal role in the project's success. Testing software using Java typically involves using various testing frameworks and libraries to ensure the quality and reliability of Java applications. Here are some common approaches and tools for testing software in Java:

JUnit is one of the most popular Java testing frameworks for writing and running unit tests. It allows you to define test cases, assert expected outcomes, and automate the testing process. TestNG is another widely used testing framework that provides more advanced features compared to JUnit. It supports various testing strategies, including unit testing, integration testing, and end-to-end testing. Selenium is a popular open-source framework for automating web application testing. You can use Java to write Selenium tests to interact with web elements and validate web application behavior. Mockito is a Java mocking framework that helps you create mock objects for testing. It's useful for isolating the code under test and simulating interactions with dependencies. Cucumber is a behavior-driven development framework that allows you to write tests in a human-readable format. It's often used for acceptance and integration testing of Java applications. Test frameworks in Java Integrated Development Environments like Eclipse and IntelliJ IDEA have built-in support for running and debugging Java tests. You can create and execute test cases directly within your IDE. Continuous Integration tools like Jenkins, Travis CI, and CircleCI can be integrated with your Java projects to automate the testing process, run tests on each code commit, and generate reports. Load Testing For performance testing, tools like Apache JMeter can be used in combination with Java to simulate and analyze the behavior of an application under load.

Test Automation Frameworks: When building complex test suites or end-to-end tests, you can create custom test automation frameworks in Java to manage and organize your tests efficiently.

Table 1 Study of Various types of Chaotic Mapping Methods

IMAGE	Bernoulli	Logistic	Quadratic	Sine	Tent
Original Image Entropy	0.15658549505	0.15658549505	0.02025277832	0.15653749505	0.156549505
Carrier Image Entropy	0.15658588824	0.15658577751	0.15658630971	0.15658400236	0.156614799
Entropy Difference	-0.000000393196	-0.000005407587	-0.000008814662	0.000000492685	-0.0001652948
Correlation Difference	0.999998637121	0.999994765612	0.999995685202	0.999994634734	0.9999669674
MSE	0.020141156250	0.020353930664	0.020252778320	0.020379809570	0.0201386523
PNR	68.08986328575	68.04323001598	68.06607747841	68.03997485043	68.0906818610

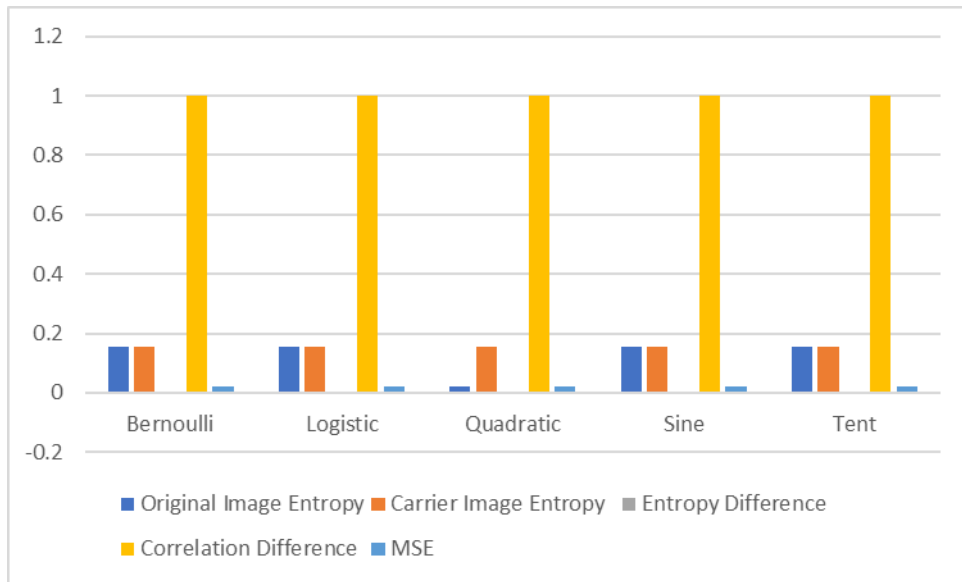


Fig 4.3.1 Comparative Study of Various types of Chaotic Mapping Methods

4.4 RESULTS & DISCUSSION

The introduction of an image in another image produces noise which makes it obvious for the steganalysts to decipher the enciphered media. The use of LSB Chaotic Mapping algorithm makes it harder for the programmers to find the pixels which have been modified to embed the secret image. Even if they are successful in doing so, it would be harder for them to join the pictures into something meaningful. This algorithm does not use regular pixel values which makes it unique.

A comprehensive and comparative study has been done in *Fig 4.3.1* with the original image and the enciphered image for the changes. The table depicts the usage of various available chaotic mapping algorithms and their different evaluation parameters. It also encompasses various error calculation methods and changes in the entropies according to the formulas and equations mentioned.

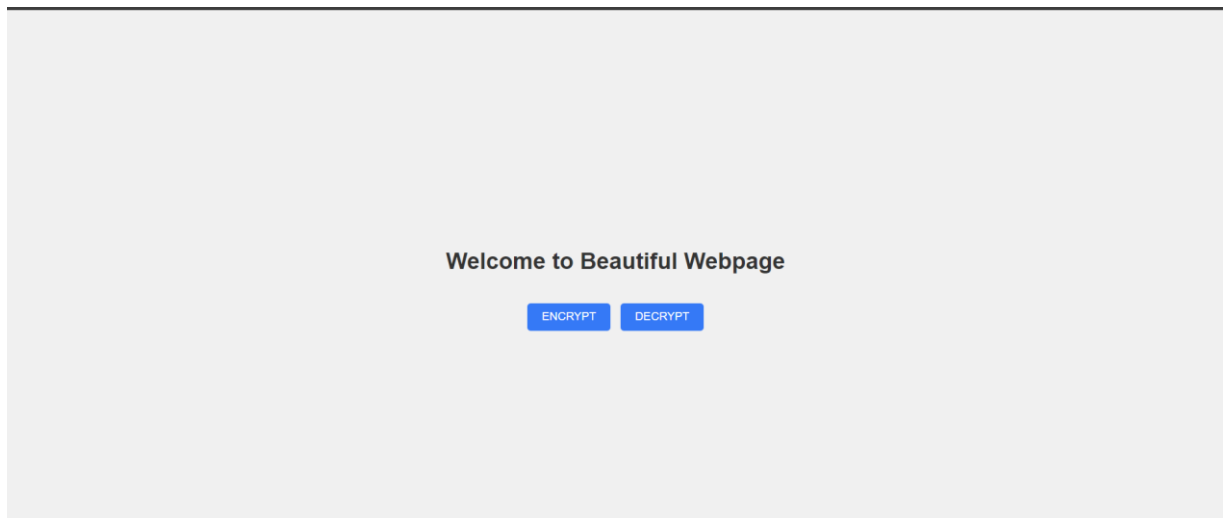


Fig 4.4.1 Welcome Page of Application when started

These days, data and information as shown in *Fig 4.4.2* are highly valued, and transmitting and receiving data securely is crucial. The methods utilized for secure transmission and private information protection are cryptography and steganography. Current methods combine LSB encoding *Fig 4.4.3* steganography with cryptographic algorithms for encryption. They also introduce several methods for LSB encoding using Chaotic Maps. In many eHealth applications, such as data management, identity theft prevention, storage, and retrieval, image security is crucial. This paper includes a brief explanation of a few well-known picture assaults that are used to test the performance of the suggested algorithms on images *Fig 4.4.4*. In order to encourage more study in this field, we have covered the cryptographic technique and its traits, types, prerequisites, working domain, and probable problems. Moreover, various performance measures for assessing algorithms have been shown. The paper presents a methodology for video steganography that uses text, audio, and image files. A carrier video is retrieved, and the frames that the audio, text, and image files can be inserted within are used to reconstruct the video. According to the mathematical model, the suggested architecture should perform.



Fig 4.4.2 Animal



Fig 4.4.3 Human



Fig 4.4.4 Embedded

A screenshot of a web application interface. The page has a blue header and a white body. In the center, there is a white box titled "Upload Files to Hide". Inside this box, there are two sections: "Upload Cover File" and "Upload Secret File". Each section has a "Choose File" button, a text input field showing "No file chosen", and a "Submit" button. Below these sections, there is a "Submit" button. At the bottom of the page, there are social media icons for Instagram, LinkedIn, Twitter, and Facebook, followed by a navigation menu with links for "Home", "Services", "About", "Terms", and "Privacy Policy". The footer text reads "Stegos © 2024".

