A Project report on

DWT BASED AUDIO STEGANOGRAPHY USING AES ENCRYPTION

Submitted in partial fulfilment of the Academic requirements for the award of the degree of

Bachelor of Technology

In

ELECTRONICS & COMMUNICATION ENGINEERING

by

K. SIDDHARTHA	18H51A04K0
B. SUSMITA	18H51A04M7
G. ANUDEEP	18H51A04N3

Under the guidance of

Mrs. A. Jyothi

Associate Professor, ECE Department

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

(NAAC Accredited with 'A+' Grade & NBA Accredited)

(Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501401

2021-2022

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

(NAAC Accredited with ‘A+’ Grade & NBA Accredited)

(Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501401

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled “**DWT BASED AUDIO STEGANOGRAPHY USING AES ENCRYPTION**” is a bonafide work done by **K.SIDDHARTHA (18H51A04K0)**, **B.SUSMITA(18H51A04M7), G.ANUDEEP(18H51A04N3)** in partial fulfilment of requirements for the award of degree of Bachelor of Technology in Electronics & Communication Engineering, submitted to the Department of Electronics & Communication Engineering, CMR College of Engineering & Technology, Hyderabad during the Academic Year 2021-2022

Mrs. A. JYOTHI
Assistant Professor,
ECE Department (Project Guide)

Prof. E.N.V PURNA CHANDRA RAO
HOD
ECE Department

ACKNOWLEDGEMENT

We are highly indebted and grateful to our guide Mrs. A. JYOTHI, Assistant Professor, Department of ECE, CMRCET for his excellent guidance and constant encouragement throughout for the successful completion of the Project.

We are obliged and grateful to thank, Prof. E.N.V. PURNA CHANDRA RAO HOD**,** Department of ECE, CMRCET**,** for his cooperation in all respects.

We are obliged and grateful to thank, Dr. B. LOKESHWAR RAO, Dean (Academics), CMRCET, for his cooperation in all aspects.

We would like to thank Major Dr. V. A. NARAYANA, Principal, CMRCET, for his support in the course of this project work.

We would like to thank Sri Ch. GOPAL REDDY garu, Secretary& Correspondent of CMRCET, for his cooperation in all respects during the course.

It gives immense pleasure in expressing our deep sense of gratitude to project In- charge, Mr. P. RAVEENDRABABU, Associate Professor, Department of ECE, & project Coordinator Mr. V. PANDURANGA, Associate Professor, Department of ECE CMRCET for their valuable suggestions in each and every review during the course of my project.

Finally, we would like to thank all teaching & non- teaching staff members of the department, for their cooperation and support throughout the duration of our course.

Ultimately we own all our success to our beloved parents, whose vision, love and inspiration has made us to reach out for these glories.

		SIGNATURE
K. SIDDHARTHA	(18H51A04K0)	
B. SUSMITA	(18H51A04M7)	
G. ANUDEEP	(18H51A04N3)	

DECLARATION

We hereby declare that results embodied in this report of project on "**DWT BASED AUDIO STEGANOGRAPHY USING AES ENCRYPTION**" are from work carried out by using partial fulfilment of the requirements for the award of B. Tech degree. We have not submitted this report to any other university for the award of the other degree.

		SIGNATURE
K.SIDDHARTHA	(18H51A04K0)	
B.SUSMITA	(18H51A04M7)	
G.ANUDEEP	(18H51A04N3)	

DATE:

ABSTRACT

The majority of today's information hiding systems uses multimedia objects like image, audio, video. Audio Steganography is a technique used to transmit hidden information by modifying an audio signal in an imperceptible manner. It is the science of hiding some secret text or audio information in a host message. The host message before steganography and stego message after steganography have the same characteristics. Embedding secret messages in digital sound is a more difficult process. Varieties of techniques for embedding information in digital audio have been established. This paper presents Audio steganography based on DWT technique using AES encryption. Audio data hiding is one of the most effective ways to protect the privacy.

INDEX

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1 INTRODUCTION

In cryptography, the structure of a message is scrambled to make it meaningless and unintelligible unless the decryption key is available. It makes no attempt to disguise or hide the encoded message. Basically, cryptography offers the ability of transmitting information between persons in a way that prevents a third party from reading it. Cryptography can also provide authentication for verifying the identity of someone or something [3]. In steganography does not alter the structure of the secret message, but hides it inside a cover image so that it cannot be seen. A message in a cipher text, for instance, might arouse suspicion on the part of the recipient while an “invisible” message created with steganographic methods will not. In other word, steganography prevents an unintended recipient from suspecting that the data exists. In addition, the security of classical steganography system relies on secrecy of the data encoding system. Once the encoding system is known, the steganography system is defeated [4, 1]. The following points can be attributed to the renaissance of steganography

i. Government ban on digital cryptography. Individuals and companies who seek confidentiality look to steganography as an important complementary since combining cryptography and steganography can help in avoiding suspicion and protect privacy.

ii. The increased need to protect intellectual property rights by digital content owners, using efficient watermarking. iii. The trend towards electronic communications and humans desire to conceal messages from curious eyes. With rapid advancement in technology, steganographic software is becoming effective in hiding information in image, video, audio or text files.

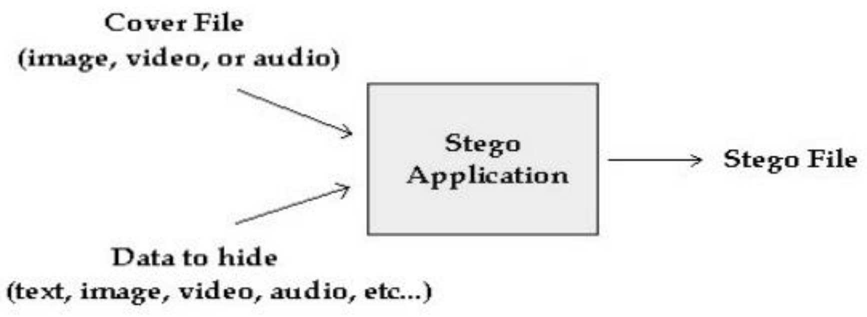


Fig. 1.1. Steganography application scenario

The steganography application hides different types of data within a cover file. The resulting stego also contains hidden information, although it is virtually identical to the cover file. What Steganography essentially does is exploit human perception; human senses are not trained to look for files that have information hidden inside of them, although there are programs available that can do what is called Steganalysis (Detecting use of Steganography.)

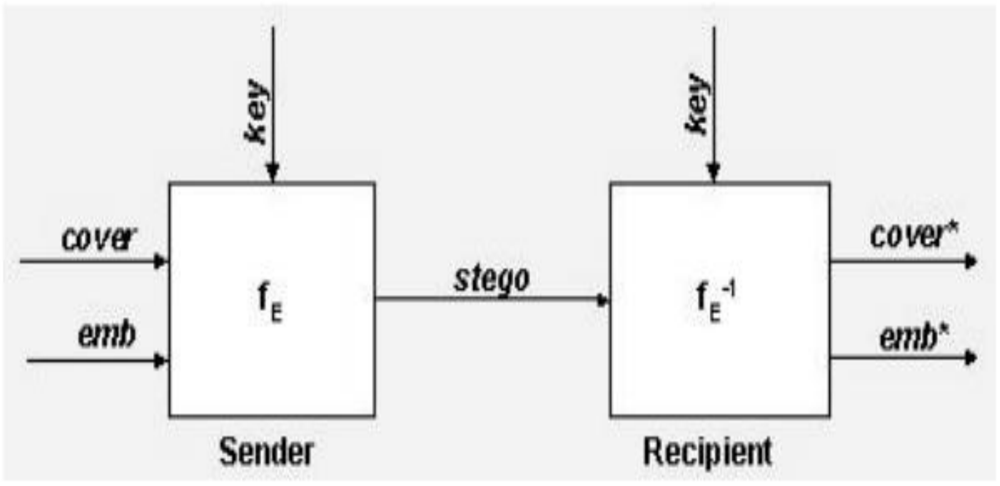


Fig. 1.2. A generic Steganography System

Fig. 2 shows the block diagram of a secure steganographic system. Input messages can be images, texts, video, etc. The components of steganographic system are: Emb: The message to be embedded. Cover: The data in which emb will be embedded. Stego: A modified version of cover that contains the embedded message emb. Key: Additional secret data that is needed for the embedding and extracting processes and must be known to both, the sender and the recipient. fE: A steganographic function that has cover, emb and key as parameters and produces stego as output. FE-1: A steganographic function that has stego and key as parameters and produces emb as output. FE-1 is the inverse function of fE in the sense that the result of the extracting process fE-1 is identical to the input E of the embedding process fE. The International Journal of Multimedia & Its Applications (IJMA) Vol.3, No.3, August 2011 88 The embedding process fE embeds the secret message E in the cover data C. The exact position (S) where E will be embedded is dependence on the key K. The result of the embedding function is slightly modified version of C: the stego data C'. After the recipient has received C' he starts the extracting process fE-1 with the stego data

C’ and the key K as parameters. If the key that is supplied by the recipient is the same as the key used by the sender to embed the secret message and if the stego data the recipient uses as input is the same data the sender has produces (i.e., it has not been modified by an adversary), then the extracting function will produce the original secret message E

Steganography is basically made from two Latin words Steganos and grapter. Steganos means covered and grapter means writing. The objective of steganography is to conceal the content by introduced messages in articles, for example, advanced pictures, audio, video, or content files. The other range of steganography is a copyright checking where the message is embedded to copyright over a record. Steganography and watermarking portray routines to insert data straightforwardly into a bearer signal. Data stowing away in sound signs is increasing boundless significance for secure correspondence of data, for example, clandestine front line and managing an account exchanges by means of open sound approach. Steganography and cryptography are nearly related only the difference in their objectives. Both are used for security purpose but with different approach or implementation. Steganography shifts from cryptography as in wherein cryptography specializes on preserving the substance of an information secret.

Audio steganography is a method for hiding facts interior an audio signal. Present audio steganography software program can embed message in Wav, Au, and maybeMp3 sound data. Embedding hidden message in analog sound is generally a more challenging manner then embedding message in different data together with virtual photographs. It is important to get routines that limit access to those sound documents furthermore for its security. Generally information is inserted in audio records with the end goal of copyright insurance or for confirmation of computerized media. In a PC-based audio Steganography framework, hidden messages a reset up in automatic sound. In Audio Steganography, the shortcoming of the HAS is utilized to conceal data inside the sound.

1.1 HISTORICAL PERSPECTIVE

Signal processing is an immense and diverse field. It is also a field that did not exist 50 years ago and one that remains mysterious, or quite unknown, to most people, Signal processing is not the transmission of signals, as through telephone wires or by radio waves, but the changes made to signals so as to improve transmission or use of the sig nals. Among the processes studied and devised by signal-processing engineers are filter ing, coding, estimating, detecting, analyzing, recognizing, synthesizing, recording, and reproducing. Though signal processing concerns both analog and digital techniques, the field is increasingly dominated by digital techniques. Indeed, the emergence of digital techniques in the 1960s and 1970s played a large part in creating a community of engi neers concerned with signal processing.

At the end of the 20th century, signal processing is a vital technology in many areas: communications, information processing, consumer electronics, control systems, radar and sonar, medical diagnosis, seismology, and scientific instrumentation generally. In addition to the wide range of application areas, there is a wide range of signal-process ing tasks. Examples of these tasks are removing echo from telephone lines, scrambling cellular-phone conversations, controlling the suspension of an automobile so that it responds to road conditions, enabling satellite imaging systems to resolve tiny objects on the ground, and making internal organs stand out in CAT scans. That so many issues related to the processing of signals have arisen in recent decades and have assumed such economic importance is a result of the fact-and further confirms it that ours is an Information Age.

Early Evidence of Steganography:

Even though the term “steganography” was coined at the end of 15th century, the history of steganography dates back to 440 BC. It was derived by combining the two Greek words, “stegano” and “grafia” which means covered/secret and writing/drawing respectively. In olden days, the different ways used for hiding secret messages were tattooing on the scalp of slaves, hiding on tablets covered with wax, writing on the stomachs of rabbits, etc.

.

Tattooing on the scalp of slaves:

The earliest known evidence of steganography was quoted by Herodotus during 484-425 BC. He dictates how Histiaeus (his master), sent one of his slave to the Ionian city of Miletus with a secret information hidden on his body. Firstly, the message was tattooed on the scalp of slave or carrier after shaving his head. The slave was sent to the Ionian city of Miletus only after allowing his hair to grow. Once the messenger reached the destination, the slaves head was again shaved in order to decode the secret message. How Demeratus informed Sparta about Xerxes’s intention to invade Greece was also documented by Herodotus.

Wax coated tablets:

As another early evidence of steganography, wax coated tablets were used for writing text in ancient Greece. To preserve the messages secrecy, the wax coat was removed off the tablets and the message was engraved on the underlying wood. After writing the secret message, the tablet was again covered with wax. As there were no apparent evidence for the presence of message in the tablets, without question they easily passed inspection by sentries. The embedded message can only be decoded by scraping away all of the wax.

Women’s earrings:

An another steganographic approach was proposed by the Greek writer, Aeneas Tactician. His idea was to choose women’s earrings as a media to conceal secret information. He also proposed the idea of using pigeons to deliver secret messages.

Invisible inks:

Apart from the methods mentioned, another ancient technique used to hide messages was using “invisible inks”. Usage of invisible inks to hide data dates back to first century AD. Ancient times, Romans used to hide secret message by writing between lines with the help of invisible inks. Those days the invisible inks were derived from readily available substances such as urine, milk, and fruit juices. By heating the letter, the secret message the letter could be heated, thus darkening the text and the secret message could be revealed or decoded. Another technique used to decode the message was to add little amounts of carbon black or soot on the letter that would stick to the invisible ink.

