

An image steganography approach based on AES

Submitted in partial fulfilment of the requirements for the award of

Bachelor of Engineering in

Electronics and Communication Engineering

by

P.C JAYAMOHAN (Reg.No:37130165)

S.HARIHARAN (Reg.No:37130155)



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

**Accredited with Grade “A” by NAAC JEPPIAAR NAGAR, RAJIV GANDHI
SALAI, CHENNAI - 600 119**



SATHYABAMA



INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai 600119

www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

**This is to certify that this Project Report is the bonafide work of
PC.JAYAMOHAN(Reg.No:37130165) and S.
HARIHARAN(Reg.No:37130155), who carried out the project entitled “An
image steganography approach based on AES” under our supervision
from October 2020 to March 2021.**

Internal Guide

Dr.V.Balamurugan.,M.Tech.,Ph.D

Associate Professor Dept of ECE

Sathyabama Institute of Science and Technology

Head of the Department

Dr.T.RAVI, M.E.,Ph.D

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

We PC.JAYAMOHAN (Reg.No:37130165) and S.HARIHARAN Reg.No:37130155), Hereby declare that the Project Report entitled (An image steganography approach based on AES)is done by us under the guidance of Dr.V.BALAMURUGAN,.M.Tech.,Ph.D is submitted in partial fulfilment of the requirements for the award of the degree in Electronics and Communication Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our 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. We are grateful to them.

We convey our thanks to Dr. N.M. NANDHITHA, M.E., Ph.D., Dean, School of Electrical and Electronics and Dr. T. RAVI, M.E., Ph.D., Head of the Department, Dept. of Electronics and Communication Engineering for providing me necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide Dr. V. BALAMURUGAN., M.Tech., Ph.D., for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work. It's been an honour working with him. Thanks for everything.

We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of Electronics and Communication Engineering and our parents who were helpful in many ways for the completion of the project.

CONTENTS

CHAPTER	DESCRIPTION	PAGE.No
S.No	ABSTRACT	I
	LIST OF CONTENT	II
1	INTRUDUCTION	1
2	LITRETURE IREVIEW	3
3	3.0 - PROPOSED METHOD	5
3	3.1 - Least Significant Bit	5
3.2	3.2 - Secure Hash Algorithm	8
3.3	3.3 – ADVANCE Lencryphtion STANDARD	9
3.4	3.4 – PHASE OF PROPOSED METHOD	11
3.4.1	3.4.1 – ALGORITHM FOR EMBEDDING DATA INTO IMAGE AND CREATING STEGO	12
3.4.2	3.4.2 – ALGORITHM FOR EXTRACTING DATA FROM STEGO IMAGE	13
3.5	3.5 EXPERIMENTAL RESULTS	15
3.5.1	3.5.1 ORIGINAL AND STEGO IMAGE	16
3.5.2	3.5.2 CODE	18
3.6	3.6 RESULT	20
4	SOFTWARE REQUIREMENTS	21
4.1	4.1 MATLAB	21
4.2	4.2 MATLAB SYSTEMS	23
4.3	4.3 DEVELOPMENT ENVIRONMENT	24
4.4	4.4 MSNIPULATING MATRICES	35

5	FILTERS	40
5.1	NON LOCAL MEANS FILTERING	40
5.2	LOG MSME FILTERING	41
5.3	FILTERING BASED METHODS	42
5.4	Inventory-Style Methods	43
5.5	MEDIAN FILTER PROCESS	45
6	DIGITAL IMAGE PROCESSING	48
6.1	CLASSIFICATION OF IMAGES	53
6.1.1	BINARY IMAGE	53
6.1.2	GRAY SCALE IMAGE	54
6.1.3	COLOUR IMAGE	55
6.2	PREPROCESSING	57
	LIST OF FIGURE	
3.1	3 PIXEL GRID OF 24 BIT PNG	5
3.2	PIXLE GRID AFTER APPLYING LSB TECHNIQUE	5
3.3	FLOW TO GET HASH VALUE	9
3.4	DIAGRAMATIC REPRESENTATION OF AES-256	9
3.5	DETAILED ARCHITECURE OF PROPOSED METHOD	15
3.6	ORIGINAL AND STEGO IMAGE	16
5.5	KERNEL TYPE	45
7	CONCLUTION	63
	REFERENCE	64

An image steganography approach based on AES

Abstract

The proposed paper, works upon the idea of securing the classified information. This is achieved by using steganography which is an approach to hide classified information into some other file while maintaining its visual aids and secondly is cryptography which works upon textual data and transform it in a way that no one can comprehend it. The proposed method secures the weaker section which is the key in Advance Encryption Standard using hashing technique.

The proposed work enhances the level of concealment of information from unauthorized access and for covert information exchange by encrypting the data and hiding it into a multimedia file known as image. The Secure Hash Algorithm 256 generates a hash key of 256 bits which is an unbreakable hashing technique after that the key is used in the process of encrypting the text with Advance Encryption Standard 256 which is an unbreakable encryption technique till this time and a cipher text is obtained.

The cipher text is embedded into a target image using Least Significant Bit method which make changes in image that cannot be understand by naked eyes. The change in byte is 0.000002%. It ensures the visual quality of an image remains intact. The distortion or change in the image remains intermittent to human eyes.

The major issue concerned for the government and security agencies such as were to exchange highly classified information in a secure and undetectable manner and abide the notion of hacker to comprehend any

such information. keywords- AES, Cryptography, Image steganography, LSB, SHA- 256.

INTRODUCTION

Data plays a vital role in merely each vertical or domains. Data provides the next level of insight into different verticals. The data has a versatile nature from personal to economic, governmental, national security, technology and so on. As we all know the data is the new oil so when storing, exchanging, and using data is done then arrives a high responsibility to maintain its confidentiality, integrity, availability of the data.

The higher the confidential information, higher the security is needed. From several years data hiding has captured the minds of security agencies and researchers. Digital watermarking, filtering & masking and other steganographic techniques are used to hide confidential data into digital images.

The cryptography converts the plain text into an incomprehensible cipher text whereas steganography conceals the

traces of the data that could be ever existed. Steganography can be performed on any type of file text, video, audio, image, etc. any multimedia file can fall under this category [1]. When hackers want to access a system, they will aim for the weakest point, which is not the encryption, but the key. AES-256 is most secure encryption technique and till now there is no report of its cracking. But the key is the weakest point in order to secure that SHA-256 (Secure Hash Algorithm) will be used that will give a hash value.

Hash is not „encryption“ because it can never be decrypted back to the initial text, it is a one way street. This makes the system robust and patches the weakest link. In this paper, our findings define that the system first the key will be entered and it will be hashed using Secure Hash Algorithm (SHA256) into a hash data or key which will be used in AES encryption where data get encrypted by the help of the key, by encryption a non-understandable cipher text is generated.

And then this cipher text is embedded into the image using the LSB technique, and the change was into byte is 0.000002% which is an almost negligible and steganographic image or target image is generated containing secret data. And invisible to naked human eyes. Hence, a secured system is created where hashing, encoding as well as data hiding all are performed respectively. This enhances the level of security in the concealment of covert data exchange and information hiding.

LITERATURE REVIEW

The evolution in technology changes the life of mankind. As the technology eases mankind life, it also brings a need for additional security. Steganography is a mechanism of hiding confidential and sensitive information in images to have secure and untraceable communication between multiple individuals. Steganography ensures the exchange of data covertly using any digital communication medium, even the toughest condition of constant monitoring. There is a possibility of doing covert communication and exchange of data without being get noticed and no one can imagine the existence of a secret message inside an image. The embedding is done by weakening some of the traits of other media such as image, audio, video files which are also known as cover.

The Final resultant has the same properties of cover media. The cover contains the crucial secret data in it. When a secret data is concealed inside a cover image, then it results in a stego image. This paper depicts the concept of steganography and proposed methodologies of steganographic images SHA-256 was developed by United States National Security Agency and then got recognized as well as published in the year 2001. SHA-256 is a cryptographic hash also known as the digest. It generates a unique 256-bit (32 bytes) code of a text. A hash is not an encryption technique so, it cannot be decrypted back to original data. It creates a fixed size of code from the source text. It is a single way cryptographic function.

The hash version of the text is always gets compared only, which enhances security. It avoids transmitting password in original view format, even if the message gets intercepted between the medium or channel then also interceptor can never reach to the original password. The NIST (National institute of standard and technology) approves that it has never been compromised [2].

The Advance Encryption Standard (AES) is a cryptographic algorithm which is used to encrypt the textual data in an incomprehensible manner. And it is a most secured encryption standard till now where no signs of its breakage been reported till yet. It is a symmetric key algorithm, which means the same key is used for both encryption and decryption both. It uses a block cipher of sizes 128, 168, 192, 224 and 256 bits. It uses several ways in AES like XOR operation, substitution, permutation operation, rows and column shifting [3].

Based on table 1 and after doing a detailed study, the proposed system technique has been chosen in order to have a robust system which in all aspect can shield the classified information and data within it.