Fig 4.4.5 Encryption Webpage

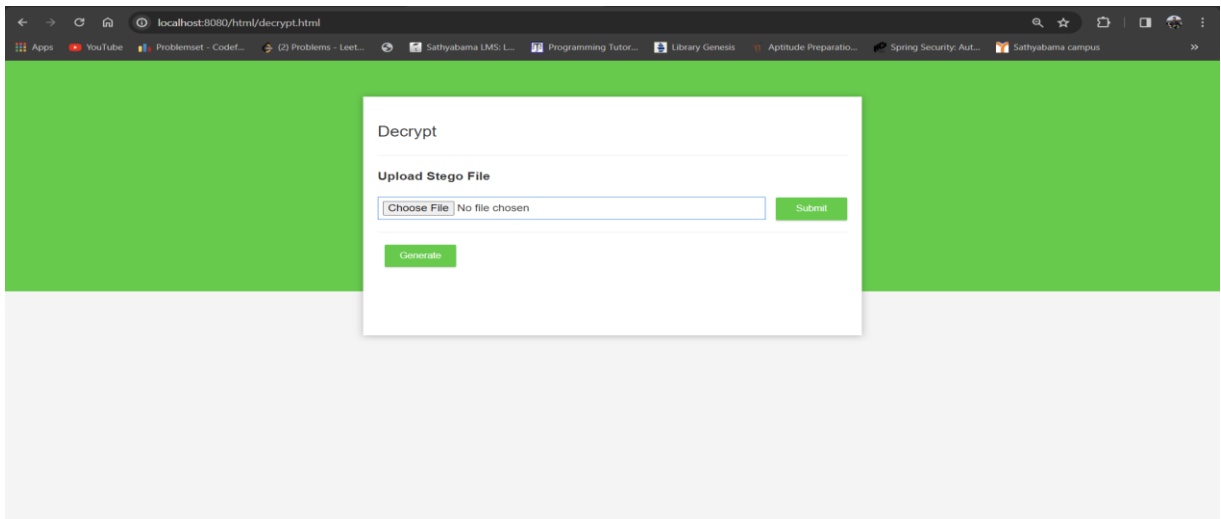


Fig 4.4.6 Decryption Webpage

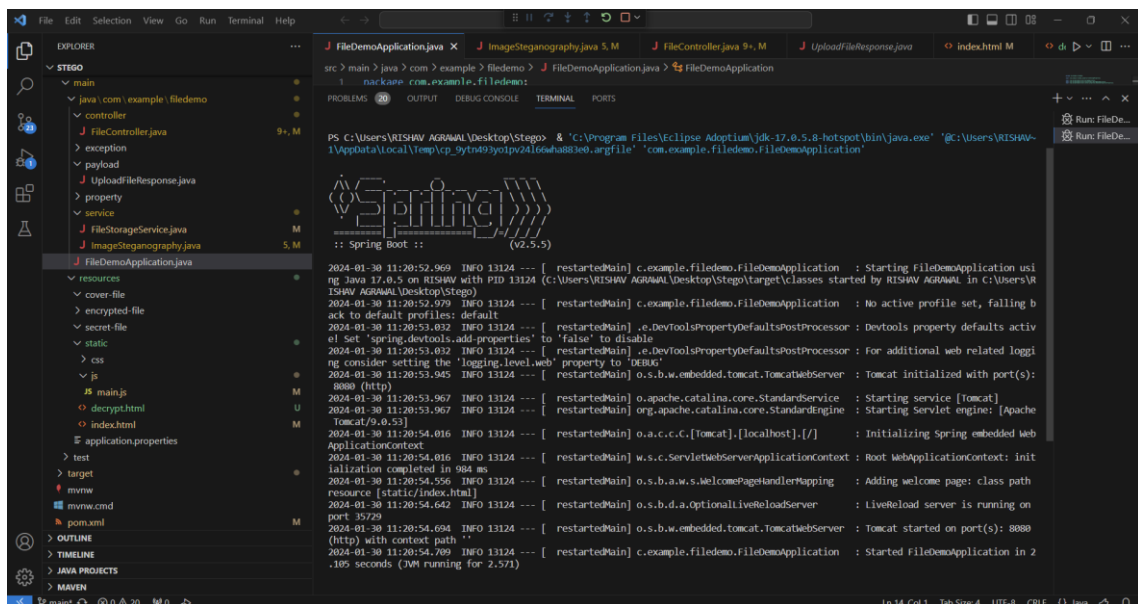


Fig 4.4.7 Spring Application Started

CHAPTER 5

IMPLEMENTATION DETAILS

Spring Boot's executable jars are ready-made for most popular cloud Platform-as-a-Service providers. These providers tend to require that you “bring your own container”. They manage application processes not Java applications specifically, so they need an intermediary layer that adapts *your* application to the *cloud*'s notion of a running process.

Two popular cloud providers, Heroku and Cloud Foundry, employ a “buildpack” approach. The buildpack wraps your deployed code in whatever is needed to *start* your application. It might be a JDK and a call to java, an embedded web server, or a full-fledged application server. A buildpack is pluggable, but ideally you should be able to get by with as few customizations to it as possible. This reduces the footprint of functionality that is not under your control. It minimizes divergence between development and production environments. Ideally, the application, like a Spring Boot executable jar, has everything that it needs to run packaged within it. By default, metadata about the running application as well as service connection information is to the application as environment variables for example: `$VCAP_SERVICES`. This architecture decision is due to Cloud Foundry's polyglot any language and platform can be supported as a buildpack nature. Process-scoped environment variables are language agnostic.

Environment variables do not always make for the easiest API, so Spring Boot automatically extracts them and flattens the data into properties that can be accessed through Spring's Environment abstraction, as shown in the below code. All Cloud Foundry properties are prefixed with vcap. You can use vcap properties to access application information such as the public URL of the application and service information such as database credentials.

```

import org.springframework.context.EnvironmentAware;
import org.springframework.core.env.Environment;
import org.springframework.stereotype.Component;

@Component
public class MyBean implements EnvironmentAware {

    private String instanceId;

    @Override
    public void setEnvironment(Environment environment) {
        this.instanceId = environment.getProperty("vcap.application.instance_id");
    }

    // ...

}

```

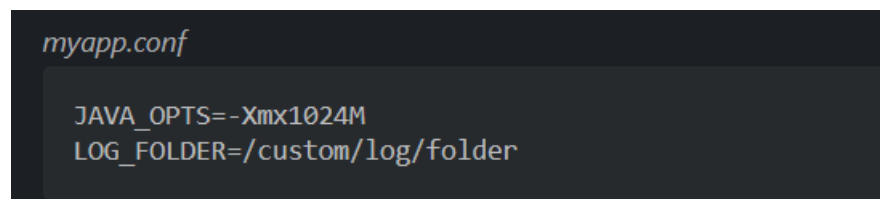
Fig 5.1 Deployment Code

5.1 INSTALLING SPRING BOOT APPLICATION

In addition to running Spring Boot applications by using `java -jar` directly, it is also possible to run them as `systemd`, `init.d` or Windows services. `systemd` is the successor of the System V `init` system and is now being used by many modern Linux distributions. Spring Boot applications can be launched by using `systemd` ‘service’ scripts. The user that runs the application, the PID file, and the console log file are managed by `systemd` itself and therefore must be configured by using appropriate fields in the ‘service’ script.

When executed as root, as is the case when root is being used to start an `init.d` service, the default executable script runs the application as the user specified in the `RUN_AS_USER` environment variable. When the environment variable is not set, the user who owns the jar file is used instead. You should never run a Spring Boot application as root, so `RUN_AS_USER` should never be root and your application’s jar file should never be owned by root. Instead, create a specific user to run your application and set the `RUN_AS_USER` environment variable or use `chown` to make it the owner of the jar file.

If root is used to control the application's service and you use a .conf file to customize its startup, the .conf file is read and evaluated by the root user. It should be secured accordingly. Use chmod so that the file can only be read by the owner and use chown to make root the owner. It often makes sense to customize elements of the start script as it is written into the jar file. For example, init.d scripts can provide a "description". Since you know the description up front and it need not change, you may as well provide it when the jar is generated. With the exception of JARFILE and APP_NAME, the settings listed in the preceding section can be configured by using a .conf file. The file is expected to be next to the jar file and have the same name but suffixed with .conf rather than .jar. For example, a jar named /var/myapp/myapp.jar uses the configuration file named /var/myapp/myapp.conf, as shown:



```
myapp.conf
JAVA_OPTS=-Xmx1024M
LOG_FOLDER=/custom/log/folder
```

Fig 5.1.1 Application Configuration

5.2 DEPLOYING SPRING BOOT APPLICATION

If you are running your application from a container, you can use an executable jar, but it is also often an advantage to explode it and run it in a different way. Certain PaaS implementations may also choose to unpack archives before they run. For example, Cloud Foundry operates this way. One way to run an unpacked archive is by starting the appropriate launcher, as follows:

```
$ jar -xf myapp.jar
```

```
$ java org.springframework.boot.loader.launch.JarLauncher
```

This is actually slightly faster on startup. Depending on the size of the jar than running from an unexploded archive. After startup, you should not expect any differences. Once you have unpacked the jar file, you can also get an extra boost to startup time by running the app with its "natural" main method instead of the JarLauncher.

```
$ jar -xf myapp.jar
```

```
$ java -cp "BOOT-INF/classes:BOOT-INF/lib/*" com.example.MyApplication
```

It's beneficial for the startup time to run your application using the AOT generated initialization code. First, you need to ensure that the jar you are building includes AOT generated code. For Maven, this means that you should build with `-Pnative` to activate the native profile:

```
$ mvn -Pnative package
```

For Gradle, you need to ensure that your build includes the `org.springframework.boot.aot` plugin. When the JAR has been built, run it with `spring.aot.enabled` system property set to true.

```
$ java -Dspring.aot.enabled=true -jar myapplication.jar
```

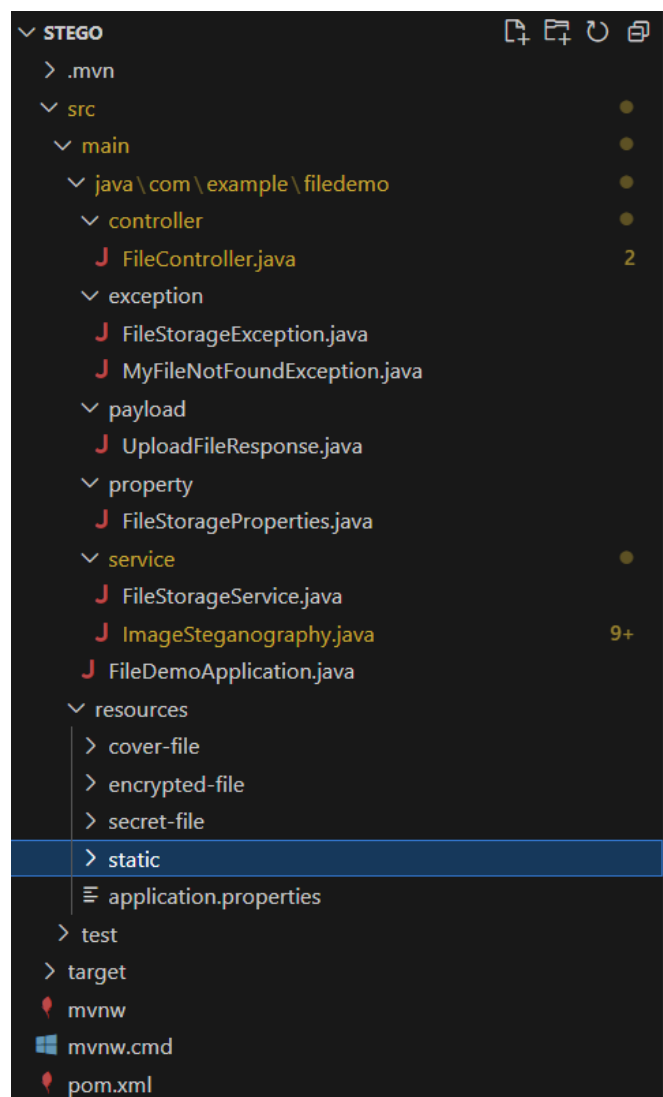


Fig 5.2.1: Directory Structure of Project

CHAPTER 7

CONCLUSION

In conclusion, the research on multimedia steganography using the LSB chaotic mapping method introduces a robust and secure approach for concealing messages within images. The algorithm utilizes chaotic maps to enhance security, incorporates Gzip compression for efficient data handling, and employs LSB embedding in color channels to minimize visual impact. Extensive testing validates the method's effectiveness, suggesting it as a promising technique for secure information concealment in multimedia applications. Further research could optimize parameters and explore alternative chaotic mapping methods, ensuring the continued evolution of reliable and resilient steganographic techniques.