With the advance of science, as the branch of chemistry progressed new techniques were in cooperated with invisible ink method in order to hide secret data. For example, gallotanic acid made from gallnuts was used as invisible ink. The hidden can be revealed by painting copper sulphate over it. Another early chemical method used during Second World War was to write secret message using solution of copper sulphate on a handkerchief and decoding by exposing to ammonia fumes.

During the fifteenth and sixteenth centuries the steganographic techniques continued to develop and truly flowered in twentieth century.

1.2 INTRODUCTION (DOMAIN-SPECIFIC)

Digital signal processing (DSP) involves developing algorithms that can be used to enhance a signal in a particular way or extract some useful information from it. Digital signal processing (DSP) is the use of digital processing, such as by computers or more specialized digital signal processors, to perform a wide variety of signal processing operations. The digital signals processed in this manner are a sequence of numbers that represent samples of a continuous variable in a domain such as time, space, or frequency. In digital electronics, a digital signal is represented as a pulse train, which is typically generated by the switching of a transistor.

Digital Signal Processing involves the representation, processing, modelling and analysis of signals, information and physical phenomena. DSP interprets the captured data and enables visualization, analysis, manipulation and control. DSP lies at the core of modern artificial intelligence (AI) and machine learning algorithms. Digital Signal Processing is at the core of virtually all of today's information technology, and its impact is felt everywhere -- in telecommunications, medical technology, radar and sonar, and in seismic data analysis Digital signal processing and analog signal processing are subfields of signal processing. DSP applications include audio and speech processing, sonar, radar and other sensor array processing, spectral density estimation, statistical signal processing, digital image processing, data compression, video coding, audio coding, image compression, signal processing for telecommunications, control systems, biomedical engineering, and seismology, among others. DSP can involve linear or nonlinear operations. Nonlinear signal processing is closely related to nonlinear system identification and can be implemented in the time, frequency, and spatio-temporal domains. The application of digital computation to signal processing allows for many advantages over analog processing in many applications, such as error detection and correction in transmission as well as data compression. Digital signal processing is also fundamental to digital technology, such as digital telecommunication and wireless communications. DSP is applicable to both streaming data and static (stored) data. Digital Signal Processing is used everywhere. DSP is used primarily in arenas of audio signal, speech processing, RADAR, seismology, audio, SONAR, voice recognition, and some financial signals. For example, Digital Signal Processing is used for speech compression for mobile phones, as well as speech transmission for mobile phones. DSP is also used in elite headset equipment to protect users from hearing damage; the same suppression and enhancement concept is equally important here. The purpose of Digital Signal Processing is, as mentioned before, to filter the analog signals from current time and space. It is used in a wide variety of technological equipment, but is an especially critical aspect of noise suppression and voice enhancement communication equipment. To digitally analyse and manipulate an analog signal, it must be digitized with an analog-to-digital converter (ADC). Sampling is usually carried out in two stages, discretization and quantization. Discretization means that the signal is divided into equal intervals of time, and each interval is represented by a single measurement of amplitude. Quantization means each amplitude measurement is approximated by a value from a finite set. Rounding real numbers to integers is an example. The Nyquist–Shannon sampling theorem states that a signal can be exactly reconstructed from its samples if the sampling frequency is greater than twice the highest frequency component in the signal. In practice, the sampling frequency is often significantly higher than this. Theoretical DSP analyses and derivations are typically performed on discrete-time signal models with no amplitude inaccuracies (quantization error), "created" by the abstract process of sampling.

1.3 EXISTING METHODOLOGY

- **LSB CODING**

Least significant bit (LSB) coding is the simplest way to embed information in a digital audio file. By substituting the least significant bit of each sampling point with a binary message, LSB coding allows for a large amount of data to be encoded. The following diagram illustrates how the message 'HEY' is encoded in a 16-bit CD quality sample using the LSB method:

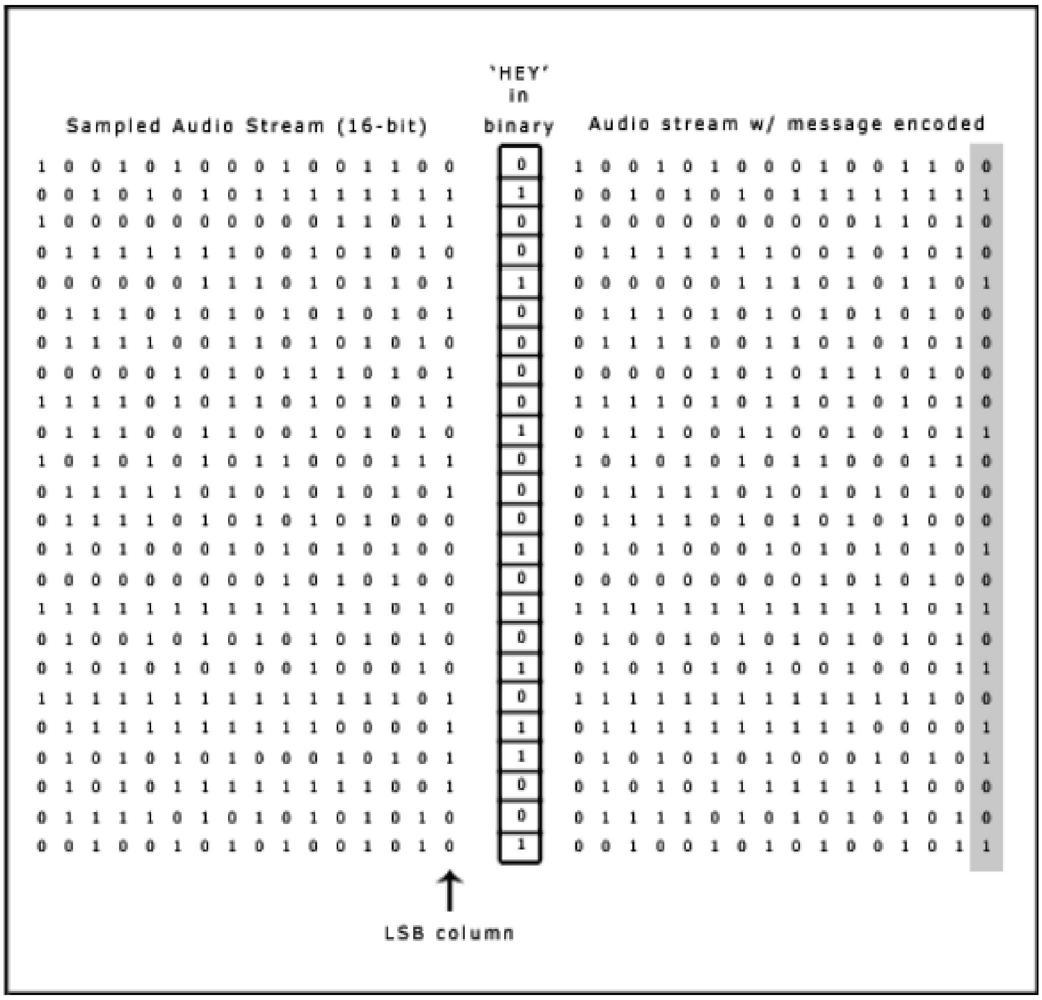

'HEY' in binary
Sampled Audio Stream (16-bit)
Audio stream w/ message encoded
LSB column

Fig. 1.1. LSB Algorithm

- **Standard LSB ALGORITHM:**

It performs bit level manipulation to encode the message. The following steps are a. Receives the audio file in the form of bytes and converted in to bit pattern. b. Each character in the message is converted in bit pattern. c. Replaces the LSB bit from audio with LSB bit from character in the message.

In LSB coding, the ideal data transmission rate is 1 kbps per 1 kHz. In some implementations of LSB coding, however, the two least significant bits of a sample are replaced with two message bits. This increases the amount of data that can be encoded but also increases the amount of resulting noise in the audio file as well. Thus, one should consider the signal content before deciding on the LSB operation to use. For example, a sound file that was recorded in a bustling subway station would mask low-bit encoding noise. On the other hand, the same noise would be audible in a sound file containing a piano solo. The main advantage of the LSB coding method is low computational complexity of the algorithm while its major disadvantage: As the number of used LSBs during LSB coding increases or, equivalently, depth of the modified LSB layer becomes larger, probability of making the embedded message statistically detectable increases and perceptual transparency of stego objects is decreased. Low Bit Encoding is therefore an undesirable method, mainly due to its failure to meet the Steganography requirement of being undetectable.

- **PHASE CODING**

  Phase coding addresses the disadvantages of the noise-inducing methods of audio Steganography. Phase coding relies on the fact that the phase components of sound are not as perceptible to the human ear as noise is. Rather than introducing perturbations, the technique encodes the message bits as phase shifts in the phase spectrum of a digital signal, achieving an inaudible encoding in terms of signal-to-perceived noise ratio.

  The phase coding method breaks down the sound file into a series of N segments. A Discrete Fourier Transform (DFT) is applied to each segment to create a matrix of the phase and magnitude. The phase difference between each segment is calculated, the first segment (s0) has an artificial absolute phase of p0 created, and all other segments have newly created phase frames. The new phase and original magnitude are combined to get the new segment, Sn.

These new segments are then concatenated to create the encoded output and the frequency remains preserved. In order to decode the hidden information, the receiver must know the length of the segments and the data interval used. The first segment is detected as a 0 or a 1 and this indicates where the message starts. This method has many advantages over Low Bit Encoding, the most important being that it is undetectable to the human ear. Like all of the techniques described so far though, its weakness is still in its lack of robustness to changes in the audio data. Any single sound operation or change to the data would distort the information and prevent its retrieval.

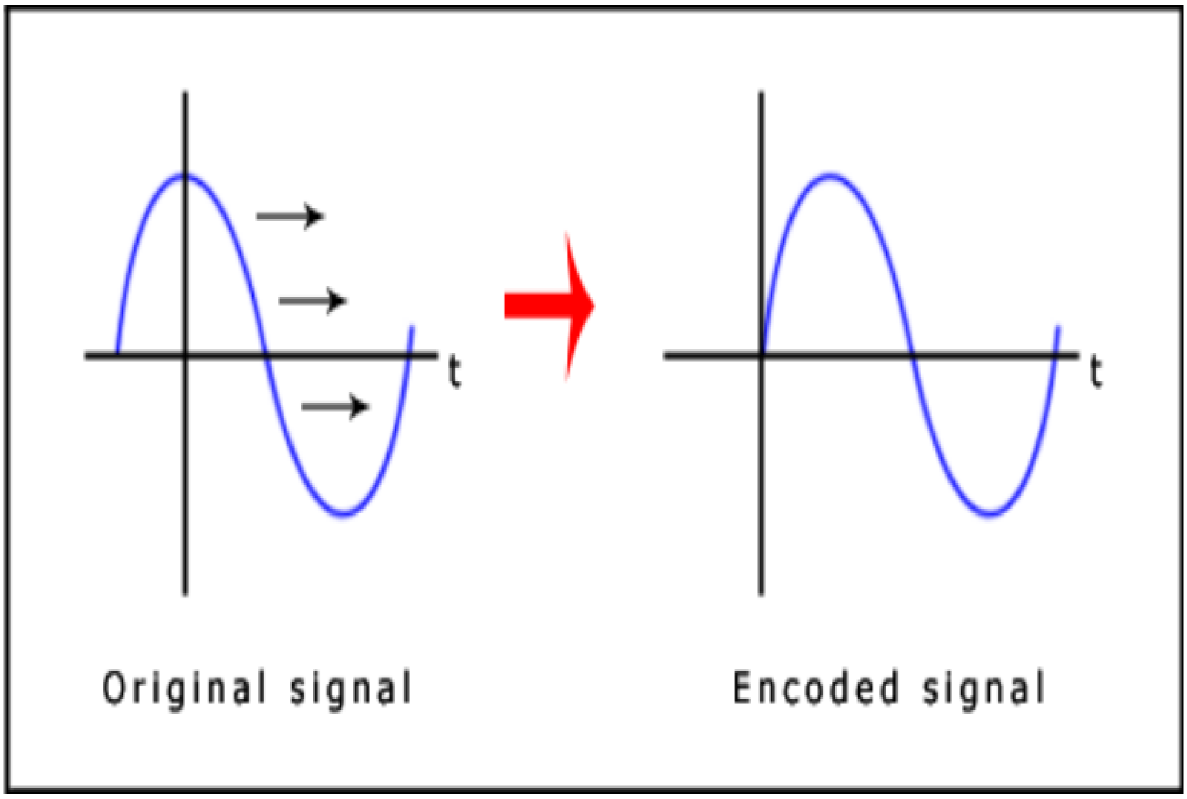


Fig. 1.2. Phase Coding

- **ECHO HIDING**

Echo hiding embeds its data by creating an echo to the source audio. Three parameters of this Artificial echo are used to hide the embedded data, the delay, the decay rate and the initial amplitude. As the delay between the original source audio and the echo decrease it becomes harder for the human ear to distinguish between the two signals until eventually a created carrier sound's echo is just heard as extra resonance.

In addition, offset is varied to represent the binary message to be encoded. One offset value represents a binary one, and a second offset value represents a binary zero. If only one echo was produced from the original signal, only one bit of information could be encoded. Therefore, the original signal is broken down into blocks before the encoding process begins. Once the encoding process is completed, the blocks are concatenated back together to create the final signal.