Table 1: Review comparison table for approaches/algorithmic techniques [4-13]

AUTHORS	APPROACHES/ ALGORITHMIC TECHNIQUES	OUTCOME
United States National Security Agency (NSA), 2001	Secure Hash Standard	Hash value is obtained containing 256-bit hash value.
Charles G. Borgelet, Jr., Newark, DE (US); Lisa M. Marvel, Churchville, MD (US); Charles T. Rister, Belcamp, MD (US), 2003	Spread spectrum and image steganography	It generates a spreading Sequence with a pseudorandom noise generator by a key, modulates the encoded text by the Spreading Sequence to obtain an embedded Signal and merging the embedded signal with a cover Signal to generate a StegoSignal.
Po-Yueh Chen* and Hung-Ju Lin, 2006	A DWT based approach for image steganography	Data is embedded in the image with a secure key matrix.
Domenico Bloisi and Luca Jocchi, 2007	Image based steganography and cryptography	A unified approach using ISC as in cryptography and steganography.
Ali Al-Ataby and Fawzi Al-Naima, 2010	A modified high capacity image steganography technique based on wavelet transform	Data is hidden in the image with wavelet transformation in message as well as image.
Shailender Gupta, Ankur Goyal, Bharat Bhushan, 2012	Information hiding using least significant bit steganography and cryptography	RSA and Diffie Hellman algorithm has been used as cryptography technique to generate cipher text and LSB is used to embed it into image.
Saiful Islam*, Mangat R Modi and Phalguni Gupta, 2014	Edge-based image steganography	Hiding of data into the image edges is achieved.
*Khan Muhammad, Jamil Ahmad, Haleem Farman, Muhammad Zubair, 2015	A novel image steganographic approach for hiding text in color images using HSI color model	It contains larger Peak Signal to Noise Ratio (PSNR) values.
Khan Muhammad, Jamil Ahmad, Muhammad Sajjad, Muhammad Zubair, 2015	Secure image steganography using cryptography and image transposition	Multiple encryption (bitxor operation, bits shuffling, and stego key-based encryption) and Image transposition is done for data hiding.
Muyaffaq Abu-Alheija, 2019	Crypto-Steganographic LSB-based System for AES-Encrypted Data	Stego image is obtained with cipher text by AES-128-bit encryption.

PROPOSED METHOD

The Proposed methodology is to create a system which could be used for covert communication without any hint that some classified data is being exchanged via using image steganography and securing that data with cryptographic technique and even securing its vulnerable area that is key by using hashing.

A.Least Significant Bit (LSB)

The steganography technique embeds the data into the cover image or target image. This can be achieved by using various steganographic techniques. From a pool of steganographic techniques I will be accomplishing this task with the usage of LSB. This technique is implemented by making changes in the least significant bit (LSB) of a byte. LSB is a most promising steganographic technique which conceals the data in the image and no one can get a whisper that the image accommodates some crucial data within it. LSB provides the least change in the image which is far from persistent of a human naked eye.

The change in a byte is 0.000002% which is almost negligible. The least significant bit of the cover image bytes is changed by a bit of the classified data. LSB functions finest with 24- bit map (BMP) and PNG images in which each pixel is constituted of 3 bytes (red, green, blue) color respectively.

LSB provides a method to store 3 bits of information in a single pixel, by making a change in one bit of each color component.

Therefore, by using a 256*256 image, LSB can store 196608 bits (i.e.

24576 bytes) of data inside this cover image.

For example: if we want to embed “A” in a image. ASCII code of “A” is 65. The binary value of “A” is 01000001.

Table 2: 3 Pixel Grid of 24 BIT PNG

RED	BLUE	GREEN
10000000	10100100	10110101
10110101	11110011	10110111
11100111	10110011	00110011

Table 3: 3 Pixel Grid after applying LSB technique

RED	BLUE	GREEN
10000000	10100101	10110100
10110100	11110010	10110110
11100110	10110010	00110011

The table 2 contains the pixel grid of a image where each pixel flourishes some colors. Whereas In table 3 it can be clearly seen that the pixel grid of the cover image has been changed in order to embed data into it.

And the change in each pixel is so nominal that it cannot be traced. LSB makes use of lossless compression provided by PNG images. consecutive bytes of the image data are used to embed the information, initially from the first image pixel of each byte of the message in respective order. Consequently, LSB is the best way to hide data and communicate covertly but if any hacker thinks that the image contains confidential data, and he will try to extract the crucial data from it by use of steganalysis tools and if he is accomplishing his goal then our crucial data is in wrong hands. It orders to achieve a higher level of security this proposed system is created which add on advance level of security. This add on intricate multiple layers of imperishable security. The image constitutes of pixels that do not engage changes in the anatomy of the message in order to accomplish secure and imperceptible communication.

A steganographic technique could embed an encrypted data in the cover image. The proposed method acquires LSB technique in order to replace the least significant bits in the cover image which contains bits of the hidden message.

In succession to enhance the security level of the proposed system, the covert data should be encrypted using AES encryption using a key

which is generated using the SHA-256 hashing technique. The proposed system ensure the difference between an original image and a cover image which remains imperceptible to human senses, meanwhile sustaining the integrity of the confidential data without any loss.

B. Secure Hash Algorithm (SHA-256)

The Secure Hash Algorithm (SHA-256) is a part of family. The family of cryptographic hash functions which was published by the National Institute of Standards and Technology as a U.S. Federal Information Processing Standard. SHA-256 is patented in US patents 6829355. Hash is not „encryption“ because it can never be decrypted back to the initial text.

It is of fixed length for any varying length of the source text. SHA-256 is achieved by the following steps like padding, append length, divide the input into 512bit blocks, add chaining variable, process block respectively. SHA-256 generates a unique 256-bit or we can say 32-byte of hash code. It does 64 rounds in order to reach to the final hash value.

The text length upto 264 bit (2.3 exabyte) will also be transformed into a digest size of 256 bits (32 bytes). 16.80 cycles per byte on 64-bit processor x86 architecture. It has 64 iterations in one cycle. Which makes it more safe and secure.

For example: Hash value of “abc” is **“ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad”**.

It is of 256 bits (32 bytes). The hash value is obtained by the respective hashing operations mentioned above which are performed in SHA-256.



Figure1: Diagram depicts the flow to get hash value.

C. Advance Encryption Standard (AES-256)

AES is a symmetric key encryption technique, which use only one secret key to cipher(encrypt) and decipher(decrypt) data. Advanced Encryption Standard (AES) is only publicly accessible encryption technique authorized by the US National Security Agency (NSA) for securing top classified secret data. AES was earlier known as Rijndael after its two amazing developers known, Belgian cryptographers Vincent Rijmen and Joan Daemen. AES-256, uses a key length of 256 bits, it uses the largest bit length and is practically adamant by brute force attack on the basis of current computing power, which makes it the strongest encryption standard.

The AES encryption method first performs substitution of data using a substitution table; which is followed by shifting of data rows, then after that mixing of the columns takes place, and finally the last transformation with a simple exclusive (XOR) operation performed on each column using a different part of the encryption key The following table shows that possible key combinations exponentially increase with the key length. And it also makes it hard to crack it

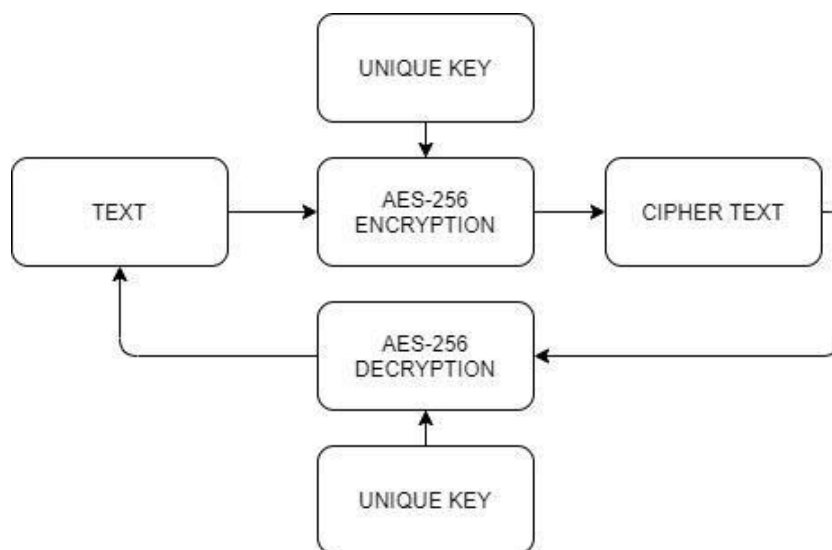


Figure 2: The Diagrammatic representation of AES-256

Table 4: key length with its possible combinations

Key Length	Possible Combinations
1 bit	2
2 bits	4
4 bits	16
8 bits	256
16 bits	65536
32 bits	4.2×10^9
56 bits	7.2×10^{16}
64 bits	1.8×10^{19}
128 bits	3.4×10^{38}
192 bits	6.2×10^{57}
256 bits	1.1×10^{77}

The length of the key is of 256 bit (32 byte) so, 14 rounds of AES encryption have been performed and as a result an incomprehensible cipher text is generated. Same key is used in decryption process also and as a result the text is obtained by running 14 rounds of decryption. All the process executes in a reverse order in a way to get the plain text.