In conclusion, the LSB chaotic mapping method for multimedia steganography stands as a promising and viable technique for secure information concealment. As digital communication continues to evolve, the presented approach contributes to the ongoing efforts in developing robust and reliable methods for covert data transmission in multimedia environments.

7.1 FUTURE WORK

The project opens several avenues for further enhancement and expansion. The introduction of multiplayer functionality stands as a significant opportunity, promising to add a new dimension of interaction and competitiveness to the game. Expanding the game's environments to include a wider variety of settings will not only enhance the visual experience but also enrich the gameplay, offering more diverse challenges and narratives. Additionally, continuous optimization for various applications platforms remains a priority, ensuring that the game is accessible and delivers a high-quality experience.

REFERENCES

- [1] Artz, D. 2021. Digital steganography: Hiding data within data. *IEEE Internet Computing Journal*, 5(3):75-80.
- [2] J. Fridrich, M. Goljan and Rui Du, Detecting LSB steganography in color, and gray-scale images, *IEEE Multimedia*, vol. 8, no. 4, pp. 22- 28, Oct.-Dec. 2021, doi: 10.1109/93.959097
- [3] C. Kurak, J. McHugh, A cautionary note on image downgrading, in: *Proceedings of the IEEE 8th Annual Computer Security Applications Conference*, 30 November–4 December, 2022, pp. 153–159.
- [4] W. Bender, W. Butera, D. Gruhl, R. Hwang, F.J. Paiz, S. Pogreb, Applications for data hiding, *IBM Systems Journal* 39 (3&4) (2022)547–568.
- [5] Y. Chen, H. Wang, H. Wu, Z. Wu, T. Li, and A. Malik, “Adaptive video data hiding through cost assignment and STCs,” *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1320–1335, Jun. 2021.
- [6] V. Holub and J. Fridrich, “Designing steganographic distortion using directional filters,” in *Proc. IEEE 4th Int. Workshop Inf. Forensics Secur.*, 2020, pp. 234–239.
- [7] Ren T, Kotera M, Ogawa T, et al. Text-Guided Style Transfer-based Image Manipulation Using Multimodal Generative Models. *IEEE Access*, 2021, PP(99):1-1.
- [8] Taburet T, Bas P, Sawaya W, et al. Natural Steganography in JPEG Domain with a Linear Development Pipeline. *IEEE Transactions on Information Forensics and Security*, 2020, PP(99):1-1.
- [9] Petitcolas FAP, et al. “Information hiding-a survey”. *Proceedings of the IEEE*, 87(7), 1062- 1078, 2020.
- [10] S. Xu, et al. “No-reference/Blind Image Quality Assessment: A Survey,” *IETE Tech. Rev.* (Institution Electron. Telecommun. Eng. India), vol. 34, pp. 223–245, 2017.
- [11] S.Thenmozhi,M.Chandrasekaran, “Novel approach for image steganography based on integer wavelet transform,” 2020 *IEEE International Conference on Computer Intelligence and Computing Research* ,2012,pp.1-5,doi: 10.1109/ICCIC.2020.651030
- [12] C. Kurak, J. McHugh, A cautionary note on image downgrading, in: *Proceedings of the IEEE 8th Annual Computer Security Applications Conference*, 30 November–4 December, 2022, pp. 153–159.

- [13] W. Bender, W. Butera, D. Gruhl, R. Hwang, F.J. Paiz, S. Pogreb, Applications for data hiding, IBM Systems Journal 39 (3&4) (2022)547–568.
- [14] Y. Chen, H. Wang, H. Wu, Z. Wu, T. Li, and A. Malik, “Adaptive video data hiding through cost assignment and STCs,” IEEE Trans. Dependable Secure Comput., vol. 18, no. 3, pp. 1320–1335, Jun. 2021.
- [15] V. Holub and J. Fridrich, “Designing steganographic distortion using directional filters,” in Proc. IEEE 4th Int. Workshop Inf. Forensics Secur., 2020, pp. 234–239.
- [16] Ren T, Kotera M, Ogawa T, et al. Text-Guided Style Transfer-based Image Manipulation Using Multimodal Generative Models. IEEE Access, 2021, PP(99):1-1.
- [17] Taburet T, Bas P, Sawaya W, et al. Natural Steganography in JPEG Domain with a Linear Development Pipeline. IEEE Transactions on Information Forensics and Security, 2020, PP(99):1-1.
- [18] Petitcolas FAP, et al. “Information hiding-a survey”. Proceedings of the IEEE, 87(7), 1062- 1078, 2020.
- [19] S. Xu, et al. “No-reference/Blind Image Quality Assessment: A Survey,” IETE Tech. Rev. (Institution Electron. Telecommun. Eng. India), vol. 34, pp. 223–245, 2017.
- [20] S.Thenmozhi,M.Chandrasekaran, “Novel approach for image steganography based on integer wavelet transform,” 2020 IEEE International Conference on Computer Intelligence and Computing Research ,2012,pp.1-5,doi: 10.1109/ICCIC.2020.651030

APPENDIX

A. SOURCE CODE

// FileDemoApplication.java

```
package com.example.filedemo;

import com.example.filedemo.property.FileStorageProperties;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.context.properties.EnableConfigurationProperties;

@SpringBootApplication
@EnableConfigurationProperties({
    FileStorageProperties.class,
})

public class FileDemoApplication {

    public static void main(String[] args) {

        SpringApplication.run(FileDemoApplication.class, args);

    }

}
```

// FileStorageService.java

```
package com.example.filedemo.service;

import com.example.filedemo.exception.FileStorageException;
import com.example.filedemo.exception.MyFileNotFoundException;
import com.example.filedemo.property.FileStorageProperties;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.Resource;
import org.springframework.core.io.UrlResource;
```

```

import org.springframework.stereotype.Service;

import org.springframework.util.StringUtils;

import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;

import java.net.MalformedURLException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardCopyOption;

@Service

public class FileStorageService {

    private Path coverFileStorageLocation;

    private Path secretFileStorageLocation;

    private Path encryptedFileStorageLoacation;

    @Autowired

    public FileStorageService(FileStorageProperties fileStorageProperties) {

        this.coverFileStorageLocation = Paths.get("spring-boot-file-upload-download-
rest-api-example/src/main/resources/cover-file");

        this.secretFileStorageLocation = Paths.get("spring-boot-file-upload-
download-rest-api-example/src/main/resources/secret-file");

        this.encryptedFileStorageLoacation = Paths.get("spring-boot-file-upload-
download-rest-api-example/src/main/resources/encrypted-file");

        try {

            Files.createDirectories(this.coverFileStorageLocation);

            Files.createDirectories(this.secretFileStorageLocation);

            Files.createDirectories(this.encryptedFileStorageLoacation);

```

```

    } catch (Exception ex) {

        throw new FileStorageException("Could not create the directory where the
        uploaded files will be stored.", ex);

    }
}

public String storeCoverFile(MultipartFile file) {

    // Normalize file name

    String fileName = StringUtils.cleanPath(file.getOriginalFilename());

    try {

        // Check if the file's name contains invalid characters

        if(fileName.contains("..")) {

            throw new FileStorageException("Sorry! Filename contains invalid path
            sequence " + fileName);

        }

        // Copy file to the target location (Replacing existing file with the same
        name)

        Path targetLocation = this.coverFileStorageLocation.resolve(fileName);

        Files.copy(file.getInputStream(), targetLocation,
        StandardCopyOption.REPLACE_EXISTING);

        return fileName;

    } catch (IOException ex) {

        throw new FileStorageException("Could not store file " + fileName + ".
        Please try again!", ex);

    }
}

public String storeSecretFile(MultipartFile file) {

    // Normalize file name

```

```

String fileName = StringUtils.cleanPath(file.getOriginalFilename());

try {

    // Check if the file's name contains invalid characters

    if(fileName.contains("..")) {

        throw new FileStorageException("Sorry! Filename contains invalid path
sequence " + fileName);

    }

    // Copy file to the target location (Replacing existing file with the same
name)

    Path targetLocation = this.secretFileStorageLocation.resolve(fileName);

    Files.copy(file.getInputStream(), targetLocation,
StandardCopyOption.REPLACE_EXISTING);

    return fileName;

} catch (IOException ex) {

    throw new FileStorageException("Could not store file " + fileName + ".
Please try again!", ex);

}

}

public Resource loadFileAsResource(String fileName) {

    try {

        Path filePath =
this.coverFileStorageLocation.resolve(fileName).normalize();

        Resource resource = new UrlResource(filePath.toUri());

        if(resource.exists()) {

            return resource;

        } else {

            throw new MyFileNotFoundException("File not found " + fileName);

```

```

        }

    } catch (MalformedURLException ex) {

        throw new MyFileNotFoundException("File not found " + fileName, ex);

    }

}

}

```

// ImageSteganography.java

```

package com.example.filedemo.service;

import org.opencv.core.Core;

import org.opencv.core.CvType;

import org.opencv.core.Mat;

import org.opencv.imgcodecs.Imgcodecs;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.example.filedemo.exception.FileStorageException;

import com.example.filedemo.property.FileStorageProperties;

import java.io.File;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.Random;

@Service

public class ImageSteganography {

    private String input1;

    private String input2;

    private String output;