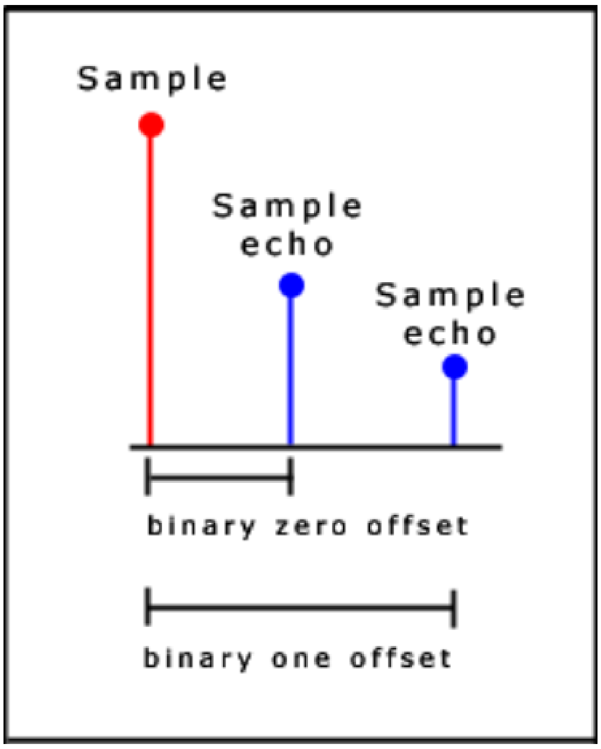


Fig. 1.3. Final signal

The blocks are recombined to produce the final signal. The "one" echo signal is then multiplied by the "one" mixer signal and the "zero" echo signal is multiplied by the "zero" mixer signal. Then the two results are added together to get the final signal. The final signal is less abrupt than the one obtained using the first echo hiding implementation. This is because the two mixer echoes are complements of each other and that ramp transitions are used within each signal. These two characteristics of the mixer signals produce smoother transitions between echoes.

The following diagram summarizes the second implementation of the echo hiding process.

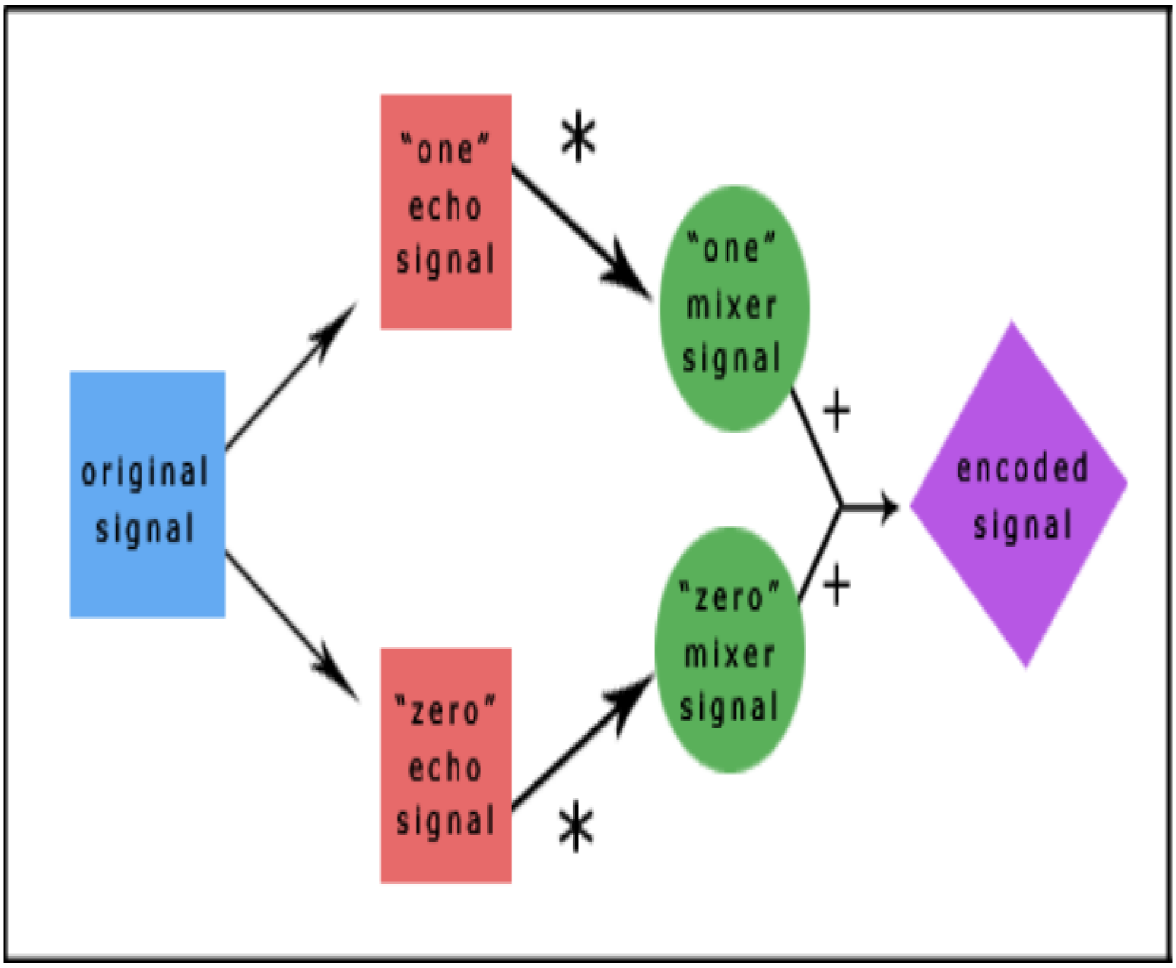


Fig. 1.4. Second Implementation

To extract the secret message from the stego-signal, the receiver must be able to break up the signal into the same block sequence used during the encoding process. Then the autocorrelation function of the signal's spectrum (the spectrum is the Forward Fourier Transform of the signal's frequency spectrum) can be used to decode the message because it reveals a spike at each echo time offset, allowing the message to be reconstructed.

Much like phase encoding this has considerably better results than Low Bit Encoding and makes good use of research done so far in psychoacoustics. As with all sound file encoding, we find that working in audio formats such as WAV is very costly, more so than with bitmap images in terms of the "file size to storage capacity" ratio. The transmission of audio files via e-mail or over the web is much less prolific than image files and so is much more suspicious in comparison. It allows for a high data transmission rate and provides superior robustness when compared to the noise inducing methods.

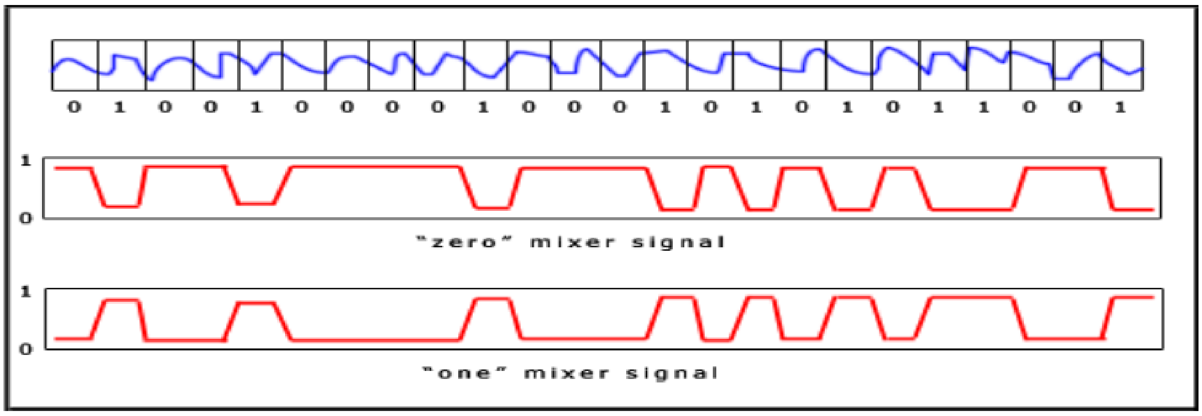


Fig. 1.5. Echo Hiding

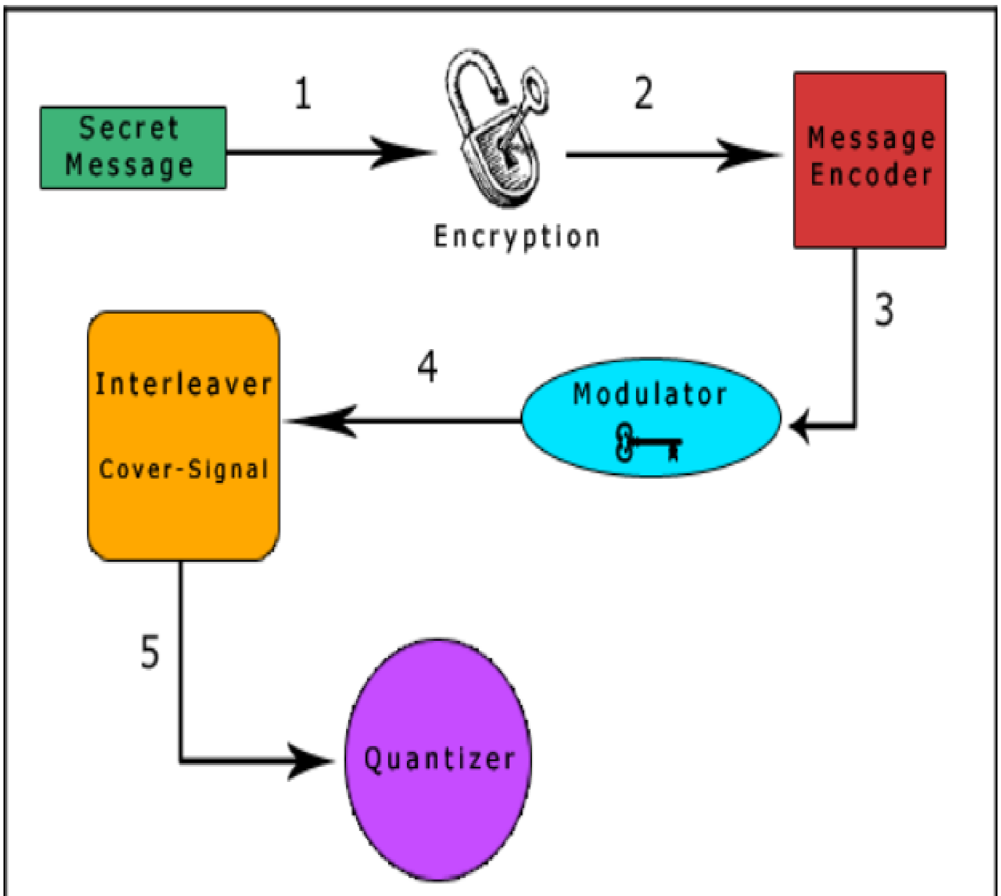
- **SPREAD SPECTRUM**

Spread spectrum systems encode data as a binary sequence which sounds like noise but which can be recognized by a receiver with the correct key. The technique has been used by the military since the 1940s because the signals are hard to jam or intercept as they are lost in the background noise. Spread spectrum techniques can be used for watermarking by matching the narrow bandwidth of the embedded data to the large bandwidth of the medium.

Two versions of SS can be used in audio Steganography: the direct-sequence and frequency hopping schemes. In direct-sequence SS, the secret message is spread out by a constant called the chip rate and then modulated with a pseudorandom signal. It is then interleaved with the cover-signal. In frequency-hopping SS, the audio file's frequency spectrum is altered so that it hops rapidly between frequencies.

Spread Spectrum Steganography has significant potential in secure communications – commercial and military. Audio Steganography in conjunction with Spread Spectrum may provide added layers of security. Spread spectrum encoding techniques are the most secure

means by which to send hidden messages in audio, but it can introduce random noise to the audio thus creating the chance of data loss. They have the potential to perform better in some areas than LSB coding, parity coding, and phase coding techniques in that it offers a moderate data transmission rate while also maintaining a high level of robustness against removal techniques. The following procedural diagram illustrates the design:



1. The secret message is encrypted using a symmetric key, *k1*.
2. The encrypted message is encoded using a low-rate error-correcting code. This step increases the overall robustness of the system.
3. The encoded message is then modulated with a pseudorandom signal that was generated using a second symmetric key, *k2*, as a seed.
4. The resulting random signal that contains the message is interleaved with the cover-signal.
5. The final signal is quantized to create a new digital audio file that contains the message.
6. This process is reversed for message extraction.

Fig. 1.5. Final design

- **PARITY CODING**

Parity coding is one of the robust audio steganographic technique. Instead of breaking a signal into individual samples, this method breaks a signal into separate samples and embeds each bit of the secret message from a parity bit. If the parity bit of a selected region does not match the secret bit to be encoded, the process inverts the LSB of one of the samples in the region. Thus, the sender has more of a choice in encoding the secret bit. Figure 6, shows the parity coding procedure.

