



# **SECURE TRANSMISSION OF MEDICAL IMAGES USING STEGANOGRAPHY AND ENCRYPTION**

**A PROJECT REPORT**

*Submitted by*

**CRYSTAL DARLING B  
INFANT BENITA L  
PONMANI P R  
SHEKINAH T**

*in partial fulfilment for the award of the degree of*

**BACHELOR OF ENGINEERING**

*in*

**COMPUTER SCIENCE AND ENGINEERING**

**LOYOLA-ICAM COLLEGE OF ENGINEERING AND  
TECHNOLOGY**

**CHENNAI – 600034**

**ANNA UNIVERSITY: 600025**

**APRIL 2022**  
**ANNA UNIVERSITY CHENNAI: 600025**

**BONAFIDE CERTIFICATE**

Certified that this project report “**Secure Transmission of Medical Images Using Steganography and Encryption**” is the bonafide work of **CRYSTAL DARLING B (311119104016)**, **INFANT BENITA L (311119104028)**, **PONMANI P R (311119104049)** and **SHEKINAH T (311119104056)** who carried out the project work under my supervision.

**SIGNATURE**

Dr. K. Gopalakrishnan

**HEAD OF THE DEPARTMENT**

Department of Computer Science  
and Engineering  
Loyola-ICAM College of  
Engineering and Technology  
Loyola Campus,  
Nungambakkam,  
Chennai-600034

**SIGNATURE**

Ms. Jeevitha

**SUPERVISOR**

Department of Computer Science  
and Engineering  
Loyola-ICAM College of  
Engineering and Technology  
Loyola Campus,  
Nungambakkam,  
Chennai-600034

Submitted for the project viva voce held on .....

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

Firstly, we are deeply obliged to the almighty for this opportunity and for leading us to the successful completion of the project. The experience has been very fruitful.

We express our sincere gratitude to the beloved Director **Rev Dr S Maria Wenisch SJ**, the Dean of Students **Dr D Caleb Chanthi Raj** and the Dean of Engineering Education **Mr Nicolas Juhel**, for their continuous support and Encouragement.

We are deeply obliged to the Principal **Dr L Antony Michael Raj** for his inspiring guidance and invaluable advice during the project work. We also express our sincere thanks to the Head of the Department and Project Coordinator **Dr K Gopalakrishnan**, the teaching and non-teaching faculty of our department for their advice, support and guidance.

We extend our gratitude to our project guide **Ms Jeevitha** for providing valuable insights and resources. Without her supervision and constant help, this project would not have been possible.

Last but not the least, we are extremely grateful to our family and friends, who have been a constant source of support during the preparation of this project work.

## ABSTRACT

Patients' data are stored in an electronic record known as Electronic Health Record (EHR). This EHR should be kept confidential because it contains sensitive patient information. Recent development in communication and information technology provides easy and simple access to any data, but the most significant requirement is the establishment of secure communication. Patient's information is available in various online repositories due to the insecure network of hospitals and clinics. This is a serious ethical issue that has to be addressed.

Also, these records are transmitted over multiple networks whether local or wide area networks. These transmissions are governed by Digital Imaging and Communications in Medicine (DICOM) standard, to avoid any altering of the data, capturing and copyright protection. The DICOM file that contains the patients' information needs to be kept confidential.

Several techniques are developed for safe communication. Some of them include cryptography and steganography. By leveraging the advantages of both techniques, it is possible to safely transmit medical information and images across hospital networks.

The major contribution of this project is combining several steganography and encryption techniques, for secure transmission of data without leakage and unauthorized access. This is achieved as follows:

- Firstly, the patient's data which is present in the DICOM header is encrypted using DNA based Advanced Encryption Standard (AES) algorithm.
- Secondly, Least Significant Bit (LSB) embedding is performed to hide the patient's information inside the image that is used as a cover object.
- Thirdly, to keep the stego image size reasonable Discrete Wavelet Transform compression is performed and transmitted to the receiver.
- Finally, the receiver performs the reverse of the proposed process to extract the secret data.

Experiments are conducted to evaluate the performance of proposed combined steganography techniques using various statistical metrics and results indicate that the proposed technique outperforms all existing techniques.

**Keywords:** *Steganography; secure data transmission; DNA; AES; cryptography; telemedicine; EHR; compression; DICOM*

## TABLE OF CONTENT

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	iv
	<b>LIST OF FIGURES</b>	vii
<b>1</b>	<b>INTRODUCTION</b>	
	1.1 Problem definition	1
	1.2 Existing applications	1
	1.3 Need for the system	4
<b>2</b>	<b>LITERATURE SURVEY</b>	
	2.1 Securing Medical Images using Encryption and LSB Steganography	5
	2.2 Steganography-Based Transmission of Medical Images Over Unsecure Network for Telemedicine Applications	5
	2.3 A Safe and Secured Medical Textual Information Using an Improved LSB Image Steganography	5
	2.4 An effective and secure digital image steganography scheme using two random function and chaotic map	5
	2.5 A Methodology Based on Steganography and Cryptography to Protect Highly Secure Messages	6
	2.6 A modification of least significant digit (LSD) digital watermark technique	6
	2.7 Performance analysis of LSB-based data hiding techniques	6
	2.8 Image steganography for authenticity of visual contents in social networks	6

	2.9 Hiding shares by multimedia image steganography for optimized counting-based secret sharing	6
	2.10 Encryption and Decryption of Color Images using Visual Cryptography	7
<b>3</b>	<b>SYSTEM ANALYSIS</b>	
	3.1 System requirements	8
	3.1.1 Requirements	8
	3.1.2 Packages	9
<b>4</b>	<b>SYSTEM DESIGN</b>	
	4.1 Object oriented design	11
	4.2 Structural diagram	11
	4.3 Behavioral diagram	12
	4.4 Use case diagram	12
	4.5 Architecture diagram	13
<b>5</b>	<b>SYSTEM IMPLEMENTATION</b>	
	5.1 Algorithm and proposed technique	14
	5.1.1 DNA based Encryption algorithm	14
	5.1.2 LSB Embedding for Steganography	14
	5.1.3 Discrete Wavelet Transformation	14
	5.1.4 Extraction	15
	5.1.5 Decryption	15
<b>6</b>	<b>PERFORMANCE METRICS</b>	
	6.1 Evaluation metrics	16
	6.1.1 Mean Squared Error (MSE)	16
	6.1.2 Peak Signal to Noise Ratio (PSNR)	16
<b>7</b>	<b>CONCLUSION AND FUTURE WORK</b>	
	7.1 Conclusion	18

7.2 Future work	18
-----------------	----

<b>APPENDIX</b>	
<b>APPENDIX A: SOURCE CODE</b>	19
<b>REFERENCES</b>	30

## TABLE OF FIGURES

<b>FIGURE NO.</b>	<b>FIGURE TITLE</b>	<b>PAGE NO.</b>
1.1	Xiao Steganography	2p
1.2	Steghide	2
1.3	RSteg	3
1.4	OpenStego	3
1.5	Hide'N'Send	4
1.6	Use case diagram	12
1.7	Architecture diagram	13
1.8	Original Image	22
1.9	Stego Image	22
1.10	Compression	26

## **CHAPTER 1**

### **INTRODUCTION**

#### **1. PROBLEM DEFINITION**

Medical images are widely popular as it helps in detecting, staging and treating different kinds of diseases in the human body with the use of various scans. It is also a crucial part of a patients' electronic records. The recent advancements in telemedicine include teleconsulting, tele-diagnosis and telesurgery. There is a regular trade of medical images between hospitals and experts like radiologists and physicians. However, the exchanged data has to be secured as communication is done via open communication methods which may raise the chances of manipulation and misappropriation.