PHASES OF THE PROPOSED METHOD

The first phase involves key generation which is done by using SHA-256 a hashing technique. Hash is not „encryption” because it can never be decrypted back to the initial text. It is of fixed length for any varying length of the source text. SHA-256 is achieved by the following steps like padding, append length, divide the input into 512bit blocks, add chaining variable, process block respectively. SHA-256 generates a unique 256-bit or we can say 32byte of hash code. It does 64 rounds in order to reach its final hash value. So, it is impermeable and till now there is no reporting of its breakage. By this, we generate the most secure key. The second phase is converting the Plain text data into cipher text with the help of key. The AES encryption performs several transformations on data.

The initial step is to put the data into an array, after

which the cipher transformations are repeated over a number of encryption rounds. The number of time the encryption process with take place one after the other or the number of rounds depends upon the key length, 10 rounds mean a 128-bit keys, 12 rounds mean 192bit keys and 14 rounds mean 256- bit keys.

For the purpose of the proposed method, 14 rounds of transformation were endorsed resulting in a 256-bit key. As the HASH-256 generates a hash value of 256 bit which will be used as a key in AES system. The AES encryption method first performs substitution of data using a substitution table; which is followed by shifting of data rows, then after that mixing of the columns takes place, and finally the last transformation with a simple exclusive (XOR) operation performed on each column using a different part of the encryption key.

As AES is a symmetric data encryption technique or method, the same key is being used for both the operations encryption as well as decryption. As a result, cipher text is obtained which is incomprehensible. There has not been any report of breakage of AES encryption till now. By which it adds on a one more secure layer over the system. The final phase is embedding the cipher text into the image in order that there should be the least distortion in the image that could not be persistent to human naked eyes.

Each pixel depicts a color, mainly it comprises of three primary colors: Red, Green, Blue. Red, Green and Blue intensities can vary from 0 to 255. The WHITE = (255,255,255) has the highest value and BLACK = (0,0,0) has the lowest value. A pixel uses 3 bytes of memory space, one for each component that is why the maximum value is 255. A byte consists of 8 bits, and it is represented in a binary number (example: 1010 0101). The max. value a byte can have is 1111 1111, which is 255 in decimal number. The cipher text gets embedded into the cover image using a technique known as least significant bit.

Least significant bit changes only change a single bit of each color, respectively. It means that a pixel is comprised of three colors red, blue, green, each pixel consists of 3 bytes of information of pixel. And

in a single byte of a single color we only change the last bit of the color. It summarizes that 1 pixel contains 3 bits of data only, which signifies that only three characters. So, the noise in the image is least. And no change in original and cover image can be seen. For example, the color intensity of red is 255 and we change the last bit its value becomes 254, there is not a major change in the intensity of the color, thus it can't be detected by the human eye.

A.Algorithm For Embedding Data Into Image And Creating Stego-Image.

Step 1 Start

Enter the text (T) of any length.

Step 2

// User should enter classified information.

Enter the key (k) of any length.

Step 3 //The user needs to enter a password to protect its data via encryption and decryption is done using it

Enter the Cover image (I) with extension and path.

Step 4 //Enter the path of image file with its name which will be hiding the classification information in it.

The key (k) undergoes SHA-256 hashing technique and a

Step 5 hash value (H) is obtained.

// A secure key is obtained which secures weaker section.

Hash value (H) is used as a key in AES-256 encryption

Step 6 and plain text (T) is converted into cipher text (CT) by using AES-256 encryption algorithm

Cipher text is embedded into the image (I) using LSB a steganographic technique and a stego-image (ST) is

Step 7

obtained. // Stego-image can be shared now using any communication channel.

Step 8 End

B.Algorithm For Extracting Data From Stego-Image

Step 1 Start

Enter the Stego image (ST) with extension and path.

Step 2

//Enter the path of image file with its name which is having classified information in it.

Steganalysis on the Stego image (ST) is performed and cipher text (CT) is obtained.

Step 3

//Steganalysis is a process to recover data hidden data from the image

Enter the key (K) thete undergoes SHA-256 hashing techniques and a hash value (H) is obtained.

Step 4

//A secure key is obtained by which data can be only retrieved.

Hash value (H) is used as a key in AES-256 decryption and cipher text (CT) is converted into plain text (T) by using AES-256 decryption algorithm. //Original classified information is obtained.

Step 5

Step 6 End

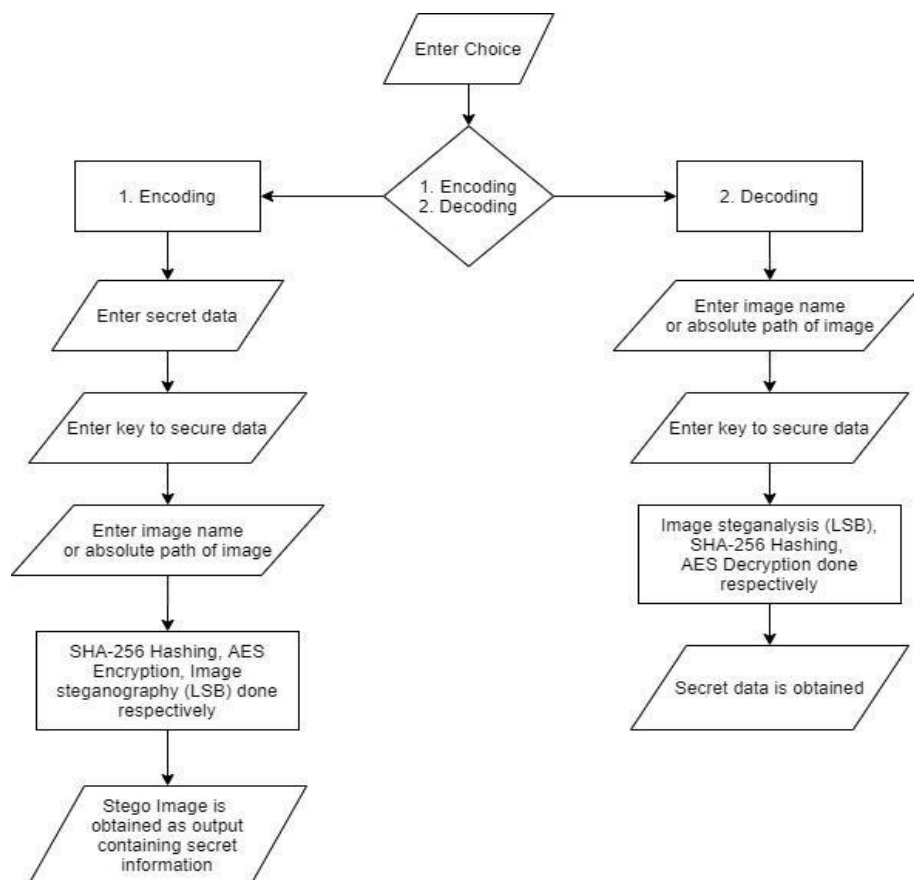


Fig 3. Flowchart of proposed system (Interface Diagram)

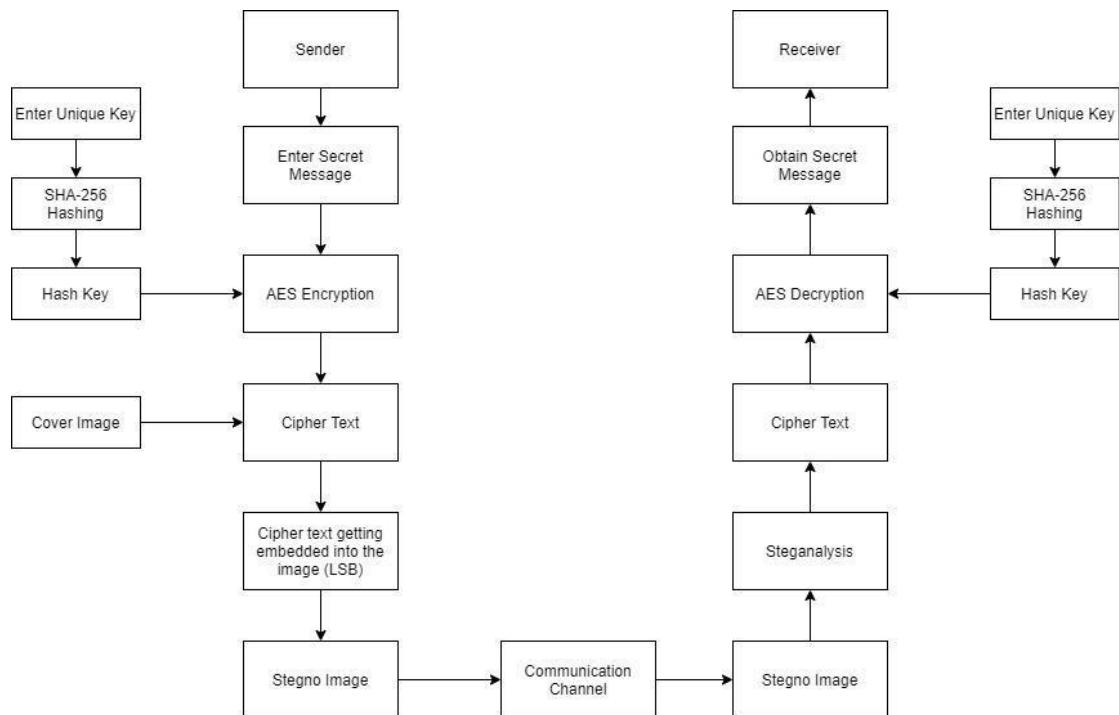


Fig 4. Detailed Architecture of Proposed Method

EXPERIMENTAL RESULT

The proposed method has been implemented in order to invent a secure and impermeable image system. And to do an analysis of the result in order to find out whether the proposed system is working accordingly as per design in consideration of requirements or not.