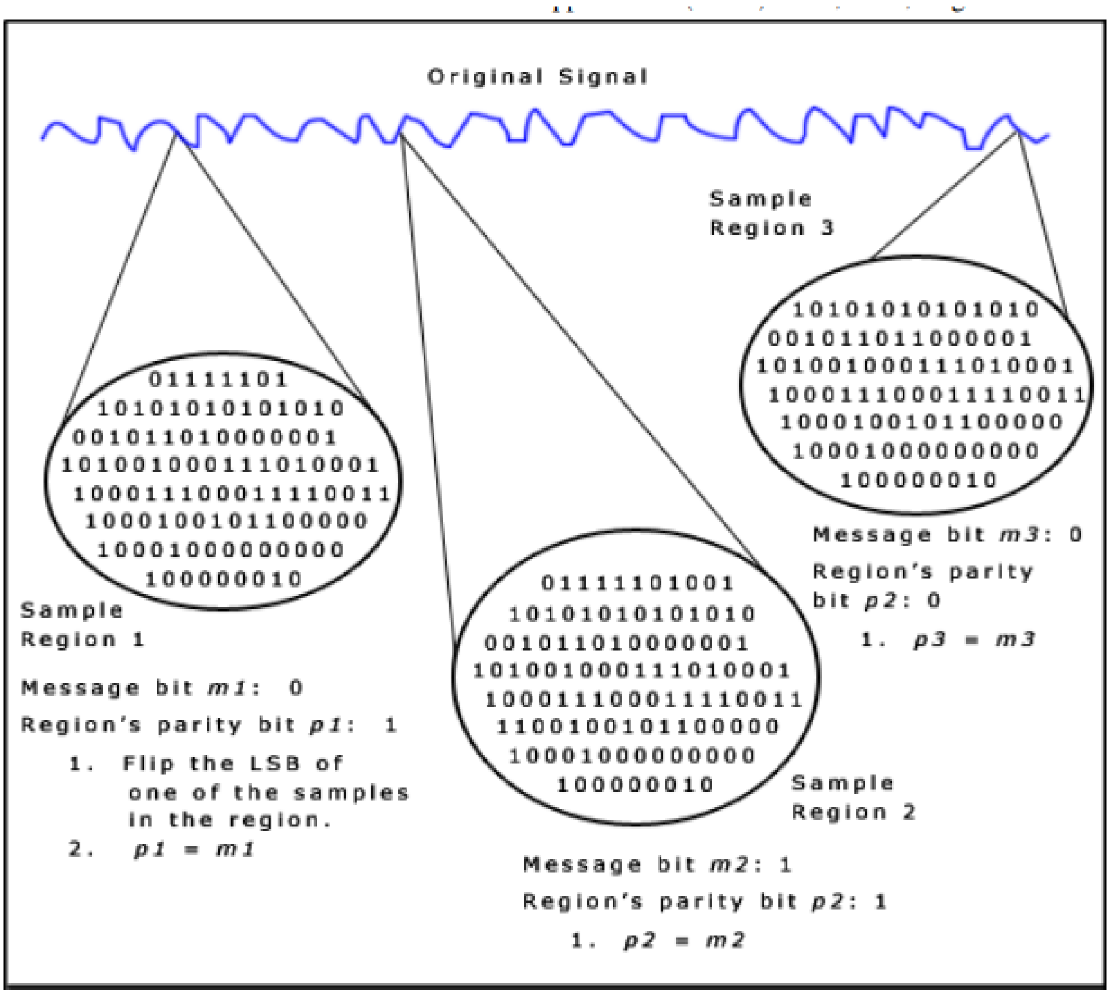


Fig. 1.3.6. Parity Encoding

1.4 OBJECTIVES

The goal of modern cryptography is to ensure the preservation of information properties through mathematically sound means. There are many attributes of information that can be assured in this way, but here are three big ones:

Confidentiality is the assurance that only the intended recipient of a message can read it. This is what most people think of when they hear "cryptography," and it is the primary goal of classical cryptography.

Integrity is the assurance that a piece of information has not been altered. This is the purpose of message authentication codes (MACs), message digests, and cryptographic hashing.

Authenticity is the assurance that the sender of a message is who they say they are. This is the realm of digital signature, and public-key cryptography in general.

1.5 OUTLINE OF THE PROJECT

The word steganography comes from the Greek Steganos, which means covered or secret and -graphy means writing or drawing. Therefore, steganography means, literally, covered writing. Steganography is the art and science of hiding secret information in a cover file such that only sender and receiver can detect the existence of the secret information. A secret information is encoded in a manner such that the very existence of the information is concealed.

The main goal of steganography is to communicate securely in a completely undetectable manner and to avoid drawing suspicion to the transmission of a hidden data. It is not only preventing others from knowing the hidden information, but it also prevents others from thinking that the information even exists. If a steganography method causes someone to suspect there is a secret information in a carrier medium, then the method has failed.

The basic model of Audio steganography consists of Carrier (Audio file), Message and Password. Carrier is also known as a cover-file, which conceals the secret information. Basically, the model for steganography is shown in Fig. Message is the data that the sender wishes to remain it confidential. Message can be plain text, image, audio or any type of file. Password is known as a stego-key, which ensures that only the recipient who knows the corresponding decoding key will be able to extract the message from a cover-file. The cover-file with the secret information is known as a stego-file.

.

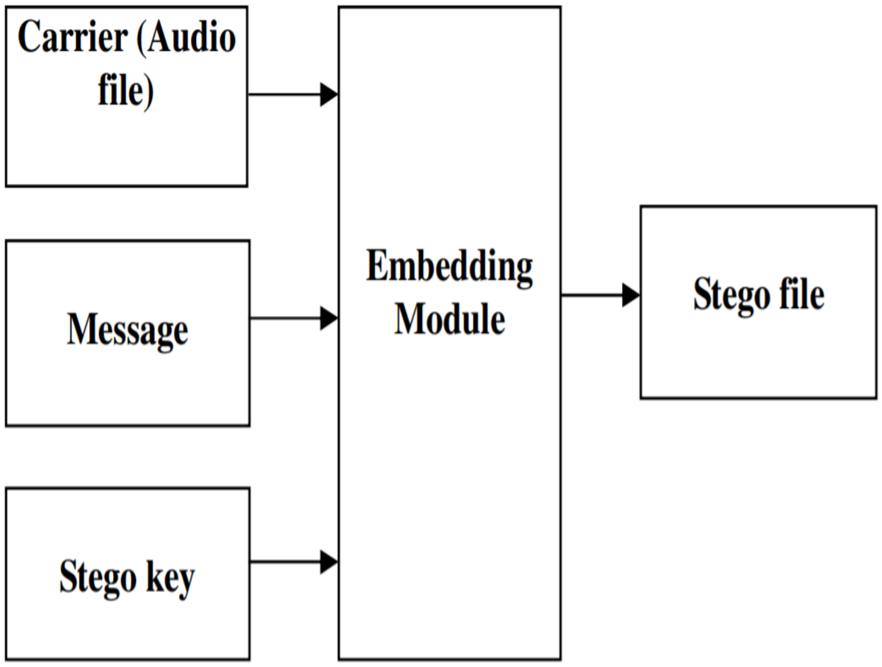


Fig. 1.5.1. Model for steganography

The information hiding process consists of following two steps:

- Identification of redundant bits in a cover-file. Redundant bits are those bits that can be modified without corrupting the quality or destroying the integrity of the cover-file.
- To embed the secret information in the cover file, the redundant bits in the cover file is replaced by the bits of the secret information

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE REVIEW

1. Data Hiding System Using Cryptography & Steganography: A Comprehensive Modern Investigation.

Reference: International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056 Volume: 02 Issue: 01 | Apr-2015 pISSN: 2395-0072.

Aarti Mehndiratta, 2015 presents a paper for different steganographic techniques and the basic knowledge of the encryption and decryption techniques based on symmetric and asymmetric cryptography.

- The steganography techniques are DCT, DWT & the encryption techniques are RSA and DES.
- DCT transforms a cover signal from spatial domain to frequency domain.
- The DWT divides the signal into a high frequency coefficient and the low frequency coefficient.
- Cryptography techniques RSA & DES are used for the data encryption.

Advantage:

Provides a high level of security and robustness.

Disadvantage:

The one of the disadvantages of DCT is it works only with the JPEG format for cover image file

2.Genetic Algorithm in Audio Steganography

Reference: International Journal of Engineering Trends and Technology (IJETT) – Volume 13 Number 1 – Jul 2014.

Nishu Gupta, Mrs.Shailja, 2014 shows a new technique at which the data is firstly encrypted and then after it is encrypted by using DES algorithm.

After the encryption process the encrypted data is embedded into the cover media. This enhances the security of secret messages.

Hashing functions are used to maintain the confidentiality of information.

Disadvantage:

This technique still can be attacked from unauthorized access.

3. Secret Message Integrity of Audio Steganography

Reference: IJCSNS International Journal of Computer Science and Network Security, VOL.15 No.7, July 2014.

Uma Mehta, Mr. Daulat Sihag, 2014 presents technique over the traditional LSB techniques.

- At this technique data must be encrypted by using blowfish algorithm which is much better than DES and RSA algorithms.
- Blowfish algorithm takes lots of time to trying to reconstruct the original message & provides level of security.
- By using the audio file & image some part of message is hidden into the audio and some part is hidden into the image & also hashing function is applied.

Disadvantage:

Time consuming process in reconstructing the original signal. Due to this technique several attacks can affect the original message.

4. Audio Steganography and Security

Reference: www.ijetae.com (ISSN 2250- 2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 2, February 2014).

- Rohit Tanwar, Bhasker Sharma, Sona Malhotra, 2014 presents a new robust substitution technique over the traditional substitution technique.
- The new approach presented here is the data of secret message is hidden into the 3rd & 4th LSB of the cover signal.
- In proposed method currently uses 2 bits per byte of audio sample. The technique uses the deeper layer of cover signal to hide the secret message.

5. A Novel Approach for Embedding Text in Audio to Ensure Secrecy

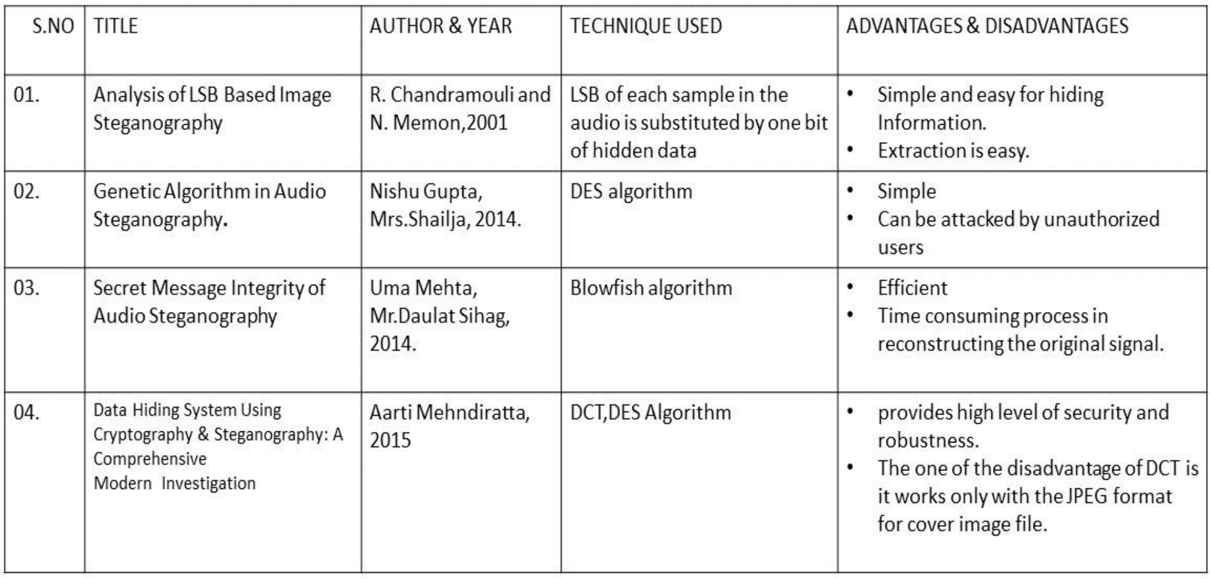
Reference: (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (4),2012.

Prof. Samir Kumar Bandyopadhyay and Barnali Gupta Banik, 2012 represented a multilevel steganography by sending two secret messages on a single cover object.

- In this paper two level securities were presented by using the techniques LSB modification and parity encoding.
- At first level the text message embedded into the audio cover file & generates a stego file.
- After first encryption this stego file used as a cover file for next secret message & generates another stego file. This final stego file contains both the secret message.

Advantage:

This method is providing the two-level security of message & output stego object is decoded with difficulties which makes this method successful to hide data from adversary.