A patients' medical health records contain the medical history of a patient with their personal data, vital signs, diagnosis reports, laboratory reports, etc. These records are stored electronically in databases of medical organizations which is known as Electronic Health Record(EHR). 90% of these medical records are made of medical images like X-rays, endoscopy images and videos, MR images, etc. In order to protect these medical images, DICOM says that header information of the images should contain patient's information. Header file is embedded with a medical image and unique information about each patient. But, attacks and other malicious acts can be done to the header. Therefore extra protection is needed. To provide this extra protection steganography and cryptography are introduced to assist in the exchange of medical images.

#### **1.2 EXISTING APPLICATION**

##### **1.2.1. Xiao Steganography**

Xiao Steganography is a free and best Steganography tool that can be used to hide secret files in the image as well as audio files. The most frequently used file formats are BMP for Images and WAV for audio files. Open the tool, load the required files and the secret message into it. You can select any of the following algorithms for encryption like DES, DES 112, RC2. The hashing include SHA, MD4, MD2, and MD5. The same Xiao Steganography tool is used for decryption and it is simple than it sounds. Just drop the encrypted file and get the secret text visible.



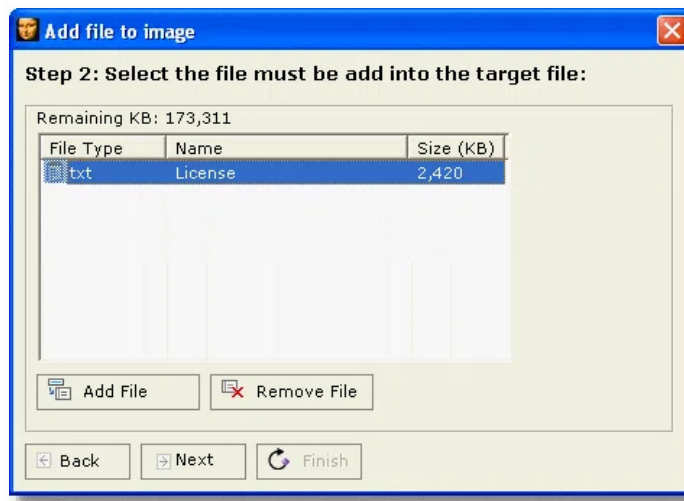


Fig 1.1 Xiao Steganography

### 1.2.2. Steghide

Steghide is able to hide image and audio files. The main features of Steghide include compression, encryption, and automatic integrity. The input file formats include JPEG, BMP, WAV, and AU (sound file). The secret data is not restricted to text. It can be any type of data. The encryption algorithm used here is Rijndael with a key of 128 bits (AES – Advanced Encryption Service). It is a command line software, hence recommended as best Steganography software for Mac and Linux users. Hence you should be aware of the basic commands to encrypt and decrypt to feel comfortable using Steghide.

```

C:\WINDOWS\system32\cmd.exe
E:\Downloads\Downloads\Compressed\steghide-0.5.1-win32_2\steghide>steghide --help
steghide version 0.5.1

the first argument must be one of the following:
embed, --embed          embed data
extract, --extract      extract data
info, --info            display information about a cover- or stego-file
info <filename>        display information about <filename>
encinfo, --encinfo      display a list of supported encryption algorithms
version, --version       display version information
license, --license      display steghide's license
help, --help            display this usage information

embedding options:
-cf, --embedfile        select file to be embedded
-cf <filename>          embed the file <filename>
-cf, --coverfile        select cover-file
-cf <filename>          embed into the file <filename>
-p, --passphrase         specify passphrase
-p <passphrase>         use <passphrase> to embed data
-sf, --stegofile         select stego file
-sf <filename>          write result to <filename> instead of cover-file
-e, --encryption         select encryption parameters
-e <a>[<n>]!<n>[<a>]    specify an encryption algorithm and/or mode
-e none                 do not encrypt data before embedding
-z, --compress           compress data before embedding (default)
-z <l>                  using level <l> (<l> best speed...9 best compression)
-u, --dontcompress       do not compress data before embedding
-n, --nochecksum         do not embed crc32 checksum of embedded data
-N, --dontembedname      do not embed the name of the original file
-f, --force              overwrite existing files
-q, --quiet              suppress information messages
-v, --verbose            display detailed information

extracting options:
-sf, --stegofile        select stego file
-sf <filename>          extract data from <filename>
-p, --passphrase         specify passphrase
-p <passphrase>         use <passphrase> to extract data
-xf, --extractfile       select file name for extracted data
-xf <filename>          write the extracted data to <filename>
-f, --force              overwrite existing files
-q, --quiet              suppress information messages
-v, --verbose            display detailed information

options for the info command:
-p, --passphrase         specify passphrase
-p <passphrase>         use <passphrase> to get info about embedded data

To embed enb.txt in cur.jpg: steghide embed -cf cur.jpg -sf enb.txt
To extract embedded data from stg.jpg: steghide extract -sf stg.jpg
E:\Downloads\Downloads\Compressed\steghide-0.5.1-win32_2\steghide>

```

Fig 1.2 Steghide

### 1.2.3. RSteg

RSteg is yet another Steganography tool developed using Java. You need to have Java installed on your machine to run RSteg. Another striking advantage is its portable feature. Hence no need to install it, just run and the software windows pop up. Performing Steganography using RSteg is simple. You need an Image file, text to be encrypted and password to be set for decryption. The final output is stored as PNG. Plug the image into the same Steganography detection tool for decryption along with a password.

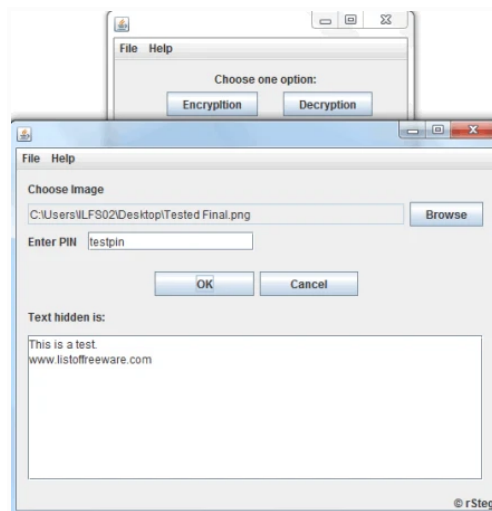


Fig 1.3 RSteg

### 1.2.4. OpenStego

OpenStego provides data hiding as well as Watermarking. Data hiding is the top priority when it comes to Steganography. Using OpenStego, you can perform Steganography effectively with image files of type JPEG, JPG, BMP, GIF, PNG etc. The output of OpenStego is a PNG file. It is an open source and free Steganography tool developed using Java. Watermarking provides an authority to imprint invisible water mark on the image file. It is used to detect an unauthorized copy of image files.

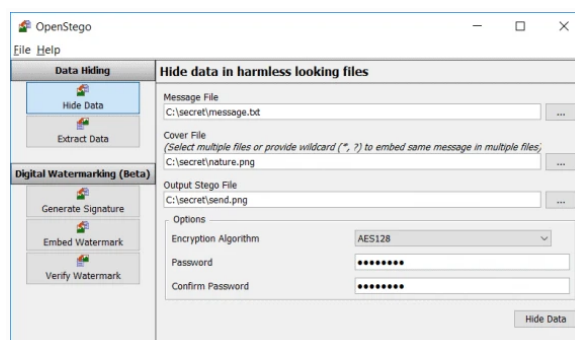


Fig 1.4 OpenStego

### 1.2.5. Hide'N'Send

Hide'N'Send is one of the best image Steganography tools. It includes encryption and hiding of data under an image file (JPEG format). It encrypts the data using the F5 steganography algorithm. Hiding of data is done using the LSB (Least Significant Bit) algorithm for Image Steganography. Instead of hiding in a file structure, these algorithms hide the data inside the image.