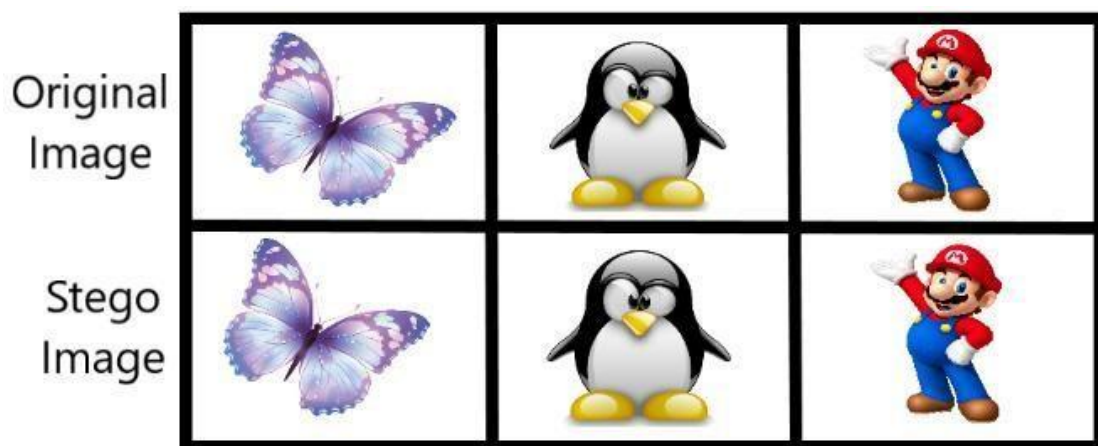


Table 5: Original and Stegno images

The above table 5 depicts that no change can be seen or persistent to human naked eyes and we cannot find out the difference between both the images i.e. original and stego image. So, by this confidentiality and covertness of the crucial data remains stagnant. Now, if we move forward to table 6 it depicts that no change in the dimension of the image is seen no matter how much amount of data been input into the image.

Apparently, the data should always be less than the capacity of the image to hold it. The major difference is that if the data i.e. embedded in the original image comprises less than 1 KB then we cannot see the difference in the size of the image. Otherwise, the size of the image gets slightly increase as per the size of the data.

I have used python (Programming language) in order to implement the algorithm and to create a data securing system. And the libraries which I have used are:-

1.pycryptodome in order to perform AES (Advance Encryption Standard) encryption and decryption.

2. Stegano in order to implement image steganography via LSB

(Least Significant Bit).

3.Hashlib in order to perform hashing via SHA-256 (Secure Hashing Algorithm).

Table 6: Result after performing the experiment

Image Name	Original Image size	Original Image Dimension	Size of the Data entered	Stego Image Size	Stego Image Dimension
Butterfly	176 kB	386 * 395	350 B	176 KB	512 * 486
Penguin	47.1 kB	386 * 395	2 kB	49.5 KB	386 * 395
Mario	22.1 kB	219 * 150	3 KB	25 KB	219 * 150

The screenshot of the Proposed working system

1.Encoding

```
1. Encode
2. Decode
}
Enter Data : My Name is yash kumar shukla, i am a student of MCA from JSS ACADEMY OF TECHNICAL EDUCATION
Enter Key : password
Enter absolute Path with file name & extension : D:\Project\imagestegnography\image\112.png
Enter absolute Path with file name & extension for new image : D:\Project\imagestegnography\image\new.png
Cipher text is : 0u1Y+Jp7RFGl+Hae2D57oIrCW1Sgi5BVGqT7TkQDhWgtQkUW6QVWwI95WlxRym+g8Bw0Cjrcf+q1Fq/dsdikkysH2dGHORi6f0j0gdFt8DDz5lw2741g6Wit0JoSa/wln71y3zyl.vvq0L.SuZScGA==
```

2.Decoding

```
D:\Project\imagestegnography\venv\Scripts\python.exe D:/Project/imagestegnography/Project.py
:: Welcome to Steganography ::
1. Encode
2. Decode
2
Enter absolute Path with file name & extension :D:\Project\imagestegnography\image\new.png
Enter Key : password
data recovered is : My Name is yash kumar shukla, i am a student of MCA from JSS ACADEMY OF TECHNICAL EDUCATION
```

3.Wrong Password

```
D:\Project\imagestegnography\venv\Scripts\python.exe D:/Project/imagestegnography/Project.py
:: Welcome to Steganography ::
1. Encode
2. Decode
2
Enter absolute Path with file name & extension :D:\Project\imagestegnography\image\new.png
Enter Key : pass
data recovered is : None
```

The above screenshot depicts that the working of the proposed system is efficient and accomplish the task and functionally robust. If the key is wrong the data cannot be extracted from it, even if the 1 st layer of protection is been breached by any technique and the hacker identifies the image contains data. Then also the data is armored with other layer of protection and it cannot extract data from it as it is still being protected from cryptographic technique. The proposed system is robust and if a hacker tries to extract the data from the image, it is shielded with hashing, encryption, and image steganography.

In which each technique is impermeable and combination of all the three make it extremely secure. Hence, the proposed method manifest to provide robustly and secure image steganography as well as a secure and inoculative system against data hacking.

CODE

1)Key Creation and Hide Method

```

def keyCreation(key):
    block_size = 32
    # Getting a hash value of 256 bit (32 byte)
    key = hashlib.sha256(key.encode()).digest()
    return key

def hide(input_filename, output_filename, data, key):

    block_size = 32
    # Get a random initialization vector
    iv = Random.new().read(AES.block_size)
    # using Cipher Block Chaining (CBC) Mode
    encryption_suite = AES.new(key, AES.MODE_CBC, iv)

    # If it is string convert to byte string before use it
    if isinstance(data, str):
        data = data.encode()

    # Encrypt the random initialize vector added with the padded data
    cipher_data = encryption_suite.encrypt(iv + pad(data, block_size))

    # Convert the cipher byte string to a base64 string to avoid decode padding error
    cipher_data = base64.b64encode(cipher_data).decode()
    print("Cipher text is :", cipher_data)

    # Hide the encrypted data in the image via LSB technique.
    secret = lsb.hide(input_filename, cipher_data)
    secret.save(output_filename)

```

2) Retrieve Method

```

def retrieve(input_image_file, key):

    block_size = 32
    cipher_data = lsb.reveal(input_image_file)

    if not cipher_data:
        return None

    cipher_data = base64.b64decode(cipher_data)
    # Retrieve the dynamic initialization vector saved
    iv = cipher_data[:AES.block_size]
    # Retrieved the cipher data
    cipher_data = cipher_data[AES.block_size:]

    try:
        decryption_suite = AES.new(key, AES.MODE_CBC, iv)
        decrypted_data = unpad(
            decryption_suite.decrypt(cipher_data),
            block_size
        )
        try:
            return decrypted_data.decode('utf-8')
        except UnicodeDecodeError:
            # Binary data - returns as it is
            return decrypted_data
    except ValueError:
        return None

```

RESULT ANALYSIS

By the combination of all three methods we created a robust system. The data gets encrypted by most secure cryptographic technique and then this data gets hidden inside an image which leaves no mark of any classified information communication taking place.

As the visual aids remains intact. Its results can be seen in table 5 & table 6. And the weak point that a hacker targets is also been procured which is a key by hashing technique and the utilization of that key is done in encrypting and decrypting the data. Which adds on the layer of protection to the classified data. So, our goal gets achieved firstly no sign of secret communication and for safety adding more security layers on it. The data as encrypted and embedded in the image at the time of retrieval it gives the same data while using authentic secure key.

SOFTWARE REQUIREMENTS

SOFTWARE REQUIREMENTS:

- MATLAB 2018b

MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or FORTRAN.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB uses software developed by the LAPACK and ARPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

The MATLAB System

The MATLAB system consists of five main parts:

Development Environment. This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a

command history, and browsers for viewing help, the workspace, files, and the search path.