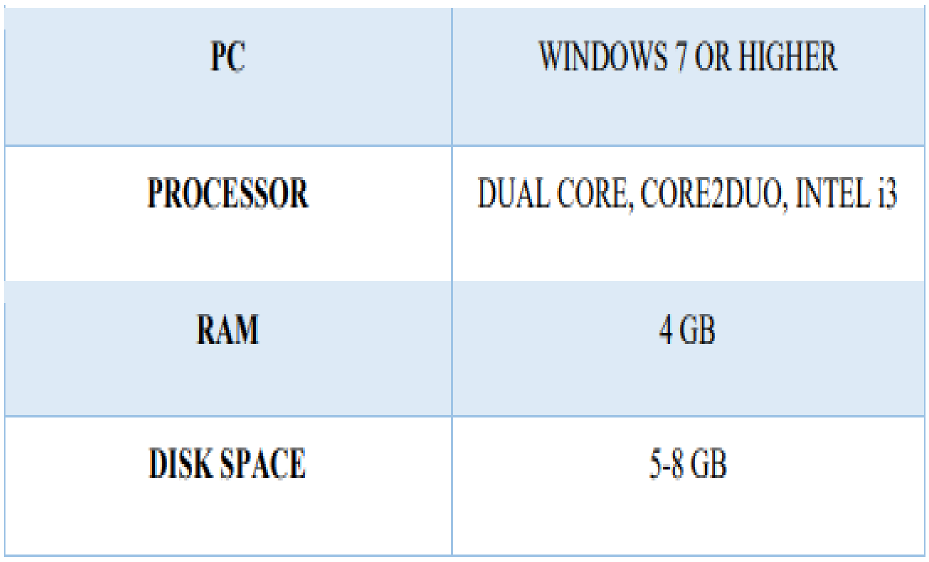
S.NO	TITLE	AUTHOR & YEAR	TECHNIQUE USED	ADVANTAGES & DISADVANTAGES
01.	Analysis of LSB Based Image Steganography	R. Chandramouli and N. Memon,2001	LSB of each sample in the audio is substituted by one bit of hidden data	• Simple and easy for hiding Information. • Extraction is easy.
02.	Genetic Algorithm in Audio Steganography.	Nishu Gupta, Mrs.Shailja, 2014.	DES algorithm	• Simple • Can be attacked by unauthorized users
03.	Secret Message Integrity of Audio Steganography	Uma Mehta, Mr.Daulat Sihag, 2014.	Blowfish algorithm	• Efficient • Time consuming process in reconstructing the original signal.
04.	Data Hiding System Using Cryptography & Steganography: A Comprehensive Modern Investigation	Aarti Mehndiratta, 2015	DCT,DES Algorithm	• provides high level of security and robustness. • The one of the disadvantage of DCT is it works only with the JPEG format for cover image file.

Table 2.1. Literature Survey

CHAPTER 3

TECHNICAL APPROACH

3.1 IDENTIFICATION OF HARDWARE ASPECTS:



PC	WINDOWS 7 OR HIGHER
PROCESSOR	DUAL CORE, CORE2DUO, INTEL i3
RAM	4 GB
DISK SPACE	5-8 GB

Table 3.1. Identification of hardware aspets

3.2 IDENTIFICATION OF SOFTWARE ASPECTS:

- MATLABV.13(R2013a)
- Intel Dual core or core2duo, Intel i3, i5, XP based personal computer.

3.3 CONTENT RELATED TO HARDWARE:

Personal Computer (PC) consists of a central processing unit (CPU) which contains the computers arithmetic, logic and control circuitry on an Integrated Circuit. There are two types of computer memory namely digital random-access memory (RAM) and auxiliary memory, such as magnetic hard disks and special optical compact disks or read only memory (ROM). System should have minimum of 4GB RAM and 5-8 GB of disk space in order to store the audio files.

3.4 CONTENT RELATED TO SOFTWARE:

Fig. 3.4. Matlab logo.

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. MATLAB stands for matrix laboratory, and was written originally to provide easy access to matrix software developed by LINPACK (linear system package) and EISPACK (Eigen system package) projects. MATLAB is therefore built on a foundation of sophisticated matrix software in which the basic element is array that does not require pre dimensioning which to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of time.

CHAPTER 4

IMPLEMENTATION OF THE PROJECT

4.1 BLOCK DIAGRAM:

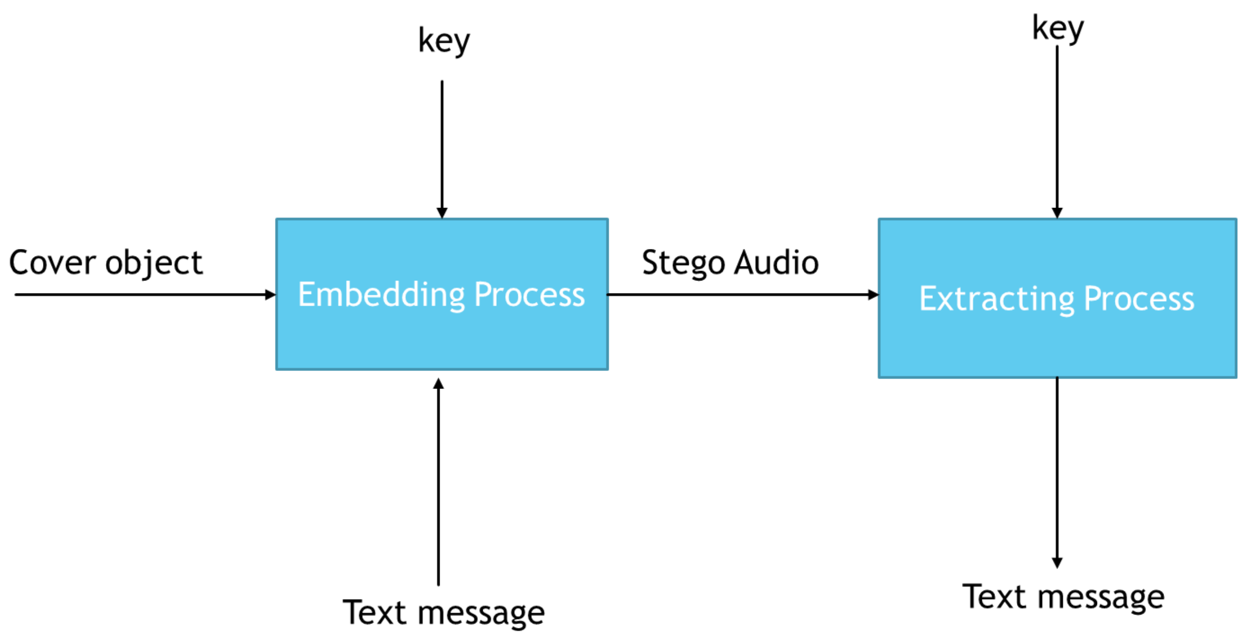


Fig. 4.1. Block Diagram

4.2 CONTENT RELATED TO THE BLOCK DIAGRAM AND INDIVIDUAL BLOCKS:

The block diagram shows that, firstly the secret text is given as input followed by a cover object that is an audio file along with a key. After all the inputs are given the embedding process takes place and the stego audio file is given as output. The stego audio file contains the hidden text. To retrieve the secret text the extraction process takes place where the stego audio file and key is given as input after which the original secret text is given as output.

Encryption Process:

The cover object is an audio file which acts as a cover to the data the user wants to hide. In the cover object it consists of key and text message(secret data) which the user wishes to hide. A key is used as a password for encryption and decryption.

During the encryption the user has to enter the text message which is to be encrypted, along with the key. The data is encrypted using the encryption technique(AES).

After the data is encrypted, it is embedded into the cover file which is an audio file using **DWT** steganography technique. The output of the Embedding Process will be a Stego File. The whole process is known as the Encryption Process.

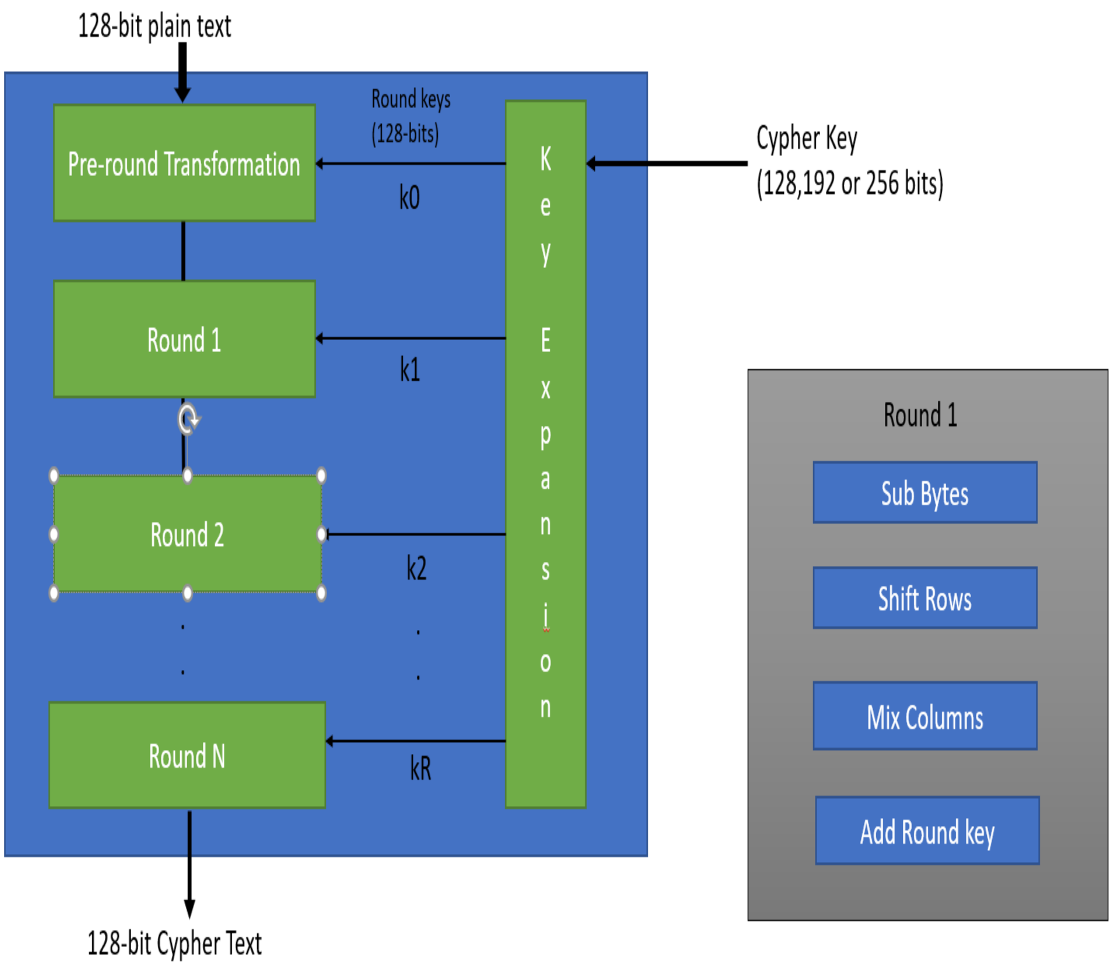


Fig. 4.2. AES Encryption block diagram

- AES is a block cipher.
- The key size can be 128/192/256 bits.
- Encrypts data in blocks of 128 bits each.

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.

Working of the cipher:

AES performs operations on bytes of data rather than in bits. Since the block size is 128 bits, the cipher processes 128 bits (or 16 bytes) of the input data at a time.

The number of rounds depends on the key length as follows:

- 128 bits key – 10 rounds
- 192 bits key – 12 rounds
- 256 bits key – 14 rounds

Creation of Round keys:

A Key Schedule algorithm is used to calculate all the round keys from the key. So the initial key is used to create many different round keys which will be used in the corresponding round of the encryption.

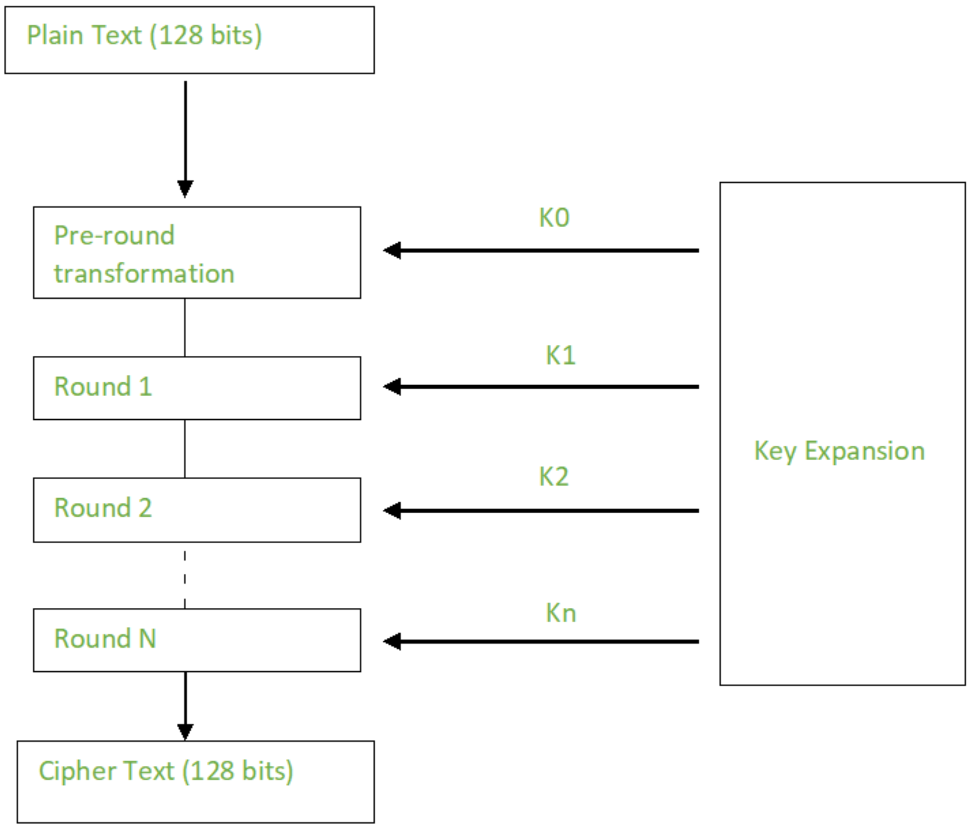


Fig. 4.2.1. Creation of round keys

Encryption:

AES considers each block as a 16 byte (4 byte x 4 byte = 128) grid in a column major arrangement.

[b0 | b4 | b8 | b12 |

| b1 | b5 | b9 | b13 |

| b2 | b6 | b10| b14 |

| b3 | b7 | b11| b15]

Each round comprises of 4 steps:

- SubBytes
- ShiftRows
- MixColumns

- Add Round Key

The last round doesn't have the MixColumns round.