```

```

public ImageSteganography() {

    String coverFile = "spring-boot-file-upload-download-rest-api-
example/src/main/resources/cover-file/";

    String secretFile = "spring-boot-file-upload-download-rest-api-
example/src/main/resources/secret-file/";

    //System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

    nu.pattern.OpenCV.loadLocally();

    File coverFolder = new File("spring-boot-file-upload-download-rest-api-
example/src/main/resources/cover-file/");

    File secretFolder = new File("spring-boot-file-upload-download-rest-api-
example/src/main/resources/secret-file/");

    if(coverFolder.listFiles().length != 0) {

        input1 = coverFile + coverFolder.listFiles()[0].getName();

    }

    if(secretFolder.listFiles().length != 0) {

        input2 = secretFile + secretFolder.listFiles()[0].getName();

    }

    output = "spring-boot-file-upload-download-rest-api-
example/src/main/resources/encrypted-file/encrypted-file.png";

}

public String getCoverFileLocation() {

    return this.input1;

}

public String getSecretFileLocation() {

    return this.input2;

}

```

```

public String getEncryptedFileLocation() {
    return this.output;
}

public void encode(String input1, String input2, String output) {
    Imgcodecs image1 = new Imgcodecs();
    Imgcodecs image2 = new Imgcodecs();
    Mat mat1 = Imgcodecs.imread(input1);
    Mat mat2 = Imgcodecs.imread(input2);
    for(int i=0; i<mat1.rows(); i++) {
        for(int j=0; j<mat1.cols(); j++) {
            double[] data2 = mat2.get(i, j);
            double[] data1 = mat1.get(i, j);
            for(int k=0; k<3; k++) {
                int temp1 = (int)(data1[k]);
                int temp2 = (int)(data2[k]);
                //System.out.println(i + " " + j + " " + k + " " + temp1 + " " + temp2);
                String s1 = Integer.toBinaryString(temp1);
                String s2 = Integer.toBinaryString(temp2);
                s1 = ("00000000" + s1).substring(s1.length());
                s2 = ("00000000" + s2).substring(s2.length());
                String s = s1.substring(0, 4) + s2.substring(0, 4);
                data1[k] = (double)(Integer.parseInt(s, 2));
                //System.out.println(i + " " + j + " " + k + " " + data1[k]);
            }
            mat1.put(i, j, data1);
        }
    }
}

```

```

    }

    Imgcodecs.imwrite(output, mat1);
}

public void decode(String path) {
    Imgcodecs image = new Imgcodecs();
    Imgcodecs image1 = new Imgcodecs();
    Imgcodecs image2 = new Imgcodecs();
    Mat mat = image.imread(path);
    Mat mat1 = new Mat(mat.rows(), mat.cols(), CvType.CV_64FC3);
    Mat mat2 = new Mat(mat.rows(), mat.cols(), CvType.CV_64FC3);
    for(int i=0; i<mat.rows(); i++) {
        for(int j=0; j<mat.cols(); j++) {
            double[] data = mat.get(i, j);
            double[] data1 = new double[3];
            double[] data2 = new double[3];
            for(int k=0; k<3; k++) {
                int temp = (int)(data[k]);
                String s = Integer.toBinaryString(temp);
                s = ("00000000" + s).substring(s.length());
                //System.out.println(i + " " + j + " " + k + " " + s);
                String s1 = s.substring(0, 4) + returnRandom();
                String s2 = s.substring(4) + returnRandom();
                //System.out.println(i + " " + j + " " + k + " " + s1 + " " + s2);
                data1[k] = (double)(Integer.parseInt(s1, 2));
                data2[k] = (double)(Integer.parseInt(s2, 2));
                //System.out.println(data1[k] + " " + data2[k]);
            }
        }
    }
}

```

```

        }

        mat1.put(i, j, data1);

        mat2.put(i, j, data2);

    }

}

Imgcodecs.imwrite(input1 + "cover.png", mat1);

Imgcodecs.imwrite(input2 + "secret.png", mat2);

}

public String returnRandom() {

    Random random = new Random();

    String ch = String.valueOf((char)(random.nextInt(2)+48));

    return String.valueOf(ch+ch+ch+ch);

}

}

```

// FileStorageProperties.java

```

package com.example.filedemo.property;

import org.springframework.boot.context.properties.ConfigurationProperties;

@ConfigurationProperties(prefix = "cover")

public class FileStorageProperties {

    private String uploadDir;

    public String getUploadDir() {

        return uploadDir;

    }

    public void setUploadDir(String uploadDir) {

        this.uploadDir = uploadDir;

    }

}

```

```

}

// UploadFileResponse.java

package com.example.filedemo.payload;

public class UploadFileResponse {

    private String fileName;

    private String fileDownloadUri;

    private String fileType;

    private long size;

    public UploadFileResponse(String fileName, String fileDownloadUri, String
        fileType, long size) {

        this.fileName = fileName;

        this.fileDownloadUri = fileDownloadUri;

        this.fileType = fileType;

        this.size = size;

    }

    public String getFileName() {

        return fileName;

    }

    public void setFileName(String fileName) {

        this.fileName = fileName;

    }

    public String getFileDownloadUri() {

        return fileDownloadUri;

    }

    public void setFileDownloadUri(String fileDownloadUri) {

        this.fileDownloadUri = fileDownloadUri;

    }

}

```

```

    }

    public String getFileType() {
        return fileType;
    }

    public void setFileType(String fileType) {
        this.fileType = fileType;
    }

    public long getSize() {
        return size;
    }

    public void setSize(long size) {
        this.size = size;
    }
}

// FileStorageException.java

package com.example.filedemo.exception;

public class FileStorageException extends RuntimeException {
    public FileStorageException(String message) {
        super(message);
    }

    public FileStorageException(String message, Throwable cause) {
        super(message, cause);
    }
}

// MyFileNotFoundException.java

package com.example.filedemo.exception;

```

```

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.NOT_FOUND)

public class MyFileNotFoundException extends RuntimeException {

    public MyFileNotFoundException(String message) {

        super(message);

    }

    public MyFileNotFoundException(String message, Throwable cause) {

        super(message, cause);

    }

}

```

// FileController.java

```

package com.example.filedemo.controller;

import com.example.filedemo.payload.UploadFileResponse;

import com.example.filedemo.service.FileStorageService;

import com.example.filedemo.service.ImageSteganography;

import org.apache.tomcat.util.http.fileupload.FileItem;

import org.apache.tomcat.util.http.fileupload.IOUtils;

import org.apache.tomcat.util.http.fileupload.disk.DiskFileItem;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.core.io.Resource;

import org.springframework.http.HttpHeaders;

import org.springframework.http.MediaType;

```

```

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.*;

import org.springframework.web.multipart.MultipartFile;

import org.springframework.web.multipart.commons.CommonsMultipartFile;

import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import java.io.*;

import java.io.FileInputStream;

import javax.servlet.http.HttpServletRequest;

import java.io.IOException;

import java.io.File;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

@RestController

public class FileController {

    private static final Logger logger =
        LoggerFactory.getLogger(FileController.class);

    @Autowired

    private FileStorageService fileStorageService;

    @Autowired

    private ImageSteganography imageSteganography;

    @PostMapping("/uploadCoverFile")

    public UploadFileResponse uploadCoverFile(@RequestParam("file")
        MultipartFile file) {

        String fileName = fileStorageService.storeCoverFile(file);

```

```

        String fileDownloadUri =
ServletUriComponentsBuilder.fromCurrentContextPath()

        .path("/downloadFile/")

        .path(fileName)

        .toUriString();

return new UploadFileResponse(fileName, fileDownloadUri,

        file.getContentType(), file.getSize());

}

@PostMapping("/uploadSecretFile")

public UploadFileResponse uploadSecretFile(@RequestParam("file")
MultipartFile file) {

    String fileName = fileStorageService.storeSecretFile(file);

    String fileDownloadUri =
ServletUriComponentsBuilder.fromCurrentContextPath()

        .path("/downloadFile/")

        .path(fileName)

        .toUriString();

return new UploadFileResponse(fileName, fileDownloadUri,

        file.getContentType(), file.getSize());

}

@GetMapping("/getEncryptedImage")

public UploadFileResponse getEncryptedFile(HttpServletRequest request) {

    String input1 = imageSteganography.getCoverFileLocation();

    String input2 = imageSteganography.getSecretFileLocation();

    String output = imageSteganography.getEncryptedFileLocation();

    imageSteganography.encode(input1, input2, output);

```

```

File file = new File(output);

FileItem fileItem = null;

try {

    fileItem = new DiskFileItem("mainFile",
Files.probeContentType(file.toPath()), false, file.getName(), (int) file.length(),
file.getParentFile());

    InputStream input = new FileInputStream(file);

    OutputStream os = fileItem.getOutputStream();

    IOUtils.copy(input, os);

    // Or faster..

    // IOUtils.copy(new FileInputStream(file), fileItem.getOutputStream());

} catch (IOException ex) {

    // do something.

}

MultipartFile multipartFile = new
CommonsMultipartFile((org.apache.commons.fileupload.FileItem) fileItem);

String fileDownloadUri =
ServletUriComponentsBuilder.fromCurrentContextPath()

    .path("/downloadFile/")

    .path(output)

    .toUriString();

return new UploadFileResponse(output, fileDownloadUri,
multipartFile.getContentType(), multipartFile.getSize());

}

@GetMapping("/downloadFile/{fileName:.+}")

public ResponseEntity<Resource> downloadFile(@PathVariable String

```

```

fileName, HttpServletRequest request) {

    // Load file as Resource

    Resource resource = fileStorageService.loadFileAsResource(fileName);

    // Try to determine file's content type

    String contentType = null;

    try {

        contentType =
request.getServletContext().getMimeType(resource.getFile().getAbsolutePath(
));

        } catch (IOException ex) {

            logger.info("Could not determine file type.");

        }

        // Fallback to the default content type if type could not be determined

        if(contentType == null) {

            contentType = "application/octet-stream";

        }

        return ResponseEntity.ok()

            .contentType(MediaType.parseMediaType(contentType))

            .header(HttpHeaders.CONTENT_DISPOSITION, "attachment;
filename=\"" + resource.getFilename() + "\"")

            .body(resource);

    }

}

// FileDemoApplicationTests.java
package com.example.testingweb;
import org.junit.jupiter.api.Test;

```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
import org.springframework.boot.test.web.client.TestRestTemplate;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.test.web.server.LocalServerPort;
import static org.assertj.core.api.Assertions.assertThat;
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
class HttpRequestTest {
    @LocalServerPort
    private int port;
    @Autowired
    private TestRestTemplate restTemplate;
    @Test
    void greetingShouldReturnDefaultMessage() throws Exception {
        assertThat(this.restTemplate.getForObject("http://localhost:" + port + "/",
            String.class)).contains("Hello, World");
    }
}