The MATLAB Mathematical Function Library. This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

The MATLAB Language. This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

Handle Graphics®. This is the MATLAB graphics system. It includes high-level commands for two-dimensional and threedimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

The MATLAB Application Program Interface (API). This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

DEVELOPMENT ENVIRONMENT

Introduction

This chapter provides a brief introduction to starting and quitting MATLAB, and the tools and functions that help you to work with MATLAB variables and files. For more information about the topics covered here, see the corresponding topics under Development Environment

in the MATLAB documentation, which is available online as well as in print.

Starting and Quitting MATLAB

Starting MATLAB

On a Microsoft Windows platform, to start MATLAB, double-click the MATLAB shortcut icon on your Windows desktop.

On a UNIX platform, to start MATLAB, type `matlab` at the operating system prompt.

After starting MATLAB, the MATLAB desktop opens - see MATLAB Desktop.

You can change the directory in which MATLAB starts, define startup options including running a script upon startup, and reduce startup time in some situations.

Quitting MATLAB

To end your MATLAB session, select Exit MATLAB from the File menu in the desktop, or type `quit` in the Command Window. To execute specified functions each time MATLAB quits, such as saving the workspace, you can create and run a `finish.m` script.

MATLAB Desktop

When you start MATLAB, the MATLAB desktop appears, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB.

The first time MATLAB starts, the desktop appears as shown in the following illustration, although your Launch Pad may contain different entries.

You can change the way your desktop looks by opening, closing, moving, and resizing the tools in it. You can also move tools outside of the desktop or return them back inside the desktop (docking). All the desktop tools provide common features such as context menus and keyboard shortcuts.

You can specify certain characteristics for the desktop tools by selecting Preferences from the File menu. For example, you can specify the font characteristics for Command Window text. For more information, click the Help button in the Preferences dialog box.

Desktop Tools

This section provides an introduction to MATLAB's desktop tools. You can also use MATLAB functions to perform most of the features found in the desktop tools. The tools are:

Command Window

Use the Command Window to enter variables and run functions and M-files.

Command History

Lines you enter in the Command Window are logged in the Command History window. In the Command History, you can view previously used functions, and copy and execute selected lines. To save the input and output from a MATLAB session to a file, use the diary function.

Running External Programs

You can run external programs from the MATLAB Command Window. The exclamation point character! is a shell escape and indicates that the rest of the input line is a command to the operating system. This is useful for invoking utilities or running other programs without quitting MATLAB. On Linux, for example,!emacs magik.m invokes an editor

called emacs for a file named magik.m. When you quit the external program, the operating system returns control to MATLAB.

Launch Pad

MATLAB's Launch Pad provides easy access to tools, demos, and documentation.

Help Browser

Use the Help browser to search and view documentation for all your Math Works products. The Help browser is a Web browser integrated into the MATLAB desktop that displays HTML documents.

To open the Help browser, click the help button in the toolbar, or type help browser in the Command Window. The Help browser consists of two panes, the Help Navigator, which you use to find information, and the display pane, where you view the information.

Help Navigator

Use the Help Navigator to find information. It includes:

Product filter - Set the filter to show documentation only for the products you specify.

Contents tab - View the titles and tables of contents of documentation for your products.

Index tab - Find specific index entries (selected keywords) in the MathWorks documentation for your products.

Search tab - Look for a specific phrase in the documentation. To get help for a specific function, set the Search type to Function Name.

Favorites tab - View a list of documents you previously designated as favorites.

Display Pane

After finding documentation using the Help Navigator, view it in the display pane. While viewing the documentation, you can:

Browse to other pages - Use the arrows at the tops and bottoms of the pages, or use the back and forward buttons in the toolbar.

Bookmark pages - Click the Add to Favorites button in the toolbar.

Print pages - Click the print button in the toolbar.

Find a term in the page - Type a term in the Find in page field in the toolbar and click Go.

Other features available in the display pane are: copying information, evaluating a selection, and viewing Web pages.

Current Directory Browser

MATLAB file operations use the current directory and the search path as reference points. Any file you want to run must either be in the current directory or on the search path.

Search Path

To determine how to execute functions you call, MATLAB uses a search path to find M-files and other MATLAB-related files, which are organized in directories on your file system. Any file you want to run in MATLAB must reside in the current directory or in a directory that is on the search path. By default, the files supplied with MATLAB and MathWorks toolboxes are included in the search path.

Workspace Browser

The MATLAB workspace consists of the set of variables (named arrays) built up during a MATLAB session and stored in

memory. You add variables to the workspace by using functions, running M-files, and loading saved workspaces.

To view the workspace and information about each variable, use the Workspace browser, or use the functions `who` and `whos`.

To delete variables from the workspace, select the variable and select Delete from the Edit menu. Alternatively, use the `clear` function.

Array Editor

Double-click on a variable in the Workspace browser to see it in the Array Editor. Use the Array Editor to view and edit a visual representation of one- or two-dimensional numeric arrays, strings, and cell arrays of strings that are in the workspace.

Editor/Debugger

Use the Editor/Debugger to create and debug M-files, which are programs you write to run MATLAB functions. The Editor/Debugger provides a graphical user interface for basic text editing, as well as for M-file debugging.

You can use any text editor to create M-files, such as Emacs, and can use preferences (accessible from the desktop File menu) to specify that editor as the default. If you use another editor, you can still use the MATLAB Editor/Debugger for debugging, or you can use debugging functions, such as `dbstop`, which sets a breakpoint.

MANIPULATING MATRICES

Entering Matrices

The best way for you to get started with MATLAB is to learn how to handle matrices. Start MATLAB and follow along with each example.

You can enter matrices into MATLAB in several different ways:

- Enter an explicit list of elements.
- Load matrices from external data files.
- Generate matrices using built-in functions.
- Create matrices with your own functions in M-files.

Start by entering Dürer's matrix as a list of its elements. You have only to follow a few basic conventions:

- Separate the elements of a row with blanks or commas.
- Use a semicolon, ; , to indicate the end of each row.
- Surround the entire list of elements with square brackets, [].

To enter Dürer's matrix, simply type in the Command Window

```
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

MATLAB displays the matrix you just entered.

A =

```
16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

This exactly matches the numbers in the engraving. Once you have entered the matrix, it is automatically remembered in the MATLAB workspace. You can refer to it simply as A.

Expressions

Like most other programming languages, MATLAB provides mathematical expressions, but unlike most programming languages, these expressions involve entire matrices. The building blocks of expressions are:

Variables

MATLAB does not require any type declarations or dimension statements. When MATLAB encounters a new variable name, it automatically creates the variable and allocates the appropriate amount of storage. If the variable already exists, MATLAB changes its contents and, if necessary, allocates new storage. For example, `num_students = 25`

Creates a 1-by-1 matrix named `num_students` and stores the value 25 in its single element.

Variable names consist of a letter, followed by any number of letters, digits, or underscores. MATLAB uses only the first 31 characters of a variable name. MATLAB is case sensitive; it distinguishes between uppercase and lowercase letters. A and a are not the same variable. To view the matrix assigned to any variable, simply enter the variable name.

Functions

MATLAB provides a large number of standard elementary mathematical functions, including `abs`, `sqrt`, `exp`, and `sin`. Taking the square root or logarithm of a negative number is not an error; the appropriate complex result is produced automatically. MATLAB also provides

many more advanced mathematical functions, including Bessel and gamma functions. Most of these functions accept complex arguments. For a list of the elementary mathematical functions, type `helpelfun`

For a list of more advanced mathematical and matrix functions,

Some of the functions, like `sqrt` and `sin`, are built-in. They are part of the MATLAB core so they are very efficient, but the computational details are not readily accessible. Other functions, like `gamma` and `sinh`, are implemented in M-files. You can see the code and even modify it if you want. Several special functions provide values of useful constants.

GUI

A graphical user interface (GUI) is a user interface built with graphical objects, such as buttons, text fields, sliders, and menus. In general, these objects already have meanings to most computer users. For example, when you move a slider, value changes; when you press an OK button, your settings are applied and the dialog box is dismissed. Of course, to leverage this built-in familiarity, you must be consistent in how you use the various GUIbuilding components.

Applications that provide GUIs are generally easier to learn and use since the person using the application does not need to know what commands are available or how they work. The action that results from a particular user action can be made clear by the design of the interface.

The sections that follow describe how to create GUIs with MATLAB. This includes laying out the components, programming them to do specific things in response to user actions, and saving and launching the GUI; in other words, the mechanics of creating GUIs. This documentation does not attempt to cover the "art" of good user interface design, which is an entire field unto itself. Topics covered in this section include:

Creating GUIs with GUIDE

MATLAB implements GUIs as figure windows containing various styles of GUI control objects. You must program each object to perform the intended action when activated by the user of the GUI. In addition, you must be able to save and launch your GUI. All of these tasks are simplified by GUIDE, MATLAB's graphical user interface development environment.

GUI Development Environment

The process of implementing a GUI involves two basic tasks:

- Laying out the GUI components
- Programming the GUI components

GUIDE primarily is a set of layout tools. However, GUIDE also generates an M-file that contains code to handle the initialization and launching of the GUI. This M-file provides a framework for the implementation of the call backs - the functions that execute when users activate components in the GUI.