Fig 1.5 Hide'N'Send

## 1.3 NEED FOR THE SYSTEM

In the modern-day medical images are becoming more vital as they help patients determine the disease they have. In hospitals, clinics and recent telemedicine applications save their patients' information with their medical images on an unsecured database. These unsecured databases have allowed ethical action of posting patients' information on several websites. Even though patients may not be aware of the problem and that their data is available on many websites without their permission, but this can cause the use of patients' information in an unethical way without their knowledge and this violates their confidentiality and privacy.

Therefore, Steganography can be used for the secure transmission of information. It finds application in many different areas in which cryptography is not suitable also. Steganography provides good authentication guarantee also. It can be used to identify traffic congestion in network paths by incorporating suitable packets to messages. Steganography can be used for the safe communication of secret data between the organizations. Steganography finds greater applications in medical imaging systems where the privacy of patient data is very much important.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **[1] Securing Medical Images using Encryption and LSB Steganography**

This research paper focuses on preserving the confidentiality and integrity through encryption. Incorporates one more layer of security by applying steganography to the medical image. It uses Matlab. The process of using multi-level security by combining Cryptography and Steganography has decreased the possibilities of a security breach such as confidentiality and integrity. The projected system is found effective than applying only LSB, by the experimental results.

#### **[2] Steganography-Based Transmission of Medical Images Over Unsecure Network for Telemedicine Applications**

In image steganography, the major features are the cover object quality and hiding data capacity. Due to poor image quality, attackers could easily hack the secret data. The main aim of this study is to combine several steganography techniques, for the secure transmission of data without leakage and unauthorized access. The experimental results indicate that the proposed steganography method offers better security, imperceptibility, and robustness and requires less processing time as compared to existing methods.

#### **[3] A Safe and Secured Medical Textual Information Using an Improved LSB Image Steganography**

This study proposed a modified least significant bit (LSB) technique capable of protecting and hiding medical data to solve the crucial authentication issue. It uses MATLAB. The proposed secured medical information system is evidenced to be proficient in secreting medical information and creating undetectable stego images with slight entrenching falsifications when likened to other prevailing approaches.

#### **[4] An effective and secure digital image steganography scheme using two random function and chaotic map**

This paper suggests a secure image steganography scheme which is known as a new stego key adaptive LSB (NSKA-LSB) scheme, which depends on four stages for the

provision of a better data-hiding algorithm in cover images by the volume, image quality, and security. The results of the experiment revealed that the algorithm has a better image quality index, peak signal-to-noise ratio, and payload used in the evaluation of the stego image.

#### **[5] A Methodology Based on Steganography and Cryptography to Protect Highly Secure Messages**

This paper will introduce, implement and test a novel methodology that can be used as a secure and highly efficient method of data hiding and data extracting. The process of hiding a secret message can be realized by applying the following phases: Steganography, Cryptography, and Message Extraction. The proposed methodology increased the security level by using 2 private keys and enhanced the efficiency compared with other existing methods.

#### **[6] A modification of least significant digit (LSD) digital watermark technique**

In this research work, the authors implemented the use of the Least Significant Digit (LSD) Digital Watermark Technique and at the same time, the method of optimizing preference is employed. The exact reliability was not attained through the unsystematic choosing of pixel's value. Optimizing preference lessens the quantity of pixel value that was altered. The digital watermark and digital cover image were in grayscale.

#### **[7] Performance analysis of LSB-based data hiding techniques**

The performance and juxtaposition of image evaluation and histogram evaluation for each one of the pixel layers were conducted in the article. The authors presented that the data hiding approach with individual bpp implanting proportion was strenuous to differentiate between the cover and stego objects.

#### **[8] Image steganography for authenticity of visual contents in social networks**

The study made use of the gleaming level surface of contributed representation for concealed records by utilizing Morton scrutinizing the precise least significant bits replacement technique. The undisclosed records were scrambled employing a trio-parallel encoded process preceding implanting, and this led to enhancing an extra steady safety for validation.

#### **[9] Hiding shares by multimedia image steganography for optimized counting-based secret sharing**

The authors used multimedia image-based steganography methods to store the optimized shares that are providing comparisons for proofed remarks. The paper experiments measure the function of the improvements by assuming various hidden sharing key sizes of 64-bit, 128-bit, and 256-bit to ensure that real differences within the security analysis. The usability of the shares was further enhanced by experimenting with five different image-based steganography techniques to embed each produced share.

#### **[10] Encryption and Decryption of Color Images using Visual Cryptography**

The images are transmitted after applying the visual cryptographic technique. The hacker cannot understand the distorted image and thus the data communication become secured. It exploits the human visual system to read the secret message from some overlapped shares. This technique overcomes the disadvantage of complex computations required in traditional cryptography. Visual cryptography can also be applied to color images by converting them into black and white binary images. In this research work, visual cryptographic technique is proposed and encryption and decryption are done using Blowfish algorithm.

## **CHAPTER 3 SYSTEM ANALYSIS**

### **3.1SYSTEM REQUIREMENTS**

The following specifications were those required by the system for the software's successful implementation and functioning.

#### **3.1.1REQUIREMENTS**

##### **SOFTWARE REQUIREMENTS**

- o PYTHON
- o GOOGLE COLABORATORY
- o ENCRYPTION ALGORITHM - DNA BASED AES
- o STEGANOGRAPHY ALGORITHMS - LSB
- o COMPRESSION ALGORITHM – DISCRETE WAVELET TRANSFORMATION

##### **PYTHON**

The python programming language is a high-level, object-oriented, and interpreted programming language. Python is of best use when it comes to Rapid Application Development and this due to the high-level built-in data structures which are combined with dynamic typing and dynamic binding. This programming language has a simple and easy-to-learn syntax which emphasizes readability and reduces the overall cost of program maintenance. This high-level programming language can also be used as a scripting or glue language to connect the existing components. It supports a lot of modules and packages and encourages program modularity and code reusability. Python has an interpreter and an extensive standard library that are made available in source or binary form without charge for all major platforms, and it is an open-source language.

##### **GOOGLE COLABORATORY**

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.

## **ENCRYPTION ALGORITHM - DNA BASED AES**

DNA cryptography is a new promising direction in cryptography research that emerged with the progress in the DNA computing field. Traditional cryptographic systems have a long legacy and are built on a strong mathematical and theoretical basis. So, an important perception needs to be developed that the DNA cryptography is not to negate the tradition, but to create a bridge between existing and new technology. The power of DNA computing will strengthen the existing security systems by opening up a new possibility of a hybrid cryptographic system.

Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.

That means it takes 128 bits as input and outputs 128 bits of encrypted cipher text as output. AES relies on substitution-permutation network principle which means it is performed using a series of linked operations which involves replacing and shuffling of the input data.

## **STEGANOGRAPHY ALGORITHMS - LSB**

LSB-Steganography is a steganography technique in which we hide messages inside an image by replacing Least significant bit of image with the bits of message to be hidden. By modifying only the first most right bit of an image we can insert our secret message and it also makes the picture unnoticeable, but if our message is too large it will start modifying the second rightmost bit and so on and an attacker can notice the changes in the picture.