// application.properties

## MULTIPART (MultipartProperties)

# Enable multipart uploads

spring.servlet.multipart.enabled=true

# Threshold after which files are written to disk.

spring.servlet.multipart.file-size-threshold=2KB

# Max file size.

spring.servlet.multipart.max-file-size=200MB

# Max Request Size

spring.servlet.multipart.max-request-size=215MB

```

File Storage Properties

Please change this to the path where you want the uploaded files to be stored.

*file.upload-dir=spring-boot-file-upload-download-rest-api-
example/src/main/resources/cover-file*

*secret.file.upload-dir=spring-boot-file-upload-download-rest-api-
example/src/main/resources/secret-file*

// index.html

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0,
minimum-scale=1.0">

<title>Steganography</title>

<link rel="stylesheet" href="css/main.css" />

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.1.3/css/bootstrap.min.css">

<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/ionicons/2.0.1/css/ionicons.min.css
">

<link rel="stylesheet" href="css/style.css">

</head>

<body>

<noscript>

<h2>Sorry! Your browser doesn't support Javascript</h2>

</noscript>

<div class="upload-container">

<div class="upload-header">

```

    <h2>Upload Files to Hide</h2>

</div>

<div class="upload-content">

    <div class="cover-upload">

        <h4>Upload Cover File</h3>

        <form id="coverUploadForm" name="coverUploadForm">

            <input id="coverFileUploadInput" type="file" name="file"
class="file-input" accept=".png, .jpg, .jpeg, .bmp" required />

            <button type="submit" class="primary submit-
btn">Submit</button>

        </form>

        <div class="upload-response">

            <div id="coverFileUploadError"></div>

            <div id="coverFileUploadSuccess"></div>

        </div>

    </div>

    <br>

    <br>

    <div class="secret-upload">

        <h4>Upload Secret File</h3>

        <form id="secretUploadForm" name="secretUploadForm">

            <input id="secretFileUploadInput" type="file" name="files"
class="file-input" accept=".png, .jpg, .jpeg, .bmp" required />

            <button type="submit" class="primary submit-
btn">Submit</button>

        </form>

        <div class="upload-response">

```

```

        <div id="secretFileUploadError"></div>

        <div id="secretFileUploadSuccess"></div>

    </div>

</div>

<br><br>

<div class="generate-image">

    <form id="encryptImage" name="encryptImage">

        <input type="button" name="generate" value="Generate"
class="primary submit-btn" onclick="generateImage()">

    </form>

    <div class="upload-response">

        <div id="encryptedFileDownloadError"></div>

        <div id="encryptedFileDownloadSuccess"></div>

    </div>

</div>

</div>

</div>

<div class="footer-basic">

    <footer>

        <div class="social"><a href="#"><i class="icon ion-social-
instagram"></i></a><a href="#"><i class="icon ion-social-linkedin"></i></a><a
href="#"><i class="icon ion-social-twitter"></i></a><a href="#"><i class="icon
ion-social-github"></i></a></div>

        <ul class="list-inline">

            <li class="list-inline-item"><a href="#">Home</a></li>

            <li class="list-inline-item"><a href="#">Services</a></li>

            <li class="list-inline-item"><a href="#">About</a></li>

```

```

        <li class="list-inline-item"><a href="#">Terms</a></li>

        <li class="list-inline-item"><a href="#">Privacy Policy</a></li>

    </ul>

    <p class="copyright">Stegos © 2024</p>

</footer>

</div>

<script src="js/main.js" ></script>

<script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.1.3/js/bootstrap.bundle.min.js"></script>

</body>

</html>

// main.js

'use strict';

var coverUploadForm = document.querySelector('#coverUploadForm');
var coverFileUploadInput = document.querySelector('#coverFileUploadInput');
var coverFileUploadError = document.querySelector('#coverFileUploadError');
var coverFileUploadSuccess =
    document.querySelector('#coverFileUploadSuccess');
var secretUploadForm = document.querySelector('#secretUploadForm');
var secretFileUploadInput = document.querySelector('#secretFileUploadInput');
var secretFileUploadError = document.querySelector('#secretFileUploadError');
var secretFileUploadSuccess =
    document.querySelector('#secretFileUploadSuccess');

var encryptedError = document.querySelector('#encryptedFileDownloadError');

```

```

var encryptedSuccess =
    document.querySelector('#encryptedFileDownloadSuccess');

function uploadCoverFile(file) {

    var formData = new FormData();

    formData.append("file", file);

    var xhr = new XMLHttpRequest();

    xhr.open("POST", "/uploadCoverFile");

    xhr.onload = function() {

        console.log(xhr.responseText);

        var response = JSON.parse(xhr.responseText);

        if(xhr.status == 200) {

            coverFileUploadError.style.display = "none";

            coverFileUploadSuccess.innerHTML = "<p>File Uploaded  
Successfully.</p><p>DownloadUrl : <a href='" + response.fileDownloadUri + '"  
target='_blank'>" + response.fileDownloadUri + "</a></p>";

            coverFileUploadSuccess.style.display = "block";

        } else {

            coverFileUploadSuccess.style.display = "none";

            coverFileUploadError.innerHTML = (response && response.message) ||  
"Some Error Occurred";

        }

    }

    xhr.send(formData);

}

function uploadSecretFile(file) {

    var formData = new FormData();

```

```

formData.append("file", file);

var xhr = new XMLHttpRequest();

xhr.open("POST", "/uploadSecretFile");

xhr.onload = function() {

    console.log(xhr.responseText);

    var response = JSON.parse(xhr.responseText);

    if(xhr.status == 200) {

        secretFileUploadError.style.display = "none";

        secretFileUploadSuccess.innerHTML = "<p>File Uploaded  
Successfully.</p><p>DownloadUrl : <a href='" + response.fileDownloadUri + '"  
target='_blank'>" + response.fileDownloadUri + "</a></p>";

        secretFileUploadSuccess.style.display = "block";

    } else {

        secretFileUploadSuccess.style.display = "none";

        secretFileUploadError.innerHTML = (response && response.message) ||  
"Some Error Occurred";

    }

}

xhr.send(formData);//contentType(MediaType.parseMediaType(contentType))

}

coverUploadForm.addEventListener('submit', function(event){

    var files = coverFileUploadInput.files;

    if(files.length === 0) {

        coverFileUploadError.innerHTML = "Please select a file";

        coverFileUploadError.style.display = "block";

    }

```

```

        uploadCoverFile(files[0]);

        event.preventDefault();

    }, true);

    secretUploadForm.addEventListener('submit', function(event){

        var files = secretFileUploadInput.files;

        if(files.length === 0) {

            secretFileUploadError.innerHTML = "Please select a file";

            secretFileUploadError.style.display = "block";

        }

        uploadSecretFile(files[0]);

        event.preventDefault();

    }, true);

    function generateImage() {

        var xhr = new XMLHttpRequest();

        xhr.open("GET", "/getEncryptedImage");

        xhr.onload = function() {

            console.log(xhr.responseText);

            var response = JSON.parse(xhr.responseText);

            if(xhr.status == 200) {

                encryptedError.style.display = "none";

                encryptedSuccess.innerHTML = "<<p>DownloadUrl : <a href='" +
                response.fileDownloadUri + "' target='_blank'>" + response.fileDownloadUri +
                "</a></p>";

                encryptedSuccess.style.display = "block";

            } else {

                encryptedSuccess.style.display = "none";