Features of the GUIDE-Generated Application M-File

GUIDE simplifies the creation of GUI applications by automatically generating an M-file framework directly from your layout. You can then use this framework to code your application Mfile. This approach provides a number of advantages:

The M-file contains code to implement a number of useful features (see *Configuring Application Options* for information on these features). The M-file adopts an effective approach to managing object handles and executing call back routines (see *Creating and Storing the Object Handle Structure* for more information). The Mfiles provides a way to manage global data (see *Managing GUI Data* for more information).

Beginning the Implementation Process

To begin implementing your GUI, proceed to the following sections:

Getting Started with GUIDE - the basics of using GUIDE

Selecting GUIDE Application Options - set both FIG-file and Mfile options.

Using the Layout Editor - begin laying out the GUI.

Understanding the Application M-File - discussion of programming techniques used in the application M-file.

Application Examples - a collection of examples that illustrate techniques Which are useful for implementing GUIs.

Command-Line Accessibility

When MATLAB creates a graph, the figure and axes are included in the list of children of their respective parents and their handles are available through commands such as find object, set, and get. If you issue another plotting command, the output is directed to the current figure and axes.

GUIs are also created in figure windows. Generally, you do not want GUI figures to be available as targets for graphics output, since issuing a plotting command could direct the output to the GUI figure, resulting in the graph appearing in the middle of the GUI.

In contrast, if you create a GUI that contains axes and you want commands entered in the command window to display in this axes, you should enable command-line access.

User Interface Controls

The Layout Editor Component palette contains the user interface controls that you can use in your GUI. These components

are MATLAB uicontrol objects and are programmable via their Call back properties. This section provides information on these components.

Push Buttons

Push buttons generate an action when pressed (e.g., an OK button may close a dialog box and apply settings). When you click down on a push button, it appears depressed; when you release the mouse, the button's appearance returns to its non depressed state; and its call back executes on the button up event.

Properties to Set

String - set this property to the character string you want displayed on the push button.

Tag - GUIDE uses the Tag property to name the call-back sub function in the application M-file. Set Tag to a descriptive name (e.g., close button) before activating the GUI.

Programming the Call-back

When the user clicks on the push button, its call back executes. Push buttons do not return a value or maintain a state

Toggle Buttons

Toggle buttons generate an action and indicate a binary state (e.g., on or off). When you click on a toggle button, it appears depressed and remains depressed when you release the mouse button, at which point the call-back executes. A subsequent mouse click returns the toggle button to the non depressed state and again executes its call back.

Programming the Call back

The call back routine needs to query the toggle button to determine what state it is in. MATLAB sets the Value property equal to the Max property when the toggle button is depressed (Max is 1

by default) and equal to the Min property when the toggle button is not depressed (Min is 0 by default).

Radio Buttons

Radio buttons are similar to checkboxes, but are intended to be mutually exclusive within a group of related radio buttons (i.e., only one button is in a selected state at any given time). To activate a radio button, click the mouse button on the object. The display indicates the state of the button.

Checkboxes

Check boxes generate an action when clicked and indicate their state as checked or not checked. Check boxes are useful when providing the user with a number of independent choices that set a mode (e.g., display a toolbar or generate call back function prototypes).

Edit Text

Edit text controls are fields that enable users to enter or modify text strings. Use edit text when you want text as input. The String property contains the text entered by the user.

Axes

Axes enable your GUI to display graphics (e.g., graphs and images). Like all graphics objects, axes have properties that you can set to control many aspects of its behaviour and appearance. See Axes Properties for general information on axes objects.

Figure

Figures are the windows that contain the GUI you design with the Layout Editor. See the description of figure properties for information on what figure characteristics you can control.

FILTERS

Non Local means filtering

Non-local means filter is an algorithm in signal processing for denoising. Unlike other local smoothing filters, non-local means filter averages all observed samples to recover a single sample. The weight of each pixel depends on the distance between its intensity grey level vector and that of the target signal sample. The NLM filter is based on the assumption that image content is likely to repeat itself within some neighbourhood (in the image) and in neighbouring frames. It computes denoised sample $x(p, q)$ by the weighted sum of the surrounding pixels of $Y(p, q)$ (within frame and in the neighbouring frames). This feature provides a way to estimate the signal sample value from noise contaminated signals. In a NLM algorithm, the estimate of a pixel at position (p, q) is

Where m is the frame index, and N represents the neighbourhood of the sample at location (p, q) . K values are the filter weights, i.e.,

Where V is the window [usually a square window centred at the pixels $Y(p, q)$ and $Y(r, s)$] of sample values from a geometric neighbourhood of samples $Y(p, q)$ and $Y(r, s)$, σ is the filter coefficient, and $f(\cdot)$ is a geometric distance function. K is inversely proportional to the distance between $Y(p, q)$ and $Y(r, s)$. where Z is the blurred version of the targeted signal, and O_m is the weight matrix,

Log-MMSE Filtering:

The problem is discussed in more generality than in many other expositions specifically we allow for general filter delays (to

accommodate the pitch filtering problem, for instance) and cover both the stochastic case and block-based analyses with a single formalism. For mean-square error computations, we will only need to use at most second order statistical properties (correlations and means). For the case of stochastic signals, these notes look at the derivation of the correlation values required for a minimum mean-square error solution. We also examine systems which involve cyclo stationary signals (interpolation filter, for instance).

The important linear prediction problem is examined in detail. This includes the setup for non-equally spaced delay values. For the equally spaced delay case, we can develop a rich set of results. For the least-squares problem, these notes give a generalized view of windowing: windowing the data and/or windowing the error. This view subsumes the traditional special cases, viz the auto correlation and covariance methods. These notes present a number of examples based on “real” signals. With the background developed, the results are obtained with relatively straightforward MATLAB scripts. The results illustrate the useful insights that can be obtained when minimum mean-square error theory is appropriately fleshed out.

Consider a filter with an input $x[n]$ and an output $y[n]$ given by

Where the W_k values¹ weight the samples of the input signal at different delays D_k . We require that the delays be distinct.

A conventional causal FIR filter would simply have $D_k = k$ for $k = 0, M - 1$. We will keep the delays general until later in this document, when it becomes useful to specialize them to get further results. The goal is to find a set of filter coefficients that minimize the squared error between the output of the filter $y[n]$ and a desired signal $d[n]$.

First we write the filtering operation in vector form,

Where,

To be able to handle both the stochastic case and block-based squared-error cases with a single formulation, we define an “averaging” operation with a bar over an expression. For the case of ensemble averaging, this is the expectation operator. For other cases, it will signify a sum of squares. In many of the cases we study, the averaging operation will remove the dependency on n . For the case of wide-sense stationary processes, results are reviewed in Appendix B.

The error is,

The block diagram of the is considered as,

5.3 Filtering-Based Methods

Adaptive filtering and adaptive spectral subtraction are the most widely used methods for single-channel speech enhancement in use today. We refer to both methods collectively as filtering-based methods. The adaptation of the respective filter parameters of these methods is typically performed with the help of two essential system components: a noise spectrum estimator, such as the improved minima-controlled recursive averaging (IMCRA) algorithm proposed by Cohen or the minimum statistics algorithm by Martin and an a-priori signal-to-noise ratio (SNR) estimator, such as the decision-directed approach by Ephraim and Malah. Filter gains are determined through short time spectral amplitude (STSA) estimators or the log spectral amplitude estimator (log-MMSE) or other suitable mappings. In filtering-based techniques, the quality of the resulting enhanced signal depends greatly on the quality of the corresponding SNR and STSA estimate. Improvements of adaptive SNR and STSA estimators have, therefore, received a significant amount of attention in the research community.

5.4 Inventory-Style Methods

A key problem in inventory-style enhancement is the reliable recognition of the underlying phonetic class of a noisy speech segment. In standard approaches to automatic speech recognition or phoneme classification, it is generally assumed that preprocessing delivers a good estimate of the clean speech signal. Under severely distorted conditions, however, any estimate of the clean speech will contain residual noise and possibly additional distortions due to the applied signal processing. Therefore, Instead of using just a point estimate of the clean speech signal for phoneme recognition, it is valuable to consider clean speech as an underlying unknown process that can only be estimated with a residual, time-varying error variance. A good estimate of this variance can lead to significant improvements in recognition performance, since the recognizer can then focus on more reliable segments and/or features. These variances may be obtained directly in the feature domain, or respective variances can be estimated in the spectral domain and then propagated into a cepstral feature domain with techniques. The feature variances can then be used to improve recognition robustness via uncertainty-based speech decoding techniques. Such techniques may use the uncertainty for adaptively increasing the model variances, termed variance compensation, for obtaining an additional HMM based estimate of the clean features during decoding, so-called modified imputation, or for HMM-based feature estimation in conjunction with variance adaptation.

A conceptual block diagram of the entire enhancement system First, the incoming signal is segmented. The resulting segment stream is passed to three subsystems:

- a conventional log-spectral minimum mean square error (log-MMSE) filter after,
- a voice activity detection (VAD) mechanism after Sohn and Kim
- The proposed inventory search procedure after Section VI.