## **COMPRESSION ALGORITHM - DISCRETE WAVELET TRANSFORMATION**

In wavelet analysis, the Discrete Wavelet Transform (DWT) decomposes a signal into a set of mutually orthogonal wavelet basis functions. These functions differ from sinusoidal basis functions in that they are spatially localized – that is, nonzero over only part of the total signal length. Furthermore, wavelet functions are dilated, translated and scaled versions of a common function  $\phi$ , known as the mother wavelet. As is the case in Fourier analysis, the DWT is invertible, so that the original signal can be completely recovered from its DWT representation.

### **3.1.2PACKAGES**



These are the packages that were downloaded and utilized in our project

- pycryptodome==3.14.1
- cryptography==37.0.2
- stepic==0.5.0

## **Pycryptodome**

PyCryptodome is a self-contained Python package of low-level cryptographic primitives. It supports Python 2.7, Python 3.5 and newer, and PyPy. You can install it with: `pip install pycryptodome` All modules are installed under the `Crypto` package.

## **Cryptography**

Cryptography is a package which provides cryptographic recipes and primitives to Python developers. Our goal is for it to be your “cryptographic standard library”. It supports Python 3.6+ and PyPy3 7.2+.

Cryptography includes both high level recipes and low-level interfaces to common cryptographic algorithms such as symmetric ciphers, message digests, and key derivation functions.

## **Stepic**

Stepic hides arbitrary data inside Pillow images. Stepic is a Python module and command-line tool for hiding arbitrary data within images by slightly modifying the colours. These modifications are generally imperceptible to humans but are machine detectable. Works with RGB, RGBA, or CMYK images. Does not work with JPEG or other lossy compression schemes.

## **CHAPTER 4**

### **SYSTEM DESIGN**

#### **4.1 OBJECT ORIENTED DESIGN**

Identifying the objects in a system would be what OO (Object Oriented) analysis and design have always been about identifying their relationships. Generating a design that could be transformed into executables utilising object-oriented programming languages.

UML (Unified Modelling Language) is a standard language that uses, designs, constructing, and documenting software components. The Unified Modelling Language (UML) is an effective instrument enabling Object-Oriented Analysis and Design. The Object Management Group (OMG) created it, and in January 1997, the OMG proposed UML 1.0 as a specification draught. It started as a way to capture the behaviour of vast software and non-software systems and has since grown into such an international standard. UML is created to be process generic, indicating it could be used in a variety of circumstances. It can be used for a spectrum of uses. Business analysts, software architects, and developers are all using UML as a common language. It can be used to describe, specify, build, and document the system's business processes, including its structural and behavioural artefacts.

UML is a modelling language used to create software blueprints. Diagrams are categorized into two divisions, which are further divided into subcategories:

- **Structural Diagrams**
- **Behavioral Diagrams**

#### **4.2 STRUCTURAL DIAGRAMS**

The static aspect of the system is portrayed by the structural diagrams. These static aspects are the components of a diagram that describes the main structure and are thus stable. Classes, interfaces, objects, components, and nodes are often used to represent them. Some of the structural diagrams are:

- Class diagram
- Object diagram
- Component diagram

- Deployment diagram

### 4.3 BEHAVIORAL DIAGRAMS

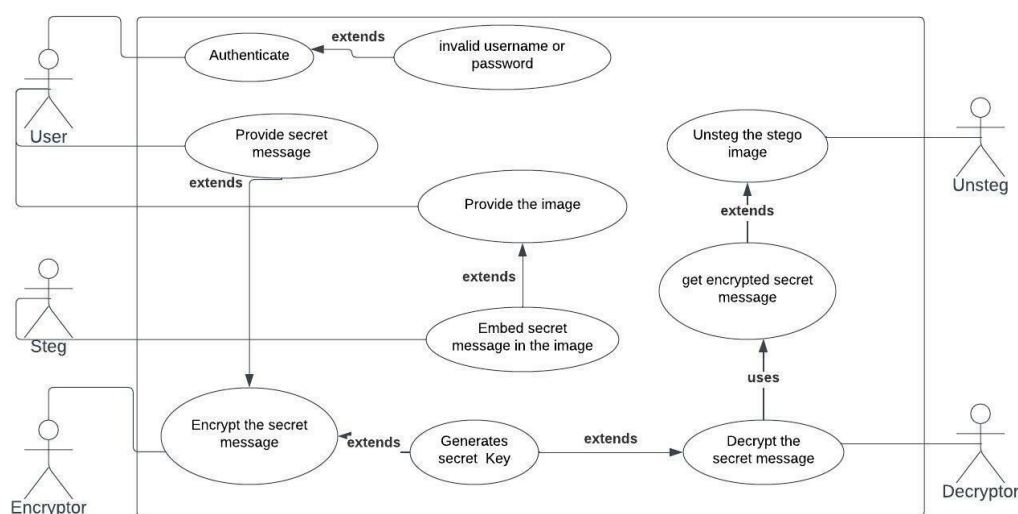
Behavioral diagrams illustrate a system's complex nature. The changing/moving parts of a system are usually known for the dynamic aspect. The following five types of behavioural diagrams are supported in UML:

- Use case diagram
- Sequence diagram
- Collaboration diagram
- State chart diagram
- Activity diagram

### 4.4 USE CASE DIAGRAM

The use case diagrams show the system's behavior concerning to the deployment environment. It illustrates the proposed system's users. A use case is a system analysis methodology for finding, explaining, and monitoring program's needs.

A use case is a collection of conceivable sequences of interactions between systems and users in a specific environment, all of which are tied to a specific purpose. A use case is represented by an ellipse with the name of the use case. A stick figure with a name is used to represent an actor.

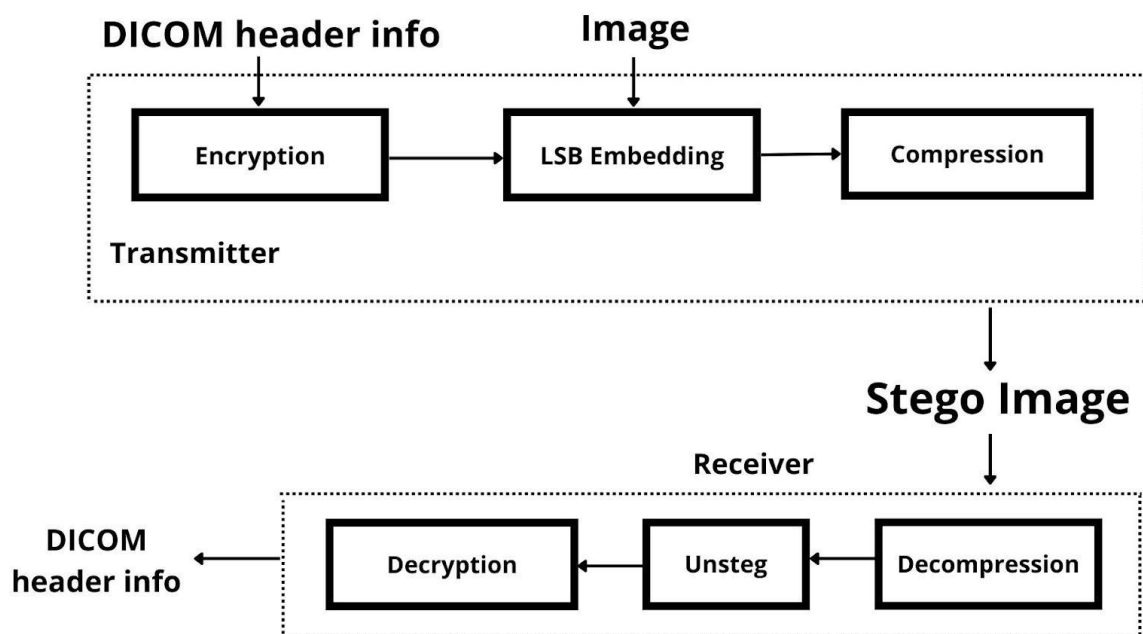


*Fig 1.6 Use Case diagram*

## 4.5 ARCHITECTURE DIAGRAM

Two inputs are requested from the user. One is the secret medical data and the other is the medical image on which the secret medical data will be embedded. The medical image is pre-processed and the secret medical data is encrypted generating public and private keys. Using LSB Embedding the secret data is embedded on the image. Then the stego image is de-noised as there may be a few disruptions created when embedding. And further the image is composed as the de-noising process would have enlarged the stego image.