            }

        }

    }

```

```
        encryptedError.innerHTML = (response && response.message) || "Some  
Error Occurred";  
    }  
}  
xhr.send();  
}
```

B. RESEARCH PAPER

Stealth and Security : Exploring Multimedia Steganography

Bharti Singh^{1,a)}, Rishav Agrawal^{2,b)}, A Sivasangari^{3,c)}, D Ramalakshmi^{4,d)}
Student¹, Student², Professor³, Assistant Professor⁴

^{1, 2, 3, 4} Department of Computer Science and Engineering, Sathyabama Institute of Science and Technology

^{a)}852bhartisingh12345@gmail.com

^{b)}aggrawalrishav@gmail.com

^{c)}sivasangarikavya@gmail.com

^{d)}laksjetish@gmail.com

Abstract. According to the current estimates, 327.77 million tera bytes of data are produced each day. Protecting and securing the transmission of this data is the key to the creation of a better digital world. Data is the contamination trouble of the knowledge age, and defending confidentiality is the substantial challenge. In the present decade, images have become a valuable source of information. They play an essential role in every field whether it be medical, agriculture, defense or education. Over the past few years, steganography has received a lot of attention for passing military sensitive details. The information is passed generally via image steganography. With the increase in amount to data to hide, other multimedia formats are gaining popularity too. Secured ideas on impressionable information amid two bodies need to consider news freedom as necessity. Steganography and signaling code maybe used to enhance the facts safety. Cryptography discloses the availability of news, when in fact steganography conceals the availability of facts because no other than two together writing parties can acknowledge it. With progress in internet requests the news mainly resides of passage, visual and audio entertainment transmitted via radio waves, image, and television type of news. This project provides a particularized study and methodology for Multimedia located steganography of facts. Also, we tend to consider the feasible methods and methods for combined use of several media steganography and their accomplishments.

INTRODUCTION

Steganography is the practice of disguising individual piece of information inside another as if it is troublesome to discover or decipher. It is often used to hide the existence of a secret message within an innocuous carrier medium, such as text, images, audio, or video files. The word “steganography” is composed up of the Greek words “steganos” meaning “covered” and “graphia” meaning “writing” and essentially describes the process of hiding information in another data form.

Multimedia steganography, specifically, focuses on concealing information within multimedia files like images, audio, and video. This branch of steganography has gained significant attention due to the proliferation of digital multimedia content on the internet. Here’s a brief introduction to steganography and multimedia steganography. Multimedia steganography focuses on hiding data within various forms of multimedia content. In image steganography, secret data is introduced within an image file. Common techniques include modifying the least significant bits of the pixel values or altering the frequency components of the image in a way that is unseen to the human eye.

In audio steganography, secret data is hidden within an audio file, often inaudibly. Techniques may involve modifying the amplitude or frequency.

Video steganography hides data within video files. It can be more challenging than other forms, as it involves both spatial and temporal data. Techniques often rely on modifying frames or altering specific video characteristics.

Steganography, especially multimedia steganography, plays a crucial role in information security, privacy, and copyright protection. It allows for the covert transmission of data while maintaining the appearance of the carrier medium, making it an essential tool in various fields, including cybersecurity and digital rights management.

Network steganography is the youngest branch of information hiding. It is a fast-developing field. Recent years have resulted in multiple new information hiding methods, which can be exploited in various types of networks. The exploitation of protocols belonging to the Open Systems Interconnection (OSI) reference model [59] is the essence of network steganography. This family of methods may utilize one or 8 more protocols

simultaneously or the relationships between them – relying on the modification of their intrinsic properties for the embedding of steganograms. Lately, another steganographic method – TranSteg (Transcoding Steganography), intended for a broad class of multimedia and real-time applications, like IP telephony, has been proposed. TranSteg bases on the general idea of transcoding (lossy compression) of the voice data from a higher bit rate codec, and thus greater voice payload size, to a lower bit rate codec with smaller voice payload size. This occurs with the least possible degradation in voice quality. In other words, compression of the overt data is utilized to make space for the steganogram in the payload field. The achieved steganographic bandwidth is as high as 32 kbit/s.

LITERATURE REVIEW

The paper[1] describes the design and development of a steganography method that involves the method of use of chaotic map. The paper begins by providing an overview of steganography and the types of steganography available. This paper focuses on image- steganography. The paper discards the use of traditional methods such as LSB manipulation or Cosine Transformation to embed the plain text into an image. The author uses LSB Chaotic Maps to encrypt the plain text. The author has proposed various chaotic mapping methods available to secure the information. The author discusses about six available methods that is used to find the pixels within the image that is to be used to encrypt the bits of plain text. This paper's work is limited to the encryption of plain text within the image. The model shows great future aspects as it limits the work that this proposed model can handle.

This author[2] addresses the use of Random Bit Shifting with the symmetric cryptographic technique, AES, for steganography imaging. In this procedure, the plain text is first encrypted using the AES algorithm, and the message is then encrypted using a secret key that is 128 bits long. The random bit will select a pixel location in the upper image and replace it with secret bits using the secret key. The author advises using cover pictures, like welcoming messages, that are less vulnerable to attacks. The many image formats and the quantity of information that can be contained in them are also covered in the study. Once more, this paper should only be used in cases where an image needs to have a text encrypted in it. No other format of multimedia can be processed with this algorithm. Though the authors have encrypted the text before embedding, it still has a lot of future aspects.

The paper[3] describes the development of a steganography method for audios. In this work, audio is being used as the cover to encrypt the text. This kind of work promises good scope for watermarking and protecting digital rights and providing shields from copyright issues. The paper discusses about the then available methods to encrypt. This paper uses LSB Chaotic map that are non- periodic, non-linear and random to embed the bits of text. The paper shows the graph amplitude vs time which shows the layer file and the secret file. This is one of the best approaches to embed the information in the audio file as any change in the bits of the cover file may lead to the change in frequency of the cover file resulting in noise and distortion. This paper has a promising future as we move on to new aspects of this paper.

The paper[4] proposes a model for a price reassignment procedure for reconciling steganography based on artificial immunity system. The importance of the cost function in securing adaptive image steganography cannot be overstated. Nevertheless, certain current cost functions are intricately designed heuristics, making it challenging to achieve optimality in undetectability against advancing Steg analyzer techniques. In the context of minimizing additive distortion, this approach recalibrates the cost through adjustments in the modified probability spread derived from the price function. The dynamic optimization of the adjustment mode is achieved through an immunity-based information encrypting model, ensuring that the altered points are concentrated in regions that pose increased difficulty for detection. This paper proposes CR-AIS method to reassign cost.

The author[5] researches deep learning-based generative image steganography schemes, comprehends related theories of deep learning and image steganography from literature, and then develops a deep learning-based generative image steganography scheme. The designed scheme is tested to determine whether it outperforms its successor method in terms of accuracy. Following a review of pertinent ideas, a deep learning-based generative image steganography method was designed. The results of the experiments indicate that the accuracy of the resulting stego image is up to 4% higher than that of its predecessors.

The author[6] describes the successful usage of conclusion of file steganography method. The author carries out the work with variety of message sizes. The author recommends the usage of video for the steganography purpose as it is larger in size and contains audio and images in a single file as well. The files are also susceptible to changes in their size, so it is very important to choose the file wisely. The methodology has been tested on various file formats such as JPEG, MP3 and MP4. The files are tested thoroughly for the chance of errors and to study the difference between the upper file and the secret file.

The authors[7] presents the different types of cryptography algorithms that are used to encrypt the image and the plaintext. The paper discusses about the use of chaotic maps and the impact it gives in the output file. The paper discusses the violent nature of chaotic map suggesting that even a small change in the parameter may produce huge changes. The chaotic-based encryption method has many pros, such as sensibility to pre conditions, haphazardness and necessity. Again, the research is only applicable for encrypting the text in a multimedia though

it uses better approach and covers more multimedia formats.

The authors[8] explain how wavelet transformation is used as the foundational algorithm and technique for image steganography. This article's work was completed using greyscale images and MATLAB. Both RGB and greyscale photos can be used to execute the data that needs to be hidden. The cover image will determine the storage capacity. The peak signal to noise ratio (PSNR) and error variance (MSE) metrics are used to assess the quality of both encrypted and decrypted images. This paper discusses the LSB approach, which uses wavelet modification to embed a picture into another image file. The report also discusses the loss of certain information when discussing hazy pictures. You can use this method further to increase the compression ratio.

The Quick Response (QR) encryption approach[9] is used by the authors to create and implement message security for data represented in images. The QR code is written in the inessential Bit (LSB) in this publication. The authors used MATLAB to carry out simulation. Since the author discusses encrypting the message using the AES ciphering technique before encoding it within a QR Code, this post has a lot of potential applications in the future. There's a lot of work to be done with various cover image sizes in addition to various QR Code sizes.