The SubBytes does the substitution and ShiftRows and MixColumns performs the permutation in the algorithm.

SubBytes :

This step implements the substitution.

In this step each byte is substituted by another byte. Its performed using a lookup table also called the S-box. This substitution is done in a way that a byte is never substituted by itself and also not substituted by another byte which is a compliment of the current byte. The result of this step

a 16 byte (4 x 4) matrix like before.

The next two steps implement the permutation.

Shift Rows:

This step is just as it sounds. Each row is shifted a particular number of times.

- The first row is not shifted
- The second row is shifted once to the left.
- The third row is shifted twice to the left.
- The fourth row is shifted thrice to the left.

(A left circular shift is performed.)

```
[ b0  | b1  | b2  | b3  ]         [ b0  | b1  | b2  | b3  ]
| b4  | b5  | b6  | b7  |    ->   | b5  | b6  | b7  | b4  |
| b8  | b9  | b10 | b11 |         | b10 | b11 | b8  | b9  |
[ b12 | b13 | b14 | b15 ]         [ b15 | b12 | b13 | b14 ]
```

Mix Columns:

This step is basically a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.

This step is skipped in the last round.

```
[ c0 ]      [ 2  3  1  1 ]  [ b0 ]
| c1 |      | 1  2  3  1 |  | b1 |
| c2 |      | 1  1  2  3 |  | b2 |
[ c3 ]      [ 3  1  1  2 ]  [ b3 ]
```

Add Round Keys:

Now the resultant output of the previous stage is XOR-ed with the corresponding round key. Here, the 16 bytes is not considered as a grid but just as 128 bits of data.

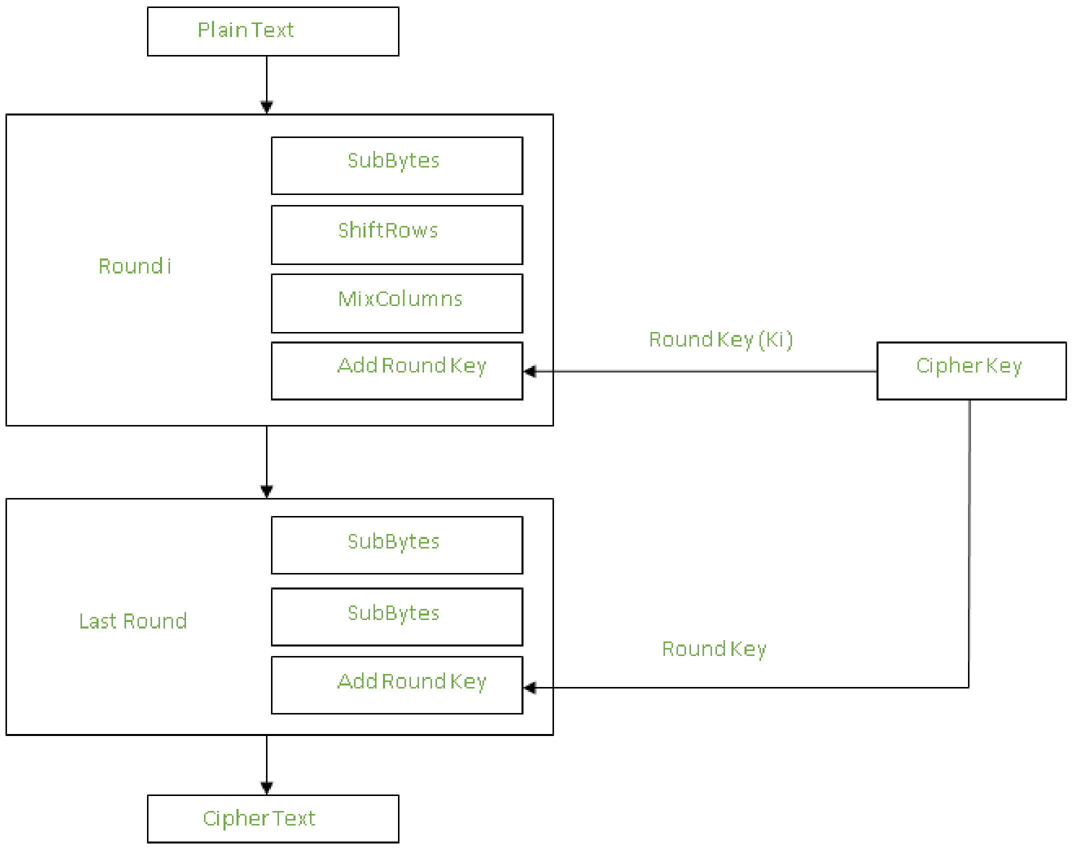


Fig. 4.2.2. Addition of round keys

After all these rounds 128 bits of encrypted data is given back as output. This process is repeated until all the data to be encrypted undergoes this process.

Decryption:

The stages in the rounds can be easily undone as these stages have an opposite to it which when performed reverts the changes. Each 128 blocks goes through the 10,12 or 14 rounds depending on the key size.

The stages of each round in decryption is as follows :

- Add round key
- Inverse MixColumns
- ShiftRows
- Inverse SubByte

The decryption process is the encryption process done in reverse so i will explain the steps with notable differences.

Inverse MixColumns:

This step is similar to the MixColumns step in encryption, but differs in the matrix used to carry out the operation.

$$\begin{bmatrix} b0 \\ b1 \\ b2 \\ b3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \begin{bmatrix} c0 \\ c1 \\ c2 \\ c3 \end{bmatrix}$$

Inverse SubBytes:

Inverse S-box is used as a lookup table and using which the bytes are substituted during decryption.

Embedding Process:

DWT is a way to transform from spatial to frequency domain. Wavelets are basically functions that integrate to zero waving below and above the x axis. For Signal and image processing, wavelets are used as the basic function like sines and cosines.

1.Cover audio file is chosen and read to get the details for frequency and samples.

2. DWT matrix is applied on cover audio files which provides the approximation coefficients (cA) and detail coefficients (cD).

3. Text file is read and encrypted with size calculated in binary.

4. Encrypted text is embedded, where the size of the secret message is rooted because the encryption and decryption of the secret text is based on message size. Here the first cD coefficient

is replaced by the size.

5. Hiding the actual message where the other cD coefficients is substituted by the encrypted message.

6. Stego Audio Signal is reconstructed using inverse DWT (IDWT) of approximate coefficient (cA) and detail coefficient (cD).

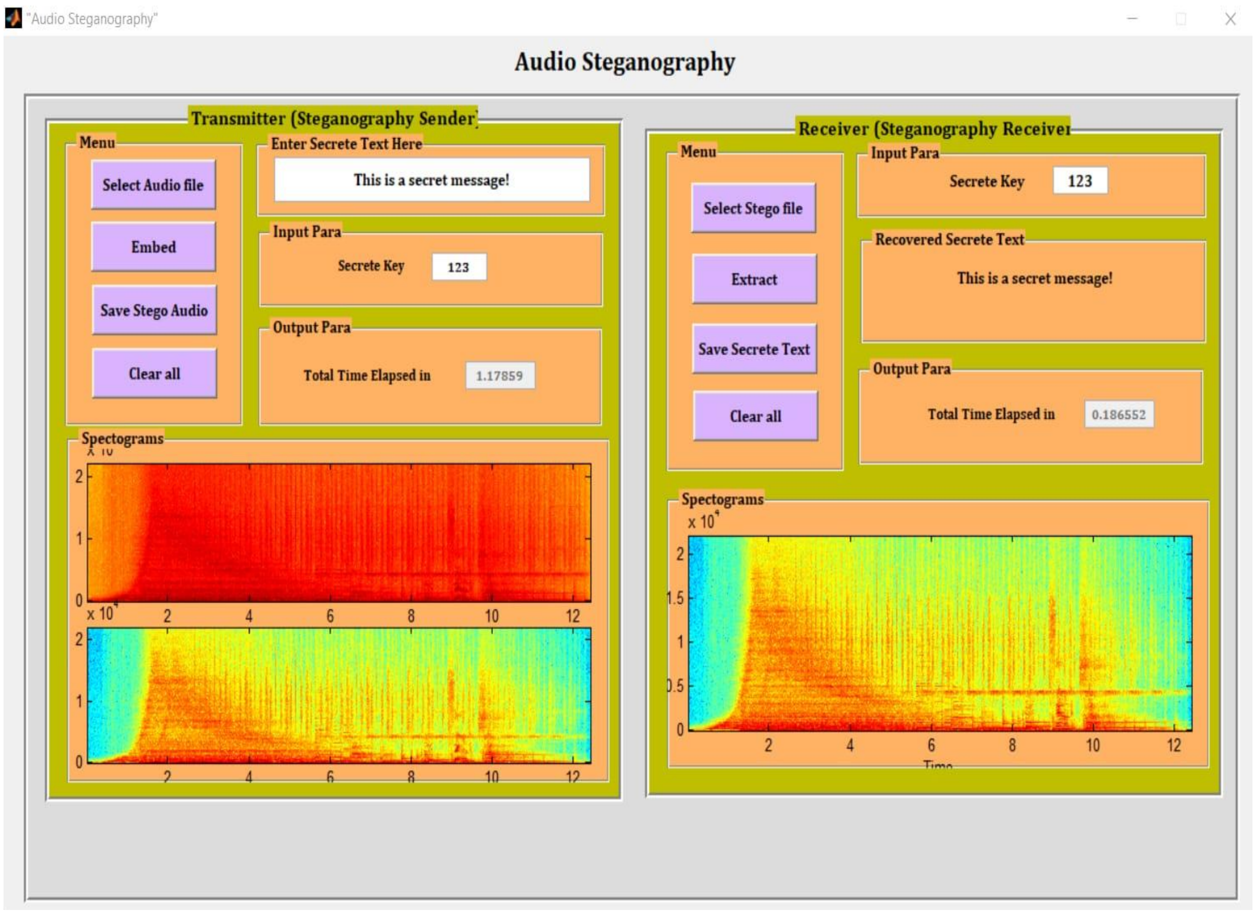
Extracting Process:

After the data is embedded and the stego file is produced, the extraction process is done using decryption technique(AES). In the extracting process the text message(secret data) is extracted from the cover file using the key.

CHAPTER-5

RESULTS AND CONCLUSION

5.1 RESULTS:



Enter Secrete Text Here: This is a secret message!

Input Para: Secrete Key 123

Output Para: Total Time Elapsed in 1.17859

Input Para: Secrete Key 123

Recovered Secrete Text: This is a secret message!

Output Para: Total Time Elapsed in 0.186552

Fig 5.1. Application Output

5.2 CONCLUSION:

In this study, we successfully designed and implemented an effective and user-friendly steganography system technique i.e., “DWT BASED AUDIO STEGANOGRAPHY USING AES ENCRYPTION”. The target users of the system are not only the people who wish their data to protected , but also professional people like who work in army, space organizations like NASA , ISRO. We have achieved ahigh secure results using this method where a user can transmit data without being concerned about the data privacy. However, there are few shortcomings of our project such as, poor transmission, receiving of the signal can cause anomalies in output.

REFERENCES:

[1] I. J. Cox, M. L. Miller, J. A. Bloom, J. Fridrich, and T. Kalker. Digital watermarking and steganography. Morgan Kaufmann, 2008. Hardcover ISBN: 9780123725851

[2] Utepbergenov, Irbulat, Janna Kuandykova, Tamerlan Mussin, and Sholpan Sagyndykova. ―Creating a Program and Research a Cryptosystem on the Basis of Cardan Grille‖, pp. 24–29, Publisher IEEE, 2013. DOI:10.1109/ICoIA.2013.6650224

[3] Judge, J.C., 2001. Steganography: Past, Present, Future. https://www.sans.org/readingroom/whitepapers/stenganography/ steganographypast-present-future52?show=552.php&cat=stenganography

[4] Shelke, S.G., and S.K. Jagtap. ―Analysis of Spatial Domain Image Steganography Techniques‖, pp. 665– 667, Publisher IEEE, 2015. DOI:10.1109/ICCUBEA.2015.136

[5] Bilal, Ifra, Rajiv Kumar, Mahendra Singh Roj, and P K Mishra. ―Recent Advancement in Audio Steganography‖, pp. 402–405, Publisher IEEE, 2014. DOI:10.1109/PDGC.2014.7030779

[6] Dalal M., Juneja M. (2018) ―Video Steganography Techniques in Spatial Domain—A Survey‖, in: Mandal J., Saha G., Kandar D., Maji A. (eds) Proceedings of the International Conference on Computing and Communication Systems, Lecture Notes in Networks and Systems, Volume 24. Publisher Springer, Singapore DOI:10.1007/978-981-10-6890- 4_67

[7] Li, Chang-Tsun, ed. Multimedia Forensics and Security. DOI: 10.4018/978-1-59904-869-7

[8] Johnson N and Jajodia S, ―Exploring Steganography: Seeing the unseen‖, Computer, Volume 31, Issue 2, pp. 26–34. DOI: 10.1109/MC.1998.4655281

[9] Sutaone M and Khandare M, ―Image Based Steganography Using LSB Insertion Technique‖, IET International Conference on Wireless, Mobile and Multimedia Networks, 2008. ISBN: 978-0-86341-887- 7

[10] Deshpande N, Kamalapur S, Jacobs D, ―Implementation of LSB Steganography and Its Evaluation for Various Bits‖, Publisher IEEE, pp. 173– 178. DOI: 10.1109/ICDIM.2007.369349

[11] Gupta Banik, B and Bandyopadhyay, S K, ―An Image Steganography Method on Edge Detection Using Multiple LSB Modification Technique‖, Journal of Basic and Applied Research International, International Knowledge Press, Volume 9 Issue 2 pp. 75

APPENDIX:

MATLAB

MATLAB features a family of applications specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow learning and applying specialized technology. These 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 system, neural networks, fuzzy logic, wavelets, simulation and many others. Typical uses of MATLAB include: Math and computation, Algorithm development, Data acquisition, Modelling, simulation, prototyping, Data analysis, exploration, visualization, Scientific and engineering graphics, Application development, including graphical user interface building. 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. MATLAB stands for matrix laboratory, and was written originally to provide easy access to matrix software developed by LINPACK (linear system package) and EISPACK (Eigen system package) projects. MATLAB is therefore built on a foundation of sophisticated matrix software in which the basic element is array that does not require pre dimensioning which to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of time.