An image with little to no difference to the plain sight will be generated. On the other hand, the receiver unstegs the stego image and uses the generated key to decrypt the secret medical data.



*Fig 1.7 Architecture diagram*

## **CHAPTER 5**

### **SYSTEM IMPLEMENTATION**

#### **5.1 ALGORITHM AND PROPOSED TECHNIQUES**

##### **5.1.1 DNA based Encryption algorithm**

The method proposed encrypts a secret message which is the patient record that contains their name, gender, birthdate, SSN, medical history, and diagnosis. The patient record is encrypted using AES with a key size of 128 bits. Firstly, the secret message is divided into pieces wherein each piece contains 8 characters, and if the last few characters aren't equal to 8 then white spaces are added to create 8 characters. After that, each piece is encrypted using the created AES key in a loop, where inside the loop the binary representation and initialization vector of each encrypted piece is taken and added in separate variables. When the loop is finished the binary representation and initialization vector is concatenated together, then converted into a string to be able to embed it in the LSB of the medical image.

##### **5.1.2 LSB Embedding for Steganography**

Next, the embedding the encrypted secret message into the LSB of the medical image. Firstly, the medical image is converted into bytes where the LSB of each byte is set to zero. Then, the encrypted secret message is converted into a sequence of bits to be inserted in the empty bits in the medical image. Lastly, by replacing the bits of the secret encrypted message in the LSB in each byte of the medical image; a stego image is created that looks the same as the original medical image, however it contains the secret message.

##### **5.1.3 Discrete Wavelet Transformation**

In numerical analysis and functional analysis, a discrete wavelet transform (DWT) is any wavelet transform for which the wavelets are discretely sampled. As with other wavelet transforms, a key advantage it has over Fourier transforms is temporal resolution: it captures both frequency and location information.

In wavelet analysis, the Discrete Wavelet Transform (DWT) decomposes a signal into a set of mutually orthogonal wavelet basis functions. These functions differ from sinusoidal basis functions in that they are spatially localized – that is, nonzero over only part of the total signal length. Furthermore, wavelet functions are dilated, translated and scaled versions of a common function  $\phi$ , known as the mother wavelet. As is the case in Fourier analysis, the DWT is invertible, so that the original signal can be completely recovered from its DWT representation.

Unlike the DFT, the DWT, in fact, refers not just to a single transform, but rather a set of transforms, each with a different set of wavelet basis functions. Two of the most common are the Haar wavelets and the Daubechies set of wavelets. Here, we will not delve into the details of how these were derived; however, it is important to note the following important properties:

1. Wavelet functions are spatially localized;
2. Wavelet functions are dilated, translated and scaled versions of a common mother wavelet; and
3. Each set of wavelet functions forms an orthogonal set of basis functions.

#### **5.1.4 Extraction**

In the extraction process, the least significant bit of every byte is pulled, then grouped to 8 bits to be converted back to a text. This text will be the ciphertext or encrypted secret message.

#### **5.1.5 Decryption**

The extracted encrypted secret message is decrypted to get the secret message. Firstly, the binary representation and the initialization vector will be separated into several parts to create their corresponding pieces that contain 8 characters. Then, each part is taken and decrypted using the AES key that was used to encrypt it. Lastly, after decrypting each piece all the pieces are joined together to get the plaintext secret message.

## CHAPTER 6

### PERFORMANCE METRICS

#### 6.1 Evaluation metrics:

##### 6.1.1 Mean Squared Error (MSE):

Mean Square Error is the averaged value of the square of the pixel-by-pixel difference between the original image and stego-image. It gives us a measure of the error produced in the cover image due to the data embedding process.

$$MSE = \frac{\sum_{i=1}^m \sum_{j=1}^n [I(i,j) - K(i,j)]^2}{(m*n)}$$

A lower value of MSE indicates a good quality embedding.

##### **Description:**

m,n -> dimensions of the image

I -> original image

K -> stego image

##### 6.1.2 Peak Signal to Noise Ratio (PSNR) :

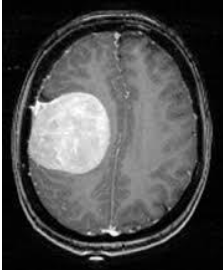
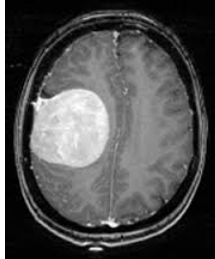
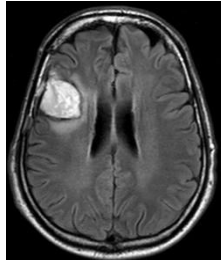
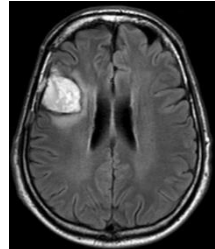
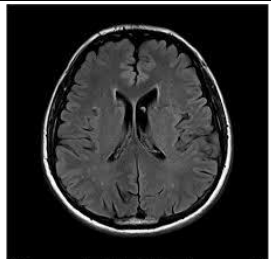
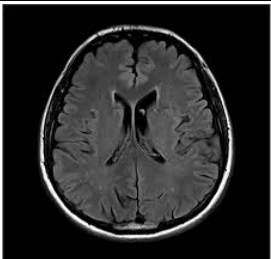
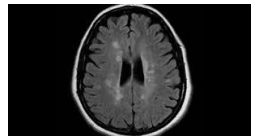
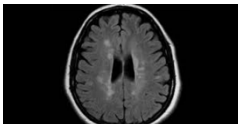
PSNR is another popular way to measure the degree of distortion in the cover image due to embedding. It is the ratio between the maximum possible value of a signal and the power of distortion noise (MSE). It is measured in dB's. A higher value of PSNR indicates a better-quality embedding.

$$PSNR = 10 * \frac{MAX^2}{MSE}$$

##### **Description:**

MAX -> 255 for a 8 bit grayscale image

Table 6.1 Evaluation of the proposed scheme

Original Cover Image	Stego Image	Performance Metrics
		MSE = 0.0302 PSNR = 63.3309 dB
		MSE = 0.0063 PSNR = 70.1539 dB
		MSE = 0.0215 PSNR = 64.8082 dB
		MSE = 0.0242 PSNR = 64.2903 dB

The results represent that the risk of an intruder to know the existence of a secret message is extremely low almost non-existent, due to the low value of the MSE and the high value of the PSNR.