The system that can be used to cipher big documents is designed and implemented, and this is described in the paper[10]. MPDFStego is a proposed program that increases a PDF document's steganography capabilities by performing steganography on several PDF documents. Here, PDFs employ the Least Significant Bit (LSB) by utilizing the Tj Operator. Before being inserted into the PDFs, the secret text is encrypted using Advanced Ciphering techniques (AES) in Enciphering Block Chaining (CBC) mode with a 128-bit key size. The QPDF utility is used to do the decompression and recompression. Additional research could be done on applying these findings to hardware. Recent work on hardware simulation virtual prototyping platforms enables early software development and system integration. Performance improvements could result from the incorporation of steganography and encryption into hardware.

The power of steganography and cryptography working together to ensure communication confidentiality is presented in this study [11]. The use of flips and rotation to thwart deciphering encoded information is covered in the study. Interactive file compression is designed to accelerate encipherment and use less memory space. For message integrity and authentication, the Hash-based substantiate Code (HMAC) algorithm is used. Through the use of histogram comparing analysis, the effectiveness of the suggested steganography methods is concluded. According to the inspection, the enciphered images generated by the current process and the suggested encircling LSB Steganography method resembled the cover image more than the suggested Flipping LSB Steganography method, and the size of the secret message very lightly decreased as it increased.

The authors[12] explain how wavelet transformation is used as the foundational algorithm and technique for image steganography. This article's work was completed using greyscale images and MATLAB. Both RGB and greyscale photos can be used to execute the data that needs to be hidden. The cover image will determine the storage capacity. The peak signal to noise ratio (PSNR) and error variance (MSE) metrics are used to assess the quality of both encrypted and decrypted images. This paper discusses the LSB approach, which uses wavelet modification to embed a picture into another image file. The report also discusses the loss of certain information when discussing hazy pictures. You can use this method further to increase the compression ratio.

The authors[13] presents the different types of cryptography algorithms that are used to encrypt the image and the plaintext. The paper discusses about the use of chaotic maps and the impact it gives in the output file. The paper discusses the violent nature of chaotic map suggesting that even a small change in the parameter may produce huge changes. The chaotic-based encryption method has many pros, such as sensibility to pre conditions, haphazardness and necessity. Again, the research is only applicable for encrypting the text in a multimedia though it uses better approach and covers more multimedia formats.

The author[14] researches deep learning-based generative image steganography schemes, comprehends related theories of deep learning and image steganography from literature, and then develops a deep learning-based generative image steganography scheme. The designed scheme is tested to determine whether it outperforms its successor method in terms of accuracy. Following a review of pertinent ideas, a deep learning-based generative image steganography method was designed. The results of the experiments indicate that the accuracy of the resulting stego image is up to 4% higher than that of its predecessors.

The author[15] describes the successful usage of conclusion of file steganography method. The author carries out the work with variety of message sizes. The author recommends the usage of video for the steganography purpose as it is larger in size and contains audio and images in a single file as well. The files are also susceptible to changes in their size, so it is very important to choose the file wisely. The methodology has been tested on various file formats such as JPEG, MP3 and MP4. The files are tested thoroughly for the chance of errors and to study the difference between the upper file and the secret file.

EXISTING SYSTEM

The Unimportant Bit (LSB) method is a well-known technique that involves inserting secret information bits at the ending bit of each byte that belongs to the media cover. Because this LSB approach only employs 1 bit for

steganography from each 1 byte of carrier cover, it has a limitation on the amount of secret information that may be entered. Another drawback[12] is that steganalysis practitioners are familiar with this method's insertion procedure due to its classic nature. The last sections of the media cover file are a simple place for steganalysis to extract hidden messages.

PROPOSED SYSTEM

Chaotic maps, specifically Logistic and Tent maps, are often used in chaotic cryptography for encryption and decryption purposes. The concept involves utilizing the chaotic behavior of these ways to produce cryptographic keys that are pseudo-random sequences. These sequences can be used for various purposes, such as key generation, substitution, and permutation in encryption and decryption algorithms. Here's a basic overview of how chaotic maps like Logistic and Tent maps can be used in encryption and decryption. The logistic map is system of discrete temporal dynamics that behaves erratically under certain conditions. It is explained by the recurrence relation. $X_{n+1} = \mu X_n(1 - X_n)$, where ' X_n ' is the current value, and ' μ ' is a parameter that determines the behavior of map. To use the logistic map for encryption, you can initialize it with a secret key, iterate it to generate a chaotic sequence, and use this sequence as a key for XORing with the plaintext. Another chaotic map that's employed in cryptography is the tent map. The piecewise-linear function defines it. $X_{n+1} = a - (1 + a) |X_n|$, where ' n ' is the current value, and ' a ' is a parameter. Similar to the logistic map, you can use the tent map to generate a chaotic sequence for encryption purposes. One characteristic of chaotic systems is their sensitivity to initial conditions, a quality known as the "butterfly effect," where minor differences in starting states lead to significantly divergent outcomes. This sensitivity is quantified by positive Lyapunov exponents, indicating the rapid divergence of trajectories and contributing to the orbital instability of chaotic systems. The ergodicity property ensures that these systems explore their entire phase space over time, facilitating statistical predictions. Exploiting these chaotic features, particularly sensitivity and ergodicity, allows for applications such as generating pseudo-random sequences and encrypting information, leveraging the unpredictability inherent in chaotic dynamics. This sensitivity is quantified by positive Lyapunov exponents, indicating the rapid divergence of trajectories and contributing to the orbital instability of chaotic systems. The ergodicity property ensures that these systems explore their entire phase space over time, facilitating statistical predictions. Exploiting these chaotic features, particularly sensitivity and ergodicity, allows for applications such as generating pseudo-random sequences and encrypting information, leveraging the unpredictability inherent in chaotic dynamics. To use the logistic map for encryption, you can initialize it with a secret key, iterate it to generate a chaotic sequence, and use this sequence as a key for XORing with the plaintext. The logistic map is system of discrete temporal dynamics that behaves erratically under certain conditions. It is explained by the recurrence relation. Among the popular chaotic mapping techniques that are now in use are: Logistics Map formula (1), Tent Map formula (2), Quadratic Map formula (3), Bernoulli Map formula (4), Sine Map formula (5), Chebyshev Maps formula (6)

$$X_{n+1} = \mu X_n(1 - X_n) \quad (1)$$

$$T_r(x) = \begin{cases} 2rx, & 0 \leq x \leq \frac{1}{2} \\ 2r(1-x), & \frac{1}{2} < x \leq 1 \end{cases} \quad (2)$$

$$X_{n+1} = r - X_n^2 \quad (3)$$

$$X_{i+1} = \begin{cases} 2\mu x_i + \frac{(1-\mu)}{2}, & 0 \leq x < 0.5 \\ (2\mu x_i - 1) + \frac{(1-\mu)}{2}, & 0.5 \leq x < 1 \end{cases} \quad (4)$$

$$X_{n+1} = \mu \sin(\pi X_n) \quad (5)$$

$$X_{n+1} = \cos(k * \cos^{-1}(X_n)) \quad (6)$$

Given the chaotic map equations, it is evident that they are all impacted by the earlier states. As a result, those maps are sensitive to the initial conditions, and it is challenging to predict the map's future result until the initial conditions are understood. The assessment of steganographic systems by the study's authors involves the consideration of vital elements like load capacity, invisibility, and defense against stego attacks and analysis. These criteria function as benchmarks for gauging the effectiveness of steganographic methods. Additionally, the research underscores the significance of statistical indeterminacy as a fundamental requirement within their evaluation framework. To appraise the specified metrics and gauge the stego image's quality, computations are conducted for Correlation analysis, entropy, mean squared error, and peak signal-to-noise ratio, or PSNR,. Equations from (7) to (10) represent the provided metrics for evaluation.

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [C(i, j) - S(i, j)]^2 \quad (7)$$

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX^2}{MSE} \right) \quad (8)$$

$$CorrCoef(C, S) = \frac{\sum_{i=1}^n (C_i - \bar{C})(S_i - \bar{S})}{\sqrt{\sum_{i=1}^n (C_i - \bar{C})^2 \sum_{i=1}^n (S_i - \bar{S})^2}} \quad (9)$$

$$Entropy = H(S) = - \sum_{i=1}^n p_{s_i} \log_2 p_{s_i} \quad (10)$$

Encryption Process: Initialize the chaotic map with a secret key or initial conditions according to *fig. 1*. Iterate the map to generate a chaotic sequence. Use the generated sequence as a key for XORing with the plaintext for encryption.

Decryption Process: Initialize the chaotic map with the same secret key or initial conditions. Iterate the map to regenerate the chaotic sequence. XOR the encrypted ciphertext with the regenerated chaotic sequence to obtain the original plaintext/media.

The assessment of steganographic systems by the study's authors involves the consideration of vital elements like load capacity, invisibility, and defense against stego attacks and analysis. These criteria function as benchmarks for gauging the effectiveness of steganographic methods. Additionally, the research underscores the significance of statistical indeterminacy as a fundamental requirement within their evaluation framework. This sensitivity is quantified by positive Lyapunov exponents, indicating the rapid divergence of trajectories and contributing to the orbital instability of chaotic systems. The ergodicity property ensures that these systems explore their entire phase space over time, facilitating statistical predictions. Exploiting these chaotic features, particularly sensitivity and ergodicity, allows for applications such as generating pseudo-random sequences and encrypting information, leveraging the unpredictability inherent in chaotic dynamics. The Unimportant Bit (LSB) method is a well-known technique that involves inserting secret information bits at the ending bit of each byte that belongs to the media cover. Because this LSB approach only employs 1 bit for steganography from each 1 byte of carrier cover, it has a limitation on the amount of secret information that may be entered. Another drawback[12] is that steganalysis practitioners are familiar with this method's insertion procedure due to its classic nature. The last sections of the media cover file are a simple place for steganalysis to extract hidden messages.