The employed log-MMSE filter uses a decision-directed approach after estimate the underlying a priori SNR and a VAD-supported time-recursive averaging method after to estimate the underlying power of the noise. The VAD block provides three important functions: it activates the inventory search part of the algorithm only during times in which a speech signal is assumed to be present in the input, it provides the log-MMSE filter with information about when to update its internal estimate of the noise power spectral density (PSD), and it is used to control the stream selection and normalization procedure that fuses the output frames of the log-MMSE filter with the output frames of the inventory search procedure. The VAD selects stream during times of voice activity and stream otherwise.

The division of the processing into separate streams and increases the robustness of the proposed method during times in which the VAD is inaccurate and/or unreliable, particularly during transition times between voice active and silence states. The output frames of the stream selection and normalization block are subjected to a cepstral smoothing procedure and are then reconcatenated via cross-fading into the enhanced output signal.

The implementation details for the proposed inventory search are comprehensively described in the following subsections. The essential goal of the inventory search is to replace the incoming frames with suitable waveforms from the inventory. We have divided our description into three sections: the robust feature extraction part that is required for both training and enhancement, the system training part that organizes the inventory and learns its underlying statistical properties, and (Section VI) the actual speech enhancement procedure as outlined in the block diagram.

5.5 The MedianFilter Process

The median of a distribution is the value for which larger and smaller values are equally probable. To calculate the median of a list of sample values, sort them in any order, and then peek the central value, or the mean between the two central values if the list is even-sized. If your list of values

has a strong central tendency, which manifests itself as a single, well defined peak on the histogram, then the median is a good estimator of the peak position. If the distribution has no central peak, or if it is bimodal (two peaks), then the median is mostly meaningless.

Applied to images, median calculation leads to a useful morphological filter. First define a neighborhood for each pixel in the image. This can be as simple as taking a square box centered on each pixel, or can be somewhat more complex. Anyway, each pixel will have its own associated neighborhood, which will consist on a given number of surrounding source pixels. Now calculate the median of each neighborhood and store all of them in a safe place. Finally, replace each pixel with the median value of its associated neighborhood.

Among other applications, a median filter can be a useful tool for noise reduction. Why? There is a type of noise, known as impulsional noise, that is characterized by bright and/or dark high-frequency (small in relative scale) features appearing randomly over the image. Statistically, impulsional noise falls well outside the peak of the distribution of any given pixel neighborhood, so the median is well suited to learn where impulsional noise is not present, and hence to remove it by exclusion.

Stellar objects can be interpreted as a sort of impulsional noise in deep-sky images, especially when they are numerous: they are more or less randomly distributed, they are bright, and they are small. Taking advantage of this fact, the median filter can be used to remove or isolate stars on deep-sky astrophotos, usually with the purpose of building masks.

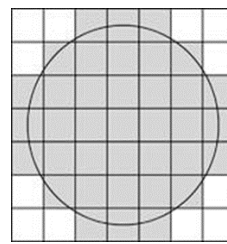
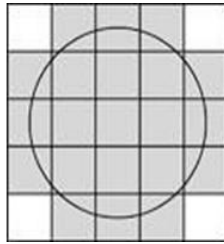
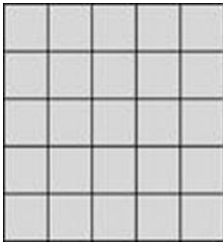
The MedianFilter process in PixInsight is defined by the following parameters:

Box Size

This is the odd size of the neighborhood considered for each pixel. The smallest box size is 3 pixels.

Kernel Type

This refers to the shape and functionality of the kernel (pixel neighborhood) used. A square kernel is used by default, but circular kernels and two-way hybrid kernels are also available. These kernels are explained in the figures below. Shaded squares correspond to pixels used to calculate median values, and white squares correspond to pixels that are ignored in the filtering process.



Circular kernel, box size = 5

Square kernel, box size = 5

5 7

Two-way hybrid median kernel, box size = 5. Two median values

D		R		D
	D	R	D	
R	R	C	R	R
	D	R	D	
D		R		D

are calculated: **Mr** is the median of horizontal and vertical

"**R**" pixels, and **Md** is the median of diagonal "**D**" pixels. The filter value is the arithmetic mean of the two median values and the

central pixel **C**, that is,

$$(\mathbf{Mr} + \mathbf{Md} + \mathbf{C}) / 3$$

A hybrid median filter preserves edges much better than a square-box median filter because it's a three-step ranking operation: data from different spatial directions are ranked separately. Circular kernels can help preserving small structures, especially rounded ones, like stars.

Amount

The Amount parameter modulates the filtering process by mixing filtered and original pixels. If m is the median-filtered pixel value corresponding to an original pixel value f , and a is the Amount parameter, then the resulting pixel value g is given by:

$g = a \times m + (1 - a) \times f$ where pixel values are in the normalized $[0,1]$ interval.

Number of Iterations

In PixInsight, morphological filters can be applied recursively. This is useful when combined with modulation, that is, when the Amount parameter is less than one.

When a median filter is applied this way, a fraction of the original unprocessed data is fed into the algorithm at the beginning of each successive iteration, and the whole process converges to a final result that can yield better detail and contrast preservation.

6-DIGITAL IMAGE PROCESSING

The identification of objects in an image would probably start with image processing techniques such as noise removal, followed by (low-level) feature extraction to locate lines, regions and possibly areas with certain textures.

The clever bit is to interpret collections of these shapes as single objects, e.g. cars on a road, boxes on a conveyor belt or cancerous cells on a microscope slide. One reason this is an AI problem is that an object can appear very different when viewed from different angles or under different lighting. Another problem is deciding what features belong to what object and which are background or shadows etc. The human visual system

performs these tasks mostly unconsciously but a computer requires skillful programming and lots of processing power to approach human performance. Manipulating data in the form of an image through several possible techniques. An image is usually interpreted as a twodimensional array of brightness values, and is most familiarly represented by such patterns as those of a photographic print, slide, television screen, or movie screen. An image can be processed optically or digitally with a computer.

To digitally process an image, it is first necessary to reduce the image to a series of numbers that can be manipulated by the computer. Each number representing the brightness value of the image at a particular location is called a picture element, or pixel. A typical digitized image may have 512×512 or roughly 250,000 pixels, although much larger images are becoming common. Once the image has been digitized, there are three basic operations that can be performed on it in the computer. For a point operation, a pixel value in the output image depends on a single pixel value in the input image. For local operations, several neighbouring pixels in the input image determine the value of an output image pixel. In a global operation, all of the input image pixels contribute to an output image pixel value.

These operations, taken singly or in combination, are the means by which the image is enhanced, restored, or compressed. An image is enhanced when it is modified so that the information it contains is more clearly evident, but enhancement can also include making the image more visually appealing.

An example is noise smoothing. To smooth a noisy image, median filtering can be applied with a 3×3 pixel window. This means that the value of every pixel in the noisy image is recorded, along with the values of its nearest eight neighbours. These nine numbers are then ordered according to size, and the median is selected as the value for the pixel in the new image. As the 3×3 window is moved one pixel at a time across the noisy image, the filtered image is formed.

Another example of enhancement is contrast manipulation, where each pixel's value in the new image depends solely on that pixel's value in the old image; in other words, this is a point operation. Contrast manipulation is commonly performed by adjusting the brightness and contrast controls on a television set, or by controlling the exposure and development time in printmaking. Another point operation is that of pseudo colouring a black-and-white image, by assigning arbitrary colours to the gray levels. This technique is popular in thermograph (the imaging of heat), where hotter objects (with high pixel values) are assigned one color (for example, red), and cool objects (with low pixel values) are assigned another color (for example, blue), with other colours assigned to intermediate values.

Recognizing object classes in real-world images is a long standing goal in Computer vision. Conceptually, this is challenging due to large appearance variations of object instances belonging to the same class. Additionally, distortions from background clutter, scale, and viewpoint variations can render appearances of even the same object instance to be vastly different. Further challenges arise from interclass similarity in which instances from different classes can appear very similar. Consequently, models for object classes must be flexible enough to accommodate class variability, yet discriminative enough to sieve out true object instances in cluttered images. These seemingly paradoxical requirements of an object class model make recognition difficult. This paper addresses two goals of recognition are image classification and object detection. The task of image classification is to determine if an object class is present in an image, while object detection localizes all instances of that class from an image. Toward these goals, the main contribution in this paper is an approach for object class recognition that employs edge information only. The novelty of our approach is that we represent contours by very simple and generic shape primitives of line segments and ellipses, coupled with a flexible method to learn discriminative primitive combinations. These primitives are complementary in nature, where line segment models straight contour and ellipse models curved contour. We choose an ellipse as it is one of the simplest circular shapes, yet is sufficiently flexible to model curved shapes. These shape

primitives possess several attractive properties. First, unlike edge-based descriptors they support abstract and perceptually meaningful reasoning like parallelism and adjacency. Also, unlike contour fragment features, storage demands by these primitives are independent of object size and are efficiently represented with four parameters for a line and five parameters for an ellipse.