## **CHAPTER 7**

### **CONCLUSION AND FUTURE WORKS**

#### **7.1 CONCLUSION**

This project presents a proposed steganography method, which is a combination of cryptography and steganography techniques, for secure transmission of secret data over unsecure network from sender to receiver without any data loss or modification or unauthorized access. The combined techniques are DNA based AES, LSB and Discrete wavelet transform. Firstly, public and private keys with digital signature are generated for the purpose of authentication. Then, the secret data is encrypted with a generated secret key using the DNA based AES algorithm. Next, the LSB embedding is carried out to hide the secret data inside the cover image. Finally, the Discrete wavelet transform is used to compress the obtained stego image before sending it to the receiver. At the receiver end, the reverse of the proposed processes is performed to extract the embedded data from the stego image. The experimental results indicate that the proposed steganography method offers better security, authenticity and robustness and requires less processing time as compared to existing methods.

#### **7.2 FUTURE WORK**

A complete blockchain system for EHR storage can be built by extending this project. PSNR values can be increased by experimenting with various algorithms.

## APPENDIX

### APPENDIX A - SOURCE CODE

#### A.KEY GENERATION

```
from hashlib import sha256
import base64
from Crypto import Random
from Crypto.Cipher import AES
import pandas as pd
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.serialization import load_pem_private_key
from cryptography.hazmat.primitives.serialization import load_pem_public_key
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.exceptions import InvalidSignature
from cryptography.hazmat.primitives import serialization, hashes

def gen_key():
    private_key = rsa.generate_private_key(
        public_exponent=65537, key_size=2048, backend=default_backend())

    public_key = private_key.public_key()
    return private_key, public_key

def save_pvkkey(pk, filename):
    pem = pk.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.TraditionalOpenSSL,
        encryption_algorithm=serialization.NoEncryption()
    )
    with open(filename, 'wb') as pem_out:
        pem_out.write(pem)

def save_pukey(pk, filename):
    pem = public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    )
    with open(filename, 'wb') as pem_out:
        pem_out.write(pem)

private_key, public_key = gen_key()

save_pvkkey(private_key, 'private_key')
save_pukey(public_key, 'public_key')
print("private key and public key generated.")
```

**O/P:** private key and public key generated.

## B. SENDER END

```
from hashlib import sha256
import base64
from Crypto import Random
from Crypto.Cipher import AES
import pandas as pd
from cryptography.hazmat.primitives.serialization import load_pem_private_key
from cryptography.hazmat.primitives.serialization import load_pem_public_key
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.exceptions import InvalidSignature
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.asymmetric import padding
from PIL import Image
import stepic
```

## DNA CRYPTO

```
DNA_data = {
"words":["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W",
        "X","Y","Z"," ",".",":","0","1","2","3","4","5","6","7","8","9"],
        "DNA_code":
["CGA","CCA","GTT","TTG","GGC","GGT","TTT","CGC","ATG","AGT","AAG","TGC","TCC","TCT","GGA","GT
G",
        "AAC","TCA","ACG","TTC","CTG","CCT","CCG","CTA","AAA","CTT","ACC","TCG","GAT","GCT
","ACT","TAG",
        "ATA","GCA","GAG","AGA","TTA","ACA","AGG","GCG"]
}
```

```
DNA_df = pd.DataFrame.from_dict(DNA_data)
#DNA_df
with open('info.txt','r') as f:
    word = f.read().upper()
#message = input("Please enter your message: ")
DNA_crypto_message = ""
```

```
for i in word:
    DNA_crypto_message+= str(DNA_df.loc[ DNA_df['words'] == i, 'DNA_code' ].iloc[0])

print(DNA_crypto_message)
```

**O/P:** Please enter your message: Name : George Mendes, Gender : Male, Birthdate : 5.9.1995, SSN : 15657834939, Medical History : Diabetes, Diagnosis : broken arm

```
TCTCGATCCGGCACCGCTACCTTTGGCGGATCATTTGGCACCTCCGGCTCTTTGGGCACGTCGACCTTTGGCTCTTT
GGGCTCAACCGCTACCTCCCGATGCGGCTCGACCCCAATGTTCATTCCGCTTGCGATTCCGGCACCGCTACCAGAGATG
CGGATTAGGCGGCGAGATCGACCACGACGTCTACCGCTACCTAGAGATTAAGAACAAGGGCAGAGGCGGCAGCGTCG
ACCTCCGGCTTGATGGTTCGATGCACCCGCATGACGTTTCGGATCAAAAACCGCTACCTTGATGCGACCAGGCTTCGG
CACGTCGACCTTGATGCGATTTTCTGGAACGATGACGACCGCTACCCCATCAGGAAAGGGCTCTACCCGATCATCC
```

## AES CRYPTO

```
BS = 16
pad = lambda s: bytes(s + (BS - len(s) % BS) * chr(BS - len(s) % BS), 'utf-8')
unpad = lambda s : s[0:-ord(s[-1:])]
```

```
class AESCipher:
```

```
    def __init__( self, key ):
        self.key = bytes(key, 'utf-8')
```

```
    def encrypt( self, raw ):
        raw = pad(raw)
        iv = Random.new().read( AES.block_size )
        cipher = AES.new(self.key, AES.MODE_CBC, iv )
        return base64.b64encode( iv + cipher.encrypt( raw ) )
```

```
cipher = AESCipher('LKHlhb899Y09oIUi')
AES_encrypted_message = cipher.encrypt(DNA_crypto_message)
```

```
print(AES_encrypted_message)
```

**O/P:**

```
b'60OE5hisAOH0+IGeTjKpG4XMyPvqLKDwAhGc+5zLsEoMN5kdogzhIQmceD/crmn5PRBefnewpB8iAu8gz
MpMMPBH8BvCCDVebmp6MQzAX2Ke+RKcL3sJ12haFPE7EuElHsCRyqeOgMYVtrm7ONNaHf6QUxqP8gN
yLaKBW9DH56+MHGuyNqCqR21zcmsWfjr2SNXNa+EgOyd8rPX8RQ2XsUQoQMS+fdgUNPSCUbHplpx7d3h
sTFSh+juCqvdcw1gVK8wl+y0+7lr62/coRZwp9ZXPOjfM1BqiKOAqMbgcppmE9NMjyhUbCyhwkRjO/Hx+PYM8
1LJHNezvW+v23+pD9ZNa+t4kP1pYQP/XW6u0Dvk29pj5q226U0N6QU/yDk5P0EHPGZUk+Vj+qlxRDegcZFz
1o//Va3hkh+POZDFtlidUrQXVVnRSBLfk6MclZ9W+eOBXIGWahGKGm80cE7MBg7rDb1P8HI4a0sVy6L9pbW
V924MXungzI7JlqX+3y7e9CLK8vF7llmjJGGLmN74NAeZ6UCymzeZuPbT0FGiK4g='
```

## DIGITAL SIGNATURE

```
def load_pvkey(filename):
    with open(filename, 'rb') as pem_in:
        pemlines = pem_in.read()

    private_key = load_pem_private_key(pemlines, None, default_backend())
    return private_key

message = AES_encrypted_message
private_key = load_pvkey("private_key")
signature = private_key.sign(message, padding.PSS(mgf=padding.MGF1(hashes.SHA256()),
                                                    salt_length=padding.PSS.MAX_LENGTH), hashes.SHA256())
```

## SECRET MESSAGE GENERATION

```

im = Image.open('original_image.jpg')
#Encode some text into your Image file and save it in another file
secret_msg = AES_encrypted_message + bytes("SIGNATURE", 'utf-8') + signature