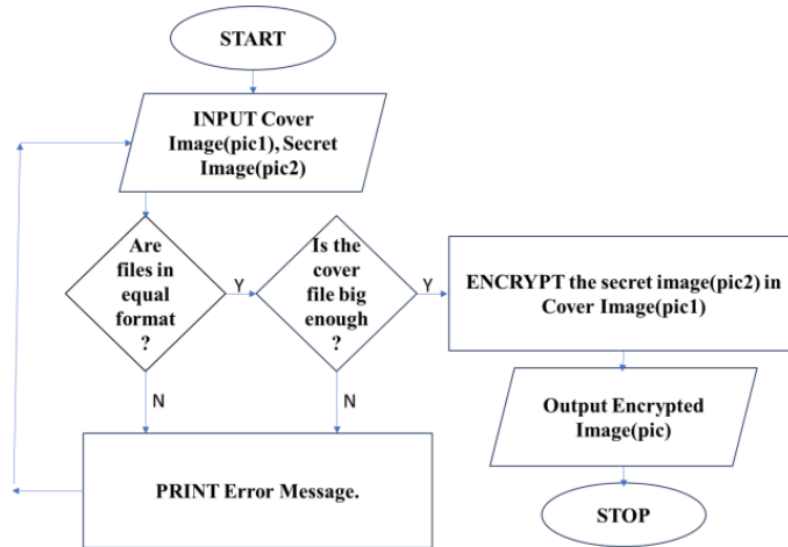


Fig 1. Flowchart showing the process of encryption within the application. The app takes 2 images as input and performs certain checks before proceeding for the encryption of one within the another. If it meets the condition, the pointer moves ahead and completes the task as per the user's requirement.

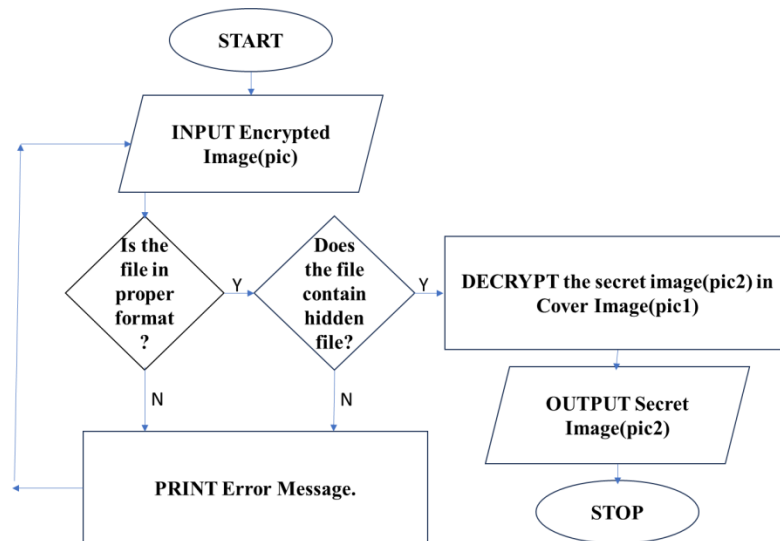


Fig 2. Flowchart showing the process of decryption within the application. The app takes encrypted image as input and performs certain checks before proceeding for the decryption of image. If it meets the condition, the pointer moves ahead and completes the task as per the user's requirement.

SYSTEM ARCHITECTURE

The architecture of this project work is made with keeping in mind the use cases of the today's requirements. In the architecture, a video which is made up of 'n' no. of frames is used to cover the multimedia. Video is used as it contains multiple frames and can handle large no. of extra multimedia files. The encryption and decryption is done with the help of algorithm and procedure mentioned in the previous section. The frame selection and the pixel selection is done with proper algorithm. The algorithm is used in such a manner that it is almost impossible for stego-analysts to decrypt it. A stego tool is used to further analyze the results obtained. The images have been used in proper order and format so as to reduce the noise. Care has been taken to properly analyze and produce little to no noise in the running images. While decrypting, the algorithm is reverse engineered to rearrange the order of images in which it was encrypted. A popular stego tool by Forczmanski et. al known as Laboratory of Virtual Steganography(VSL) for Digital Images[13] is used for steganalysis. A comprehensive and comparative study has been done with the original image and the enciphered image for the changes. The *table 1* depicts the usage of various available chaotic mapping algorithms and their different evaluation parameters. It also encompasses various error calculation methods and changes in the entropies according to the formulas and equations mentioned. They also introduce several methods for LSB encoding using Chaotic Maps[14].

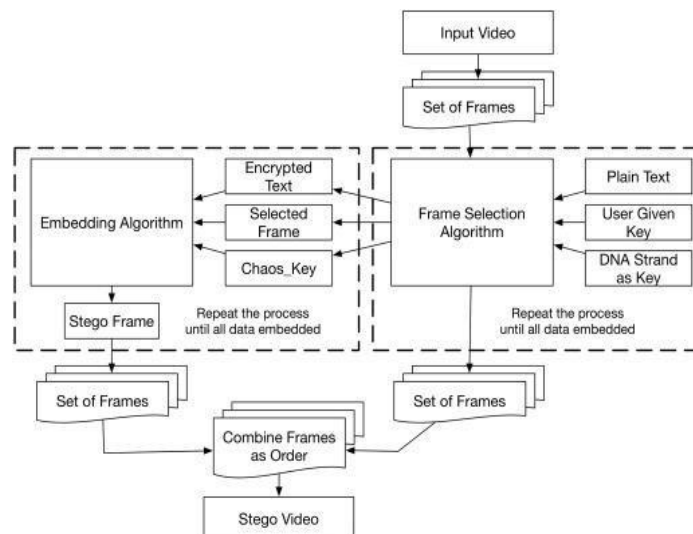


Fig 3. System architecture is the architecture of the process flow and the working of the application and algorithm. The architecture covers all the important aspects of the application in a systematic and graphical representation. The architecture explains the working of application.

RESULTS AND DISCUSSIONS



Fig 4. (a) A picture of an animal that will act as a cover image for the steganography application, (b) A picture of a human that needs to be hidden in the cover image, (c) The encrypted image of an animal, this image has embedded human's image within itself. It is also called as stego-image.

The introduction of an image as shown in *fig. 4* in another image as shown in *fig. 3* produces noise as seen in *fig. 5* which makes it obvious for the steganalysts to decipher the enciphered media. The use of LSB Chaotic Mapping algorithm makes it harder for the programmers to find the pixels which have been modified to embed the secret image. Even if they are successful in doing so, it would be harder for them to join the pictures into something meaningful. This algorithm does not use regular pixel values which makes it unique. These days, data and information are highly valued, and transmitting and receiving data securely is crucial. The methods utilized for secure transmission and private information protection are cryptography and steganography. Current methods combine LSB encoding steganography with cryptographic algorithms for encryption. They[15] also introduce several methods for LSB encoding using Chaotic Maps. In many eHealth applications, such as data management, identity theft prevention, storage, and retrieval, image security is crucial. This paper includes a brief explanation of a few well-known picture assaults that are used to test the performance of the suggested algorithms on images. In order to encourage more study in this field, we have covered the cryptographic technique and its traits, types, prerequisites, working domain, and probable problems. Moreover, various performance measures for assessing algorithms have been shown in *table 1*. The paper presents a methodology for video steganography that uses text, audio, and image files. A carrier video is retrieved, and the frames that the audio, text, and image files can be inserted within are used to reconstruct the video. According to the mathematical model, the suggested architecture should perform better than the current approaches.

TABLE 1. The table shows the comparative study of various algorithms and the noise they produced when tested with the special tool developed specially to calculate the noise in the application. The available methods after modification in the algorithm produced the following results.

IMAGE	<i>Bernoulli</i>	<i>Logistic</i>	<i>Quadratic</i>	<i>Sine</i>	<i>Tent</i>
Original Image Entropy	0.15658549505	0.15658549505	0.02025277832	0.15653749505	0.156549505
Carrier Image Entropy	0.15658588824	0.15658577751	0.15658630971	0.15658400236	0.156614799
Entropy Difference	-0.000000393196	-0.000005407587	-0.000008814662	0.000000492685	-0.0001652948
Correlation Difference	0.999998637121	0.999994765612	0.999995685202	0.999994634734	0.9999669674
MSE	0.020141156250	0.020353930664	0.020252778320	0.020379809570	0.0201386523
PNR	68.08986328575	68.04323001598	68.06607747841	68.03997485043	68.0906818610

This paper includes a brief explanation of a few well-known picture assaults that are used to test the performance of the suggested algorithms on images. In order to encourage more study in this field, we have covered the cryptographic technique and its traits, types, prerequisites, working domain, and probable problems.

Moreover, various performance measures for assessing algorithms have been shown in *table 1*. The paper presents a methodology for video steganography that uses text, audio, and image files. A carrier video is retrieved, and the frames that the audio, text, and image files can be inserted within are used to reconstruct the video. According to the mathematical model, the suggested architecture should perform better than the current approaches.

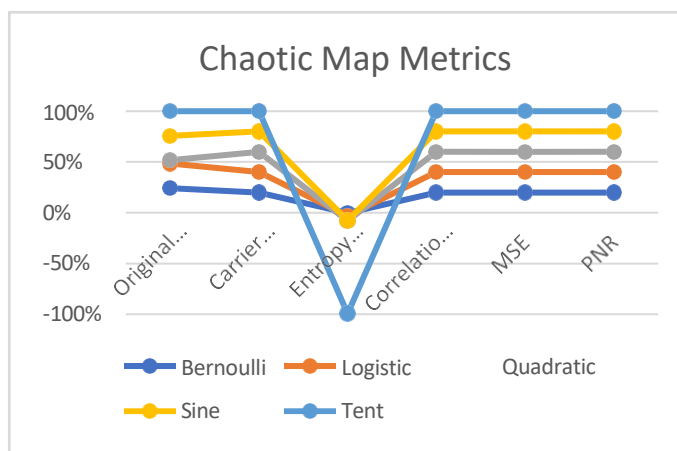


FIGURE 5. The above chart shown in shows the comparative study between the different types of chaotic mapping methods. It is clearly evident from the line chart that the Entropy in the Tent Map is the most and rest of the parameters shows the flat line

CONCLUSION

Future work might focus on optimizing the parameters of chaotic maps to enhance their cryptographic properties. Finding parameter sets that provide better chaos and improved security is an ongoing research area. Exploring dynamic key management strategies could be an avenue for future research. Adapting the chaotic map parameters or initial conditions during the encryption process could contribute to increased security. Integrating chaotic maps into hybrid cryptosystems that combine chaotic cryptography with other established cryptographic techniques. This could provide additional layers of security and improve overall robustness. Exploring the integration of chaotic maps into quantum cryptography. Quantum chaotic maps may provide unique opportunities for secure communication in quantum systems. Investigating the feasibility and efficiency of hardware implementations for LSB chaotic maps. This could lead to the development of dedicated hardware for faster and more energy-efficient chaotic encryption. Considering the post- quantum security aspects of chaotic maps. As quantum computing advances, researchers may explore the resilience of chaotic cryptography against quantum attacks and develop post-quantum versions of these algorithms.

In conclusion, the research on multimedia steganography using the LSB chaotic mapping method introduces a robust and secure approach for concealing messages within images. The algorithm utilizes chaotic maps to enhance security, incorporates Gzip compression for efficient data handling, and employs LSB embedding in color channels to minimize visual impact. Extensive testing validates the method's effectiveness, suggesting it as a promising technique for secure information concealment in multimedia applications. Further research could optimize parameters and explore alternative chaotic mapping methods, ensuring the continued evolution of reliable and resilient steganographic techniques.

In conclusion, the LSB chaotic mapping method for multimedia steganography stands as a promising and viable technique for secure information concealment. As digital communication continues to evolve, the presented approach contributes to the ongoing efforts in developing robust and reliable methods for covert data transmission in multimedia environments.

REFERENCES

1. Artz, D. 2021. Digital steganography: Hiding data within data. *IEEE Internet Computing Journal*, 5(3):75-80.
2. J. Fridrich, M. Goljan and Rui Du, Detecting LSB steganography in color, and gray-scale images, *IEEE Multimedia*, vol. 8, no. 4, pp. 22- 28, Oct.-Dec. 2021, doi: 10.1109/93.959097
3. C. Kurak, J. McHugh, A cautionary note on image downgrading, in: *Proceedings of the IEEE 8th Annual Computer Security Applications Conference*, 30 November–4December, 2022, pp. 153–159.
4. W. Bender, W. Butera, D. Gruhl, R. Hwang, F.J. Paiz, S. Pogreb, Applications for data hiding, *IBM Systems Journal* 39 (3&4) (2022)547–568.
5. Y. Chen, H. Wang, H. Wu, Z. Wu, T. Li, and A. Malik, “Adaptive video data hiding through cost assignment and STCs,” *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1320–1335, Jun. 2021.
6. V. Holub and J. Fridrich, “Designing steganographic distortion using directional filters,” in *Proc. IEEE 4th Int. Workshop Inf. Forensics Secur.*, 2020, pp. 234–239.
7. Ren T, Kotera M, Ogawa T, et al. Text-Guided Style Transfer-based Image Manipulation Using Multimodal Generative Models. *IEEE Access*, 2021, PP(99):1-1.
8. Taburet T, Bas P, Sawaya W, et al. Natural Steganography in JPEG Domain with a Linear Development Pipeline. *IEEE Transactions on Information Forensics and Security*,2020, PP(99):1-1.
9. Petitcolas FAP, et al. “Information hiding-a survey”. *Proceedings of the IEEE*, 87(7), 1062- 1078, 2020.
10. S. Xu, et al. “No-reference/Blind Image Quality Assessment: A Survey,” *IETE Tech. Rev. (Institution Electron. Telecommun. Eng. India)*, vol. 34, pp. 223–245, 2017.
11. S.Thenmozhi,M.Chandrasekaran, “Novel approach for image steganography based on integer wavelet transform,” 2020 *IEEE International Conference on Computer Intelligence and Computing Research* ,2012,pp.1-5,doi: 10.1109/ICCIC.2020.651030
12. C. Kurak, J. McHugh, A cautionary note on image downgrading, in: *Proceedings of the IEEE 8th Annual Computer Security Applications Conference*, 30 November–4 December, 2022, pp. 153–159.
13. W. Bender, W. Butera, D. Gruhl, R. Hwang, F.J. Paiz, S. Pogreb, Applications for data hiding, *IBM Systems Journal* 39 (3&4) (2022)547–568.
14. Y. Chen, H. Wang, H. Wu, Z. Wu, T. Li, and A. Malik, “Adaptive video data hiding through cost assignment and STCs,” *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1320–1335, Jun. 2021.
15. V. Holub and J. Fridrich, “Designing steganographic distortion using directional filters,” in *Proc. IEEE 4th Int. Workshop Inf. Forensics Secur.*, 2020, pp. 234–239.

C. CERTIFICATE