Basic building blocks of MATLAB:

The basic building block of MATLAB is MATRIX. The fundamental data type is the array. Vectors, scalars, real matrices and complex matrix are handled as specific class of this basic data type. The built in functions are optimized for vector operations.

MATLAB Window

The MATLAB works based on five windows: Command window, Workspace window, Current directory window, Command history window, Editor Window, Graphics window and Online-help window.

Command Window

The command window is where the user types MATLAB commands and expressions at the prompt (>>) and where the output of those commands is displayed. It is opened when the application program is launched. All commands including user-written programs are typed in this window at MATLAB prompt for execution.

Workspace Window

MATLAB defines the workspace as the set of variables that the user creates in a work session. The workspace browser shows these variables and some information about them. Double clicking on a variable in the workspace browser launches the Array Editor, which can be used to obtain information.

Current Directory Window

The current Directory tab shows the contents of the current directory, whose path is shown in the current directory window. For example, in the windows operating system the path might be as follows: C:\MATLAB\Work, indicating that directory “work” is a subdirectory of the main directory “MATLAB”; which is installed in drive C. Clicking on the arrow in the current directory window shows a list of recently used paths. MATLAB uses a search path to find M-files and other MATLAB related files. Any file run in MATLAB must reside in the current directory or in a directory that is on search path.

Command History Window

The Command History Window contains a record of the commands a user has entered in the command window, including both current and previous MATLAB sessions.

Editor Window

The MATLAB editor is both a text editor specialized for creating M-files and a graphical MATLAB debugger. The editor can appear in a window by itself, or it can be as u window in the desktop. In this window one can write, edit, create and save programs in files called Mfiles.

MATLAB edit or window has numerous pull-down menus for tasks such as saving, viewing, and debugging files.

Graphics or Figure Window

The output of all graphic commands typed in the command window is seen in this window.

M-Files

These are standard ASCII text file with ‘m’ extension to the file name and creating own matrices using M-files, which are text files containing MATLAB code. MATLAB editor or another text editor is used to create a file containing the same statements which are typed at the MATLAB command line and save the file under a name that ends in .m. There are two types of M-files:

Script Files

It is an M-file with a set of MATLAB commands in it and is executed by typing name of file on the command line. These files work on global variables currently present in that environment.

Function Files

A function file is also an M-file except that the variables in a function file are all local. This type of files begins with a function definition line.

MAT-Files

These are binary data files with '.mat' extension to the file that are created by MATLABwhenthedataissaved.ThedatawritteninaspecialformatthatonlyMATLAB can read. These are located into MATLAB with 'load' command.

MATLAB functions that work with files always accept the full paths to those files as inputs. If you do not specify the full path, then MATLAB looks for files in the current folder first, and then in folders on the *search path*. To make sure that MATLAB finds the file that you expect, you can construct and pass the full path, change to the appropriate folder, or add the folder to the path

ALGEBRIC OPERATIONS IN MATLAB:

Scalar Calculations:

+ Addition

- Subtraction

* Multiplication

/ Right division (a/b means a ÷b)

For example: 3*4 executed in 'matlab' gives ans=124/5 gives ans=0.8

Array products: Recall that addition and subtraction of matrices involved addition or subtraction of the individual elements of the matrices. Sometimes it is desired to simply multiply or divide each element of an matrix by the corresponding element of another matrix 'array operations”. Array or element-by-element operations are executed when the operator is preceded by a '.' (Period): a.*b multiplies each element of a by the respective element of b a./b respective element of b.

a.\b divides each element of b by the respective element of a a.^ b respective b element.

MATLAB WORKING ENVIRONMENT:

divides each element raise each element

of a by the of a by the

MATLAB DESKTOP

MATLAB Desktop is the main MATLAB application window. The desktop contains five sub windows, the command window, the workspace browser, the current directory window, the command history window, and one or more figure windows, which are shown only when the user displays a graphic.

The command window is where the user types MATLAB commands and expressions at the prompt (>>) and where the output of those commands is displayed. MATLAB defines the workspace as the set of variables that the user creates in a work session. can be used to obtain information and income instances edit certain properties of the variable. The current Directory tab above the workspace tab shows the contents of the current directory, whose path is shown in the current directory window. For example, in the windows operating system the path might be as follows: C:\MATLAB\Work, indicating that directory “work” is a subdirectory of the main directory “MATLAB”; WHICH IS INSTALLED IN DRIVE C. clicking on the arrow in the current directory window shows a list of recently used paths. Clicking on the button to the right of the window allows the user to change the current directory. The Command History Window contains a record of the commands a user has entered in the command window, including both current and previous MATLAB sessions. Previously entered MATLAB commands can be selected and re-executed from the command history window by right clicking on a command or sequence of commands. The Command History Window contains a record of the commands a user has entered in the command window, including both current and previous MATLAB sessions. Previously entered MATLAB commands can be selected and re-executed from the command history window by right clicking on a command or sequence of commands.

This action launches a menu from which to select various options in addition to executing the commands. This is useful to select various options in addition to executing the commands. This is a useful feature when experimenting with various commands in a work session.

Online Help Window

MATLAB provides online help for all it’s built in functions and programming language constructs. The principal way to get help online is to use the MATLAB help browser, opened as a separate window either by clicking on the question mark symbol (?) on the desktop toolbar, or by typing help browser at the prompt in the command window. The help Browser is a web browser integrated into the MATLAB desktop that displays a Hypertext Mark-up Language

(HTML) documents. The Help Browser consists of two panes, the help navigator pane, used to find information, and the display pane, used to view the information. Selfexplanatory tabs other than navigator pane are used to perform a search.

MATLAB ONLINE

- MATLAB Online provides access to MATLAB and Simulink from any standard web browser wherever you have internet access – just sign in.
- It is ideal for teaching, learning and convenient, lightweight access.
- Full integration with MATLAB Drive gives you 5GB to store, access, and manage your files from anywhere with MATLAB Online.

- Use MATLAB Drive Connector to sync your files between your computers and MATLAB Online, eliminating the need for manual upload or download.

MATLAB and Simulink for digital signal processing

Signal processing engineers use MATLAB® and Simulink® at all stages of development—from analyzing signals and exploring algorithms to evaluating design implementation tradeoffs for building real-time signal processing systems. MATLAB and Simulink offer:

- Built-in functions and apps for analysis and preprocessing of time-series data, spectral and time-frequency analysis, and signal measurements
- Apps and algorithms to design, analyze, and implement digital filters (FIR and IIR) from basic FIR and IIR filters to adaptive, multirate, and multistage designs
- An environment to model and simulate signal processing systems with a combination of programs and block diagrams
- Capabilities to model fixed-point behavior and automatically generate C/C++ or HDL code for deploying on embedded processors, FPGAs, and ASICs
-
- Tools for developing predictive models on signals and sensor data using machine learning and deep learning workflows
- Signal Analysis and Measurements
- MATLAB and Simulink help you analyze signals using built-in apps for visualizing and preprocessing

- signals in time, frequency, and time-frequency domains to detect patterns and trends without having to manually write code. You can characterize signals and signal processing systems using domain-specific algorithms across different applications such as communications, radar, audio, medical devices, and IoT.

- Model-Based Design for Signal Processing

- When designing signal processing systems, you can use a combination of block diagrams and language-based programming. You can use Simulink to apply Model-Based Design to signal processing systems for modeling, simulation, early verification, and code generation. You can use libraries of blocks with application-specific algorithms for baseline signal processing, audio, analog mixed-signal and RF, wireline and wireless communications, and radar systems. You can visualize live signals during simulations using virtual scopes, including spectrum and logic analyzers, constellations, and eye diagrams.

- Embedded Code Generation

- You can automatically generate C and C++ code from signal processing algorithms and bitaccurate system models using MATLAB Coder™ and Simulink Coder™. The generated code can be used for simulation acceleration, rapid prototyping, and embedded implementation of your system. You can also generate optimized C code for targeting embedded hardware processors such as ARM® Cortex®-A or Cortex-M.

- You can also generate portable, synthesizable Verilog® and VHDL® code from MATLAB functions and Simulink models. The generated HDL code can be used for FPGA programming or ASIC design.

- Machine and Deep Learning.

With MATLAB, you can build predictive models for signal processing applications. You can exploit built-in signal processing algorithms to extract features for machine learning systems as well as work with large datasets for ingesting, augmenting, and annotating signals when developing deep learning applications.

SOURCE CODE:

Encryption:

```
function encrypt = encrypt(edit1,pass1)


global  edit2 edit3  pass2 data_encrypt

global irt endrt ert iprt pt


data = get(edit1,'String');

data = double(uint8(data))';

data_length = length(data);

adds = 8-mod(data_length,8);

if adds ~= 0

    for i = 1:adds

        data(data_length+i) = 0;

    end

    data_length = data_length+adds;

end


key = get(pass1,'String');

keylength = length(key);

keytemp0 = uint8(key);

keys = [149 57 208 147 21 183 27];
```

```
%%%%%%%%%%%--- ---%%%%%%%%%%%
if keylength < 7
   for i = keylength+1:7
       keytemp0(i) = keys(i);
   end
end
for i=1:7
      keyusetemp(i) = keytemp0(i);
end
keyuse = char(keyusetemp);

pwb = str2bin(keyuse,7,2);
pwb = rebit(pwb,iprt);
ki = gerkey(pwb);

times = data_length/8;
for i = 0:times-1
    for j = 1:8
        tempdata(j) = data(8*i+j);
    end
    encrydata = char(tempdata);
    encrydata = des(encrydata,ki);
```

```
for k = 0:7
        temp = encrydata(k*8+1:(k+1)*8);
        data_encrypt(i*8+k+1,1) = bin2dec(temp);
    end
end

data_show =dec2bin(data_encrypt)';
set(edit2,'string',data_show)

function bin = str2bin(str,k,flag)

l = length(str);
bin = [];
temp = [];

for x = 1:l
     temp = [temp,dec2bin(str(x),8)];
end
if flag ~= 0
     n = ceil(l*8/k);
     rb = mod(l*8,k);
     sb = 0;
```

```
if rb ~= 0
  sb = k-rb;
  for i = 1:sb
     z(i) = '0';
  end
  temp = [temp z];
  end
  for x = 0:n-1
    temp1 = temp(x*k+1:(x+1)*k);
    lone = length(find(temp1 == '1'));
    if flag == 1
       if mod(lone,2) == 0
          opb = '1';
       else
          opb = '0';
       end
    else if flag == 2
          if mod(lone,2) == 0
             opb = '0';
          else
             opb = '1';
          end
```

```
 end
        end
       temp1 = [temp1 opb];
       bin = [bin temp1];
    end
else
       bin = temp;
end


function so = rebit(si,k)


lk=length(k);
for i=1:lk
    so(i)=si(k(i));
end
function ki=gerkey(k)
mt = ...
    [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16;
     1 1 2 2 2 2 2 2 1  2  2  2  2  2  2  1];
rt = ...
    [14 17 11 24  1  5  3 28 ...
     15  6 21 10 23 19 12  4 ...
26  8 16  7 27 20 13  2 ...
     41 52 31 37 47 55 30 40 ...
     51 45 33 48 44 49 39 56 ...
     34 53 46 42 50 36 29 32];
kl = k(1:28);
kr = k(29:56);
for i = mt(1,1):mt(1,16)
    kl = mr(kl,mt(2,i));
    kr = mr(kr,mt(2,i));
    k = [kl kr];
    for j = 1:48
        ki(i,j) = k(fix(rt(j)));
    end
end
function nk = mr(k,n)
l = length(k);
k1 = k(n+1:l);
k2 = k(1:n);
nk = [k1 k2];
function ef = des(pf,ki)
global irt endrt ert iprt pt
pfb = str2bin(pf,0,0);
pfb = rebit(pfb,irt);
lpfb = pfb(1:32);
rpfb = pfb(33:64);

for i = 0:15
    templ = lpfb;
    lpfb = rpfb;
    tempr = rebit(rpfb,ert);
    rpfb = char(xor(ki(i+1,:)-48,tempr-48)+48);
    rpfb = sbox(rpfb);
    rpfb = rebit(rpfb,pt);
    rpfb = char(xor(templ-48,rpfb-48)+48);
end
pfb = [rpfb lpfb];
pfb = rebit(pfb,endrt);
ef = pfb;
function so = sbox(si)
sbox1 = ...                             % S1ºÐ
       [14  4 13 1  2 15 11  8  3 10  6 12  5  9 0  7;
         0 15  7 4 14  2 13  1 10  6 12 11  9  5 3  8;
         4  1 14 8 13  6  2 11 15 12  9  7  3 10 5  0;
        15 12  8 2  4  9  1  7  5 11  3 14 10  0 6 13];
```

```
sbox2 = ...                    % S2ºÐ
       [15  1  8 14  6 11  3  4  9 7  2 13 12 0  5 10;
         3 13  4  7 15  2  8 14 12 0  1 10  6 9 11  5;
         0 14  7 11 10  4 13  1  5 8 12  6  9 3  2 15;
        13  8 10  1  3 15  4  2 11 6  7 12  0 5 14  9];
sbox3 = ...                    % S3ºÐ
       [10  0  9 14 6  3 15  5  1 13 12  7 11  4  2  8;
        13  7  0  9 3  4  6 10  2  8  5 14 12 11  5  1;
        13  6  4  9 8 15  3  0 11  1  2 12  5 10 14  7;
         1 10 13  0 6  9  8  7  4 15 14  3 11  5  2 12];
sbox4 = ...                    % S4ºÐ
       [ 7 13 14 3  0  6  9 10  1 2 8  5 11 12  4 15;
        13  8 11 5  6  5  0  3  4 7 2 12  1 10 14  9;
        10  6  9 0 12 11  7 13 15 1 3 14  5  2  8  4;
         3 15  0 6 10  1 13  8  9 4 5 11 12  7  2 14];
sbox5 = ...                    % S5ºÐ
       [ 2 12  4  1  7 10 11  6  8  5  3 15 13 0 14  9;
        14 11  2 12  4  7 13  1  5  0 15 10  3 9  8  6;
         4  2  1 11 10 13  7  8 15  9 12  5  6 3  0 14;
         1  8 12  7  1 14  2 13  6 15  0  9 10 4  5  3];
sbox6 = ...                    % S6ºÐ
       [12  1 10 15 9  2  6  8  0 13  3  4 14  7  5 11;
```

```
 10 15  4  2 7 12  9  5  6  1 13 14  0 11  3  8;
        9 14 15  5 2  8 12  3  7  0  4 10  1 13 11  6;
        4  3  2 12 9  5 15 10 11 14  1  7  6  0  8 13];
sbox7 = ...                        % S7ºÐ
      [ 4 11  2 14 15 0  8 13  3 12 9  7  5 10 6  1;
       13  0 11  7  4 9  1 10 14  3 5 12  2 15 8  6;
        1  4 11 13 12 3  7 14 10 15 6  8  0  5 9  2;
        6 11 13  8  1 4 10  7  9  5 0 15 14  2 3 12];
sbox8 = ...                        % S8ºÐ
      [13  2  8 4  6 15 11  1 10  9  3 14  5  0 12  7;
        1 15 13 8 10  3  7  4 12  5  6 11  0 14  9  2;
        7 11  4 1  9 12 14  2  0  6 10 13 15  3  5  8;
        2  1 14 7  4 10  8 13 15 12  9  0  3  5  6 11];
sbox = [sbox1 sbox2 sbox3 sbox4 sbox5 sbox6 sbox7 sbox8];
sboxout = [ ];
for i = 0:7
    sboxin(i+1,1:6) = si(i*6+1:(i+1)*6);
    rind = bin2dec([sboxin(i+1,1),sboxin(i+1,6)])+1;
    nind = bin2dec(sboxin(i+1,2:5))+1+i*16;
    sboxout = [sboxout dec2bin(sbox(rind,nind),4)];
end
so = sboxout;
```