```

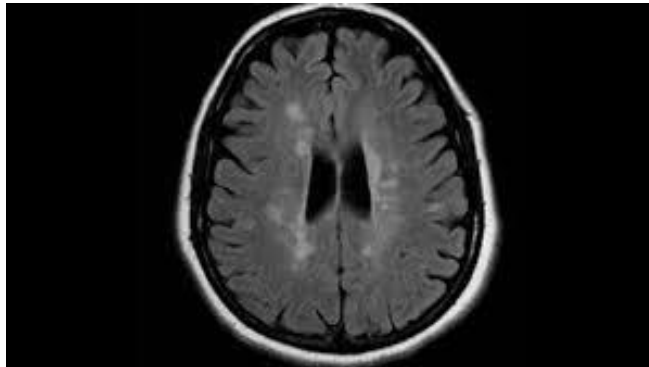


Fig 1.8 Original Image

## LSB STEGANOGRAPHY

```

im1 = stepic.encode(im, secret_msg)
im1.save('encoded_image.png', 'PNG')

```

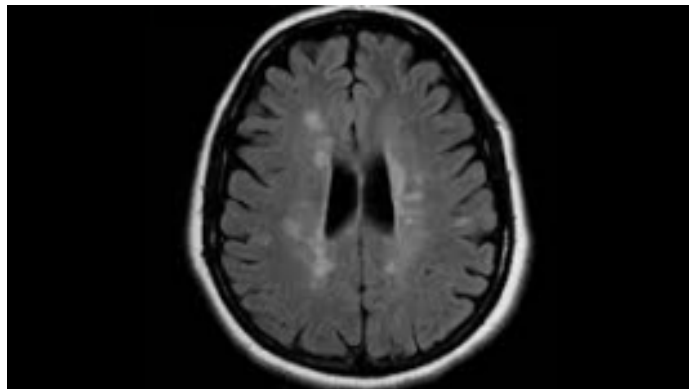


Fig 1.9 Stego Image

```

def PSNR(original, steg):
    mse = np.mean((original - steg) ** 2)
    if(mse == 0):
        return 100
    max_pixel = 255.0
    psnr = 20 * log10(max_pixel / sqrt(mse))
    return psnr

```

```

original=cv2.imread("original_image.jpg")
steg=cv2.imread("encoded_image.png")
value = PSNR(original,steg)

```

```

print(f"PSNR value: {value} dB")

```

**O/P:**

```

MSE: 0.024212962962962964

```

PSNR value: 64.29032423151567 dB

## IMAGE COMPRESSION

```
import sys, os, time, numpy, pywt
import matplotlib.pyplot as plt
from PIL import Image

def wavelet_transform(data, threshold):
    wavelet_type = 'haar'
    clean_coef = list()
    compose = list()

    cA2, cD2, cD1 = pywt.wavedec2(data, wavelet_type, level=2)
    clean_coef.append(cA2)
    clean_coef.append(cD2)

    for c in cD1:
        compose.append(numpy.where(((c<(-threshold)) | (c>threshold)), c, 0))
    clean_coef.append(tuple(compose))

    t = pywt.waverec2(clean_coef, wavelet_type)
    values = t.astype(int)
    return values

def create_image(image, values, threshold):
    matrix = list()
    for value in values:
        row = list()
        for v in value:
            row.append((int(v), int(v), int(v)))
        matrix.append(row)

    width, height = image.size
    new_image = Image.new('RGB', (width, height))
```

```

new = new_image.load()
for w in range(width):
    for h in range(height):
        new[w, h] = matrix[h][w]

image_name = str(threshold) + '.png'
new_image.save(image_name)
return new_image

```

```

def grayscale(image):
    width, height = image.size
    pixels = image.load()

    for w in range(width):
        for h in range(height):
            r, g, b = pixels[w, h]
            gray = (r+g+b)//3
            pixels[w, h] = (gray, gray, gray)
    return image

```

```

def get_rows_values(image):
    width, height = image.size
    pixels = image.load()
    matrix = list()

    for j in range(height):
        row = list()
        for i in range(width):
            pixel_value = pixels[i, j][0]
            row.append(pixel_value)
        matrix.append(row)

    array = numpy.array(matrix)
    return array

```

```

def compress(image_path, threshold):
    image = Image.open(image_path).convert('RGB')
    image = grayscale(image)

    data = get_rows_values(image)
    values = wavelet_transform(data, threshold)

    newimage = create_image(image, values, threshold)
    return compressed_percentage(image_path, threshold)

def compressed_percentage(image_path, threshold):
    original_size = os.path.getsize(image_path)
    image_name = str(threshold) + '.png'
    final_size = os.path.getsize(image_name)
    percentage = 100 - (final_size*100)//float(original_size)
    print ('Image compressed at %0.2f%%' % percentage)
    return percentage

def main():
    image_path = "encoded_image.png"

    time_list = list()
    percentages_list = list()
    thresholds_list = list()
    for threshold in range(0, 200, 20):
        start_time = time.time()
        compressed_percentage = compress(image_path, threshold)
        end_time = time.time()
        process_time = end_time - start_time
        time_list.append(process_time)
        percentages_list.append(compressed_percentage)
        thresholds_list.append(threshold)

```



```

p = plt.plot(thresholds_list, percentages_list, 'bo-', label='Percentage')
plt.legend(loc='upper left', numpoints=1)
plt.ylabel('Percentage')
plt.xlabel('Threshold value')
plt.title('Percentage vs. Threshold value')
plt.show()

```

```

average_time = sum(time_list)//len(time_list)
print ('The average time is', average_time)

```

```

if __name__ == '__main__':
    main()

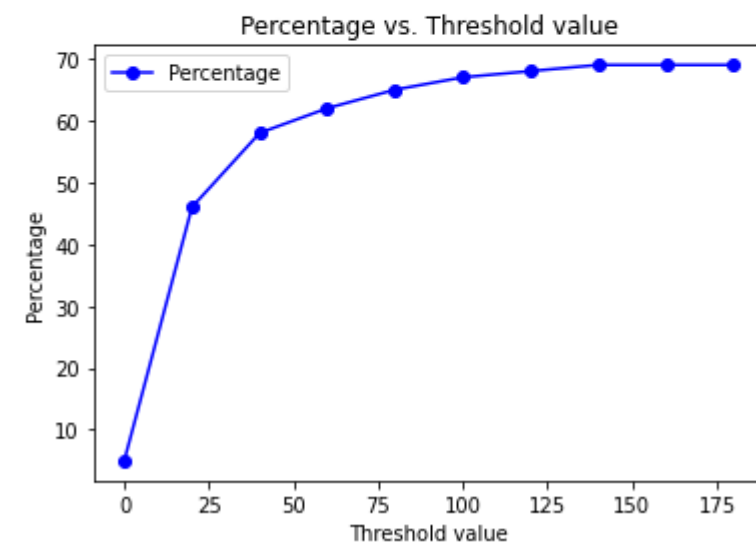
```

**O/P:**

```

Image compressed at 5.00%
Image compressed at 46.00%
Image compressed at 58.00%
Image compressed at 62.00%
Image compressed at 65.00%
Image compressed at 67.00%
Image compressed at 68.00%
Image compressed at 69.00%
Image compressed at 69.00%
Image compressed at 69.00%

```



```

The average time is 0.0

```

Fig 1.10 Compression

## C. TARGET END

```
from hashlib import sha256

import base64

from Crypto import Random

from Crypto.Cipher import AES

import pandas as pd

from cryptography.hazmat.primitives.serialization import load_pem_private_key

from cryptography.hazmat.primitives.serialization import load_pem_public_key

from cryptography.hazmat.backends import default_backend

from cryptography.hazmat.primitives.asymmetric import rsa

from cryptography.exceptions import InvalidSignature

from cryptography.hazmat.primitives import serialization, hashes

from cryptography.hazmat.primitives.asymmetric import padding

from PIL import Image

import stepic
```

## SIGNATURE AND MESSAGE DECODING

```
im = Image.open('encoded_image.png')
stegoImage = stepic.decode(im)

ind_sep = stegoImage.find('SIGNATURE')
message = bytes(stegoImage[:ind_sep], 'utf-8')
signature = bytes(stegoImage[ind_sep+9:], 'latin1')
```

## VERIFY DIGITAL SIGNATURE

```
def load_pukey(filename):
    with open(filename, 'rb') as pem_in:
        pemlines = pem_in.read()
        public_key = load_pem_public_key(pemlines, default_backend())
    return public_key

public_key = load_pukey("public_key")
try:
    public_key.verify(signature, message, padding.PSS(mgf=padding.MGF1(hashes.SHA256())),
```

```

        salt_length=padding.PSS.MAX_LENGTH),hashes.SHA256())

print(message)
except InvalidSignature:
    print('Invalid!')