Additionally, matching between primitives can be efficiently computed (e.g., with geometric properties), unlike contour fragments, which require comparisons between individual edge pixels. Finally, as geometric properties are easily scale normalized, they simplify matching across scales. In contrast, contour fragments are not scale invariant, and one is forced either to rescale fragments, which introduces aliasing effects (e.g., when edge pixels are pulled apart), or to resize an image before extracting fragments, which degrades image resolution.

In recent studies it is shown that the generic nature of line segments and ellipses affords them an innate ability to represent complex shapes and structures. While individually less distinctive, by combining a number of these primitives, we empower a combination to be sufficiently discriminative. Here, each combination is a two-layer abstraction of primitives: pairs of primitives (termed shape tokens) at the first layer, and a learned number of shape tokens at the second layer. We do not constrain a combination to have a fixed number of shape-tokens, but allow it to automatically and flexibly adapt to an object class. This number influences a combination's ability to represent shapes, where simple shapes favor fewer shape-tokens than complex ones. Consequently, discriminative combinations of varying complexity can be exploited to represent an object class. We learn this combination by exploiting distinguishing shape, geometric, and structural constraints of an object class. Shape constraints describe the visual aspect of shape tokens, while geometric constraints describe its spatial layout (configurations). Structural constraints enforce possible poses/structures of an object by the relationships (e.g., XOR relationship) between shape-tokens.

CLASSIFICATION OF IMAGES:

There are 3 types of images used in Digital Image Processing.
They are

1. Binary Image
2. Gray Scale Image
3. Color Image

BINARY IMAGE:

A binary image is a digital image that has only two possible values for each pixel. Typically the two colors used for a binary image are black and white though any two colors can be used. The color used for the object(s) in the image is the foreground color while the rest of the image is the background color.

Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit (0 or 1). This name black and white, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as gray scale images

Binary images often arise in digital image processing as masks or as the result of certain operations such as segmentation, thresholding, and dithering. Some input/output devices, such as laser printers, fax machines, and bi-level computer displays, can only handle bi-level images

GRAY SCALE IMAGE

A gray scale Image is digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray (0-255), varying from black (0) at the weakest intensity to white (255) at the strongest.

Gray scale images are distinct from one-bit black-and-white images, which in the context of computer imaging are images with only the two colors, black, and white (also called bi-level or binary images). Gray scale images have many shades of gray in between. Gray scale images are also called monochromatic, denoting the absence of any chromatic variation.

Gray scale images are often the result of measuring the intensity of light at each pixel in a single band of the electromagnetic spectrum (e.g. infrared, visible light, ultraviolet, etc.), and in such cases they are monochromatic proper when only a given frequency is captured. But also they can be synthesized from a full color image; see the section about converting to grayscale.

COLOUR IMAGE:

A (digital) color image is a digital image that includes color information for each pixel. Each pixel has a particular value which determines its appearing color. This value is qualified by three numbers giving the decomposition of the color in the three primary colors Red, Green and Blue. Any color visible to human eye can be represented this way. The decomposition of a color in the three primary colors is quantified by a number between 0 and 255. For example, white will be coded as

$R = 255, G = 255, B$

$= 255$; black will be known as $(R,G,B) = (0,0,0)$; and say, bright pink will be : $(255,0,255)$.

In other words, an image is an enormous two-dimensional array of color values, pixels, each of them coded on 3 bytes, representing the three primary colours. This allows the image to contain a total of $256 \times 256 \times 256 = 16.8$ million different colours. This technique is also known as RGB encoding, and is specifically adapted to human vision

It is observable that our behaviour and social interaction are greatly influenced by emotions of people whom we intend to interact with. Hence a successful emotion recognition system could have great impact in

improving human computer interaction systems in such a way as to make them be more user-friendly and acting more human-like.

Moreover, there are a number of applications where emotion recognition can play an important role including biometric authentication, high-technology surveillance and security systems, image retrieval, and passive demographical data collections.

It is unarguable that face is one the most important feature that characterises human beings.

By only looking ones' faces, we are not only able to tell who they are but also perceive a lot of information such as their emotions, ages and genders.

This is why emotion recognition by face has received much interest in computer vision research community over past two decades.

PREPROCESSING

Image Pre-processing is a common name for operations with images at the lowest level of abstraction. Its input and output are intensity images. The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing.

Image restoration is the operation of taking a corrupted/noisy image and estimating the clean original image. Corruption may come in many forms such as motion blur, noise, and camera misfocus. Image restoration is different from image enhancement in that the latter is designed to emphasize features of the image that make the image more pleasing to the observer, but not necessarily to produce realistic data from a scientific point of view. Image enhancement techniques (like contrast stretching or de-blurring by a nearest neighbor procedure) provided by "Imaging packages" use no a priori model of the process that created the image. With image enhancement noise can be effectively be removed by sacrificing some resolution, but this is not acceptable in many applications. In a Fluorescence Microscope resolution in the z-direction is bad as it is. More advanced image processing techniques must be applied to recover the object. De-Convolution is an example of image restoration method. It is capable of: Increasing resolution, especially in the axial direction removing noise increasing contrast.

CONCLUSION

The proposed system accomplishes the task of securing the classified information. Primarily no one gets a hint that some crucial data is being exchanged, and secondly it also gets shielded by using cryptography. The proposed system successfully hides the data using image steganography via LSB and surplus the protection of data using cryptographic technique AES-256 as well as the weak point which a hacker could target that is key, it has also been armored via the use of hashing technique SHA 256. And an impenetrable system is created.

The experimentation performed assures the concreteness of the proposed system. The results acquire a camouflaged image, a non-noticeable distortion making it almost non-viable to attract the consciousness of attackers and even after getting it traced without an authentic key no one was able to access that information. Hence, proposed system proves to be a highly secure system in the terms of securing classified information and exchanging it and it also seals up with no vulnerability in it.

REFERENCES

1. Provos, Niels, and Peter Honeyman. "Hide and seek: An introduction to steganography." IEEE security & privacy 99.3 (2003): 32-44.
2. Henri Gilbert and Helena Handschuh, "Security Analysis of SHA-256 and Sisters*", 2003.
3. Selent, Douglas. "Advanced encryption standard." Rivier Academic Journal 6.2, ISSN (Online): 2319-7064 Index Copernicus Value (2015): 78.96 ,(2010): 1-14.
4. United States National Security Agency (NSA), U.S. Department of Commerce, National Institute of Standards and Technology (NIST), Information Technology Laboratory (ITL), "Secure Hash Signature Standard (SHS) (FIPS PUB 180-2)", National Institute of Standards and Technology (NIST), 2001 : 9-22.
5. Charles G. Boncelet, Jr., Newark, DE (US); Lisa M. Marvel, Churchville, MD (US); Charles T. Retter, Belcamp, MD (US).

"Spread spectrum and image stenographer", 2003.
6. Po-Yueh Chen* and Hung-Ju Lin, "A DWT based approach for image stenographer", DOI:10.6703/IJASE.2006. 4(3).275, 2006.
7. Domenico Bloisi and Luca Iocchi, "Image based steganography and cryptography",2007.
8. Ali Al-Ataby and Fawzi Al-Naima, "A modified high capacity image steganography technique based on wavelet transform", Vol. 7, No. 4, October 2010.
9. Shailender Gupta, Ankur Goyal, Bharat Bhushan, "Information hiding using least significant bit steganography and cryptography",

10. Saiful Islam*, Mangat R Modi and Phalguni Gupta “Edge-based image steganography”
11. *Khan Muhammad, Jamil Ahmad, Haleem Farman, Muhammad Zubair, “A novel image steganographic approach for hiding text in color images using HSI color model”,
12. Khan Muhammad, Jamil Ahmad, Muhammad Sajjad, Muhammad Zubair, “Secure image steganography using cryptography and image transposition”, October 2015.
13. Mwaffaq Abu-Alhaija “Crypto-Steganographic LSB-based System for AES Encrypted Data”, (*IJACSA*) International Journal of Advanced Computer Science and Applications, Vol. 10, No. 10, 2019.

REFERENCE:

1. Suparatchainman, "CNN- based adversarial embedding for image steganography ", image processing and time consuming. OCTOBER - 2019
2. JunqiWu, Audio steganography based on iterative adversarial attacks against convolution neural networks. Year - 2020.
3. Bolin chen , enhancing JPEG steganography using iterative adversarial examples, year-2019.
4. Xinpengzhang, Separable reversible data hiding encrypted image. Year -2017
5. Shiguo Lian, Commutative encryption and watermarking in video compression. Year – 2018
6. Wenjun Lu, Secure video processing problems and challenges. YEAR – 2016
7. Wien Hong, an improved reversible data hiding in encrypted images using aide match. YEAR – 2016
8. Zhuezang , A generative method for steganography by cover synthesis with auxiliary semantics. YEAR – 2020
9. Sundar sripada, Synthesis and evaluation of improved reference matrix models for high capacity image steganography .YEAR– 2020
10. GEHAN, stego: protecting communication by key-dependent steganography with deep networks. YEAR – 2019