Decryption:

```
function decrypt = decrypt()

global edit1 edit2 edit3 pass1 pass2 data_encrypt
global irt endrt ert iprt pt

data = data_encrypt;
data_length = length(data);
adds = 8-mod(data_length,8);
if adds ~= 0
    for i = 1:adds
        data(data_length+i) = 0;
    end
    data_length = data_length+adds;
end

key = get(pass2,'String');
keylength = length(key);
keytemp0 = uint8(key);
keys = [149 57 208 147 21 183 27];
```

```
%%%%%%%%%%%--» ---%%%%%%%%%%%

if keylength < 7

   for i = keylength+1:7

       keytemp0(i) = keys(i);

   end

end

for i=1:7

      keyusetemp(i) = keytemp0(i);

end

keyuse = char(keyusetemp);


%%%%%%%%%%%%---  ---%%%%%%%%%%%%

pwb = str2bin(keyuse,7,2);

pwb = rebit(pwb,iprt);

ki = gerkey(pwb);

ki=flipud(ki);


%%%%%%%%%%%%%------%%%%%%%%%%%%%

times = (data_length-mod(data_length,8))/8;

for i = 0:times-1

    for j = 1:8

        tempdata(j) = data(8*i+j);
```

```
end
    encrydata = char(tempdata);
    encrydata = des(encrydata,ki);
    for k = 0:7
        temp = encrydata(k*8+1:(k+1)*8);
        data(i*8+k+1,1) = bin2dec(temp);
    end
end

data_decrypt = char(data)';
set(edit3,'string',data_decrypt)

%%%%%%%%%%%%%%%%%%%%%%%%--------------------
%%%%%%%%%%%%%%%%%%%%%%%%
function bin = str2bin(str,k,flag)

l = length(str);
bin = [];
temp = [];

for x = 1:l
     temp = [temp,dec2bin(str(x),8)];
```

```
end
if flag ~= 0
    n = ceil(l*8/k);
    rb = mod(l*8,k);
    sb = 0;
    if rb ~= 0
    sb = k-rb;
    for i = 1:sb
        z(i) = '0';
    end
    temp = [temp z];
    end
    for x = 0:n-1
      temp1 = temp(x*k+1:(x+1)*k);
      lone = length(find(temp1 == '1'));
      if flag == 1
          if mod(lone,2) == 0
              opb = '1';
          else
              opb = '0';
          end
      else if flag == 2
```

```
   if mod(lone,2) == 0
               opb = '0';
            else
               opb = '1';
            end
         end
      end
     temp1 = [temp1 opb];
     bin = [bin temp1];
   end
else
      bin = temp;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%----------
%%%%%%%%%%%%%%%%%%%%%%%%%%
function so = rebit(si,k)
lk=length(k);
for i=1:lk
   so(i)=si(k(i));
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%-------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ki=gerkey(k)

mt = ...

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

     1 1 2 2 2 2 2 2 1  2  2  2  2  2  2  1];

rt = ...

    [14 17 11 24  1  5  3 28 ...

     15  6 21 10 23 19 12  4 ...

     26  8 16  7 27 20 13  2 ...

     41 52 31 37 47 55 30 40 ...

     51 45 33 48 44 49 39 56 ...

     34 53 46 42 50 36 29 32];

kl = k(1:28);

kr = k(29:56);

for i = mt(1,1):mt(1,16)

    kl = mr(kl,mt(2,i));

    kr = mr(kr,mt(2,i));

    k = [kl kr];

    for j = 1:48

        ki(i,j) = k(fix(rt(j)));

    end
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%-----------                              -------------
%%%%%%%%%%%%%%%%%%%%%%%
function nk = mr(k,n)
l = length(k);
k1 = k(n+1:l);
k2 = k(1:n);
nk = [k1 k2];

%%%%%%%%%%%%%%%%%%%%%%%%%---------------
%%%%%%%%%%%%%%%%%%%%%%%%
function ef = des(pf,ki)

global irt endrt ert iprt pt
pfb = str2bin(pf,0,0);
pfb = rebit(pfb,irt);
lpfb = pfb(1:32);
rpfb = pfb(33:64);

for i = 0:15
    templ = lpfb;
    lpfb = rpfb;
```

```
 tempr = rebit(rpfb,ert);
    rpfb = char(xor(ki(i+1,:)-48,tempr-48)+48);
    rpfb = sbox(rpfb);
    rpfb = rebit(rpfb,pt);
    rpfb = char(xor(templ-48,rpfb-48)+48);
end
pfb = [rpfb lpfb];
pfb = rebit(pfb,endrt);
ef = pfb;

%%%%%%%%%%%%%%%%%%%%%%%%%------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%
function so = sbox(si)

sbox1 = ...                              % S1ºÐ
       [14  4 13 1  2 15 11  8  3 10  6 12  5  9 0  7;
         0 15  7 4 14  2 13  1 10  6 12 11  9  5 3  8;

  4  1 14 8 13  6  2 11 15 12  9  7  3 10 5  0;
        15 12  8 2  4  9  1  7  5 11  3 14 10  0 6 13];
sbox2 = ...                              % S2ºÐ
       [15  1  8 14  6 11  3  4  9 7  2 13 12 0  5 10;
    3 13  4  7 15  2  8 14 12 0  1 10  6 9 11  5;
         0 14  7 11 10  4 13  1  5 8 12  6  9 3  2 15;
        13  8 10  1  3 15  4  2 11 6  7 12  0 5 14  9];
sbox3 = ...                       % S3ºÐ
       [10  0  9 14 6  3 15  5  1 13 12  7 11  4  2  8;
        13  7  0  9 3  4  6 10  2  8  5 14 12 11  5  1;
        13  6  4  9 8 15  3  0 11  1  2 12  5 10 14  7;
         1 10 13  0 6  9  8  7  4 15 14  3 11  5  2 12];
sbox4 = ...                       % S4ºÐ
       [ 7 13 14 3  0  6  9 10  1 2 8  5 11 12  4 15;
        13  8 11 5  6  5  0  3  4 7 2 12  1 10 14  9;
        10  6  9 0 12 11  7 13 15 1 3 14  5  2  8  4;
         3 15  0 6 10  1 13  8  9 4 5 11 12  7  2 14];
sbox5 = ...                       % S5ºÐ
       [ 2 12  4  1  7 10 11  6  8  5  3 15 13 0 14  9;
        14 11  2 12  4  7 13  1  5  0 15 10  3 9  8  6;
         4  2  1 11 10 13  7  8 15  9 12  5  6 3  0 14;
         1  8 12  7  1 14  2 13  6 15  0  9 10 4  5  3];

sbox6 = ...                       % S6ºÐ
       [12  1 10 15 9  2  6  8  0 13  3  4 14  7  5 11;
        10 15  4  2 7 12  9  5  6  1 13 14  0 11  3  8;
```

```
 9 14 15  5 2  8 12  3  7  0  4 10  1 13 11  6;
        4  3  2 12 9  5 15 10 11 14  1  7  6  0  8 13];
sbox7 = ...                          % S7ºÐ
      [ 4 11  2 14 15 0  8 13  3 12 9  7  5 10 6  1;
       13  0 11  7  4 9  1 10 14  3 5 12  2 15 8  6;
        1  4 11 13 12 3  7 14 10 15 6  8  0  5 9  2;
        6 11 13  8  1 4 10  7  9  5 0 15 14  2 3 12];
sbox8 = ...                          % S8ºÐ
      [13  2  8 4  6 15 11  1 10  9  3 14  5  0 12  7;
        1 15 13 8 10  3  7  4 12  5  6 11  0 14  9  2;
        7 11  4 1  9 12 14  2  0  6 10 13 15  3  5  8;
        2  1 14 7  4 10  8 13 15 12  9  0  3  5  6 11];
sbox = [sbox1 sbox2 sbox3 sbox4 sbox5 sbox6 sbox7 sbox8];
sboxout = [ ];
for i = 0:7
    sboxin(i+1,1:6) = si(i*6+1:(i+1)*6);
    rind = bin2dec([sboxin(i+1,1),sboxin(i+1,6)])+1;
    nind = bin2dec(sboxin(i+1,2:5))+1+i*16;
    sboxout = [sboxout dec2bin(sbox(rind,nind),4)];

end
so = sboxout;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Steganography:

```
function varargout = Main_file(varargin)
 gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Main_file_OpeningFcn, ...
                   'gui_OutputFcn',  @Main_file_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});

end
```

```
function Main_file_OpeningFcn(hObject, eventdata, handles, varargin)


handles.output = hObject;

guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.

function varargout = Main_file_OutputFcn(hObject, eventdata, handles)


varargout{1} = handles.output;

function edit1_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.

function edit1_CreateFcn(hObject, eventdata, handles)


if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))

   set(hObject,'BackgroundColor','white');

end
```

```
% --- Executes on button press in pushbutton1.

function pushbutton1_Callback(hObject, eventdata, handles)


set(handles.figure1, 'pointer', 'watch')

drawnow;


[file path]=uigetfile('*.wav');


[y fs]=audioread(file);

sound(y(:,1),fs)

axes(handles.axes1)

specgram(y(:,1),1024,fs);

handles.y=y;

handles.fs=fs;


set(handles.figure1, 'pointer', 'arrow')


guidata(hObject, handles);

function edit2_Callback(hObject, eventdata, handles)
```

```
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)


if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit3_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)


if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit4_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
```

```
if, ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');

end
function edit5_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))

 set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)

set(handles.figure1, 'pointer', 'watch')
drawnow;
tic;
```

```
y=handles.y;
fs=handles.fs;
cover=y(:,1);
cover=cover';

%////////////////////Encryption/////////////////////////////
edit1=handles.edit1;
pass1=handles.edit3;
txt1=encrypt();
%//////////////////////////////////////////////////
% Sec Text handling
[textStr]=get(txt1,'string');
binVector = str2bin(textStr);
binVector=binVector';

% Sec Key handling
len1=length(binVector);
[key]=get(handles.edit3,'string');
key=str2num(key);
[sec_key]=secret_key_generation_txt(len1,key);

% Encryption of Sec Msg
```

```
binVector=xor(binVector,sec_key);

% Wavelet Decomposition

level=1;


%level=str2num(level);

[ca cd]=lwt(cover,'haar',level);

len=length(cd);


% Embedding