```

**O/P:**

```

b'DB0/gAlilqNt22n0w2m2Ni32U2fBYIVO+Vb0D8Z+nOxmv+DgPOFULpB6sopyE27jgY3+n3m5pK3iOfKZSST
Mc8KjGbxayDNi1Zw0tDv3UkEKboUOANoHKGoUFIK/0c1GZCqTkDBZsLHGgQV5iDWXChdUvg5HpvdqPH1
Pyjiz6vunkoYYSrgD1aVH0phnbGuUw38W3OZEfvOBbPxIff+T4cL/MPWeCdPr7G1X6HJ6/Gut5JYaKIUA4E4IU
SpzVQl4QTuDmi7RbWc3/e5KvAT3cDtHh3S7BZ2wtK5ysRx530WCR1+U/jFG5WqC5Gza9NB1pKfTGtG5m8w
5OcHgrP6amnaKY6yhkOigr/oHKR/EqnJ0SekHqxvo5Fkq7gmqzM6elmrw6kVoVPTi1iEjepCIEGqKfIPdINwHP
YQysty3jq/y3ytpNylXfcJDJ+g04u8atIFv88iG46R5aWdu1wR0ECmfc+UXb/jAQTyKgdEhdSdPGFcGyY0SVmuJ
mmFL/Uzpv7tPF8ZwclAaKDQe/xZQtI3ngfp99NiG/vwT4Xdg='

```

## AES DECODING

BS = 16

```
pad = lambda s: bytes(s + (BS - len(s) % BS) * chr(BS - len(s) % BS), 'utf-8')
```

```
unpad = lambda s : s[0:-ord(s[-1:])]
```

```
class AESCipher:
```

```

    def __init__( self, key ):
        self.key = bytes(key, 'utf-8')

```

```

    def decrypt( self, enc ):
        enc = base64.b64decode(enc)
        iv = enc[:16]
        cipher = AES.new(self.key, AES.MODE_CBC, iv )
        return unpad(cipher.decrypt( enc[16:] )).decode('utf8')

```

```
cipher = AESCipher('LKHlhb899Y09olUi')
```

```
AES_decrypted = cipher.decrypt(message)
```

```
print(AES_decrypted)
```

**O/P:**

```

TCTCGATCCGGCACCGCTACCTTTGGCGGATCATTTGGCACCTCCGGCTCTTTGGGCACGTCGACCT
TTGGCTCTTTGGGCTCAACCGCTACCTCCCGATGCGGCTCGACCCCAATGTCATTCCGCTTGCGATT
CGGCACCGCTACCAGAGATGCGGATTAGGCGGCGAGATCGACCACGACGTCTACCGCTACCTAGAGA
TTAAGAACAAGGGCAGAGGCGGCAGCGTCGACCTCCGGCTTGATGGTTGATGCGACCCGCATGACGT
TCGGATCAAAAACCGCTACCTTGATGCGACAGGCTTCGGCACGTCGACCTTGATGCGATTTTCTGG
AACGATGACGACCGCTACCCCATCAGGAAAGGGCTCTACCCGATCATCC

```

## DNA DECODING

```

DNA_data = {
"words":["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W",
        "X","Y","Z"," ",".",":","0","1","2","3","4","5","6","7","8","9"],
"DNA_code":
["CGA","CCA","GTT","TTG","GGC","GGT","TTT","CGC","ATG","AGT","AAG","TGC","TCC","TCT",
,"GGA","GTG",
    "AAC","TCA","ACG","TTC","CTG","CCT","CCG","CTA","AAA","CTT","ACC","TCG",
,"GAT","GCT","ACT","TAG",
    "ATA","GCA","GAG","AGA","TTA","ACA","AGG","GCG"]
}

```

```

DNA_df = pd.DataFrame.from_dict(DNA_data)

```

```

l = [AES_decrypted[i:i+3] for i in range(0, len(AES_decrypted), 3)]

```

```

original_message = ""
for i in l:
    original_message+= str(DNA_df.loc[ DNA_df['DNA_code'] == i , 'words' ].iloc[0])
print("The secret message is: ",original_message.lower())

```

### O/P:

```

The secret message is:  name : george mendes, gender : male,
birthdate : 5.9.1995, ssn : 15657834939, medical history :
diabetes, diagnosis : broken arm

```

## REFERENCES

- [1] P. A, U. R, J. N and P. S, "Securing Medical Images using Encryption and LSB Steganography," 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), 2021, pp. 1-5, doi: 10.1109/ICAECT49130.2021.9392396.
- [2] Mansour, Romany & Girgis, Moheb. (2021). Steganography-Based Transmission of Medical Images Over Unsecure Network for Telemedicine Applications. *Computers, Materials & Continua*. 68. 4069-4085. 10.32604/cmc.2021.017064.
- [3] Ogundokun, Roseline & Abikoye, Oluwakemi. (2021). A Safe and Secured Medical Textual Information Using an Improved LSB Image Steganography. *International Journal of Digital Multimedia Broadcasting*. 2021. 1-8. 10.1155/2021/8827055.
- [4] Mohanad Najm Abdulwahed (2020). An effective and secure digital image steganography scheme using two random function and chaotic map. *Journal of Theoretical and Applied Information Technology*. 2020. 1817-3195.
- [5] Rasras, Rashad & Alqadi, Ziad & Rasmi, Mutaz & Abu Sara, Mutaz. (2019). A Methodology Based on Steganography and Cryptography to Protect Highly Secure Messages. *Engineering, Technology & Applied Science Research*. 9. 3681-3684. 10.48084/etasr.2380.
- [6] Asad, Naseem & Shayeb, Ismail. (2018). A Modification of Least Significant Digit (LSD) Digital Watermark Technique. *International Journal of Computer Applications*. 179. 4-6. 10.5120/ijca2018916718.
- [7] Jung, Ki-Hyun. (2018). Performance Analysis of LSB-based Data Hiding Techniques. *Journal of Signal Processing Systems*.
- [8] Muhammad, K., Ahmad, J., Rho, S. *et al*. Image steganography for authenticity of visual contents in social networks. *Multimed Tools Appl* 76, 18985–19004 (2017). <https://doi.org/10.1007/s11042-017-4420-8>
- [9] Gutub, Adnan & Al-Ghamdi, Maimoona. (2020). Hiding shares by multimedia image steganography for optimized counting-based secret sharing. *Multimedia Tools and Applications*. 79. 10.1007/s11042-019-08427-x.

[10] M., Karolin & Thirunavukkarasu, Meyyappan & Thamarai, S.M.. (2018). Encryption and decryption of color images using visual cryptography. International Journal of Pure and Applied Mathematics. 118. 277-280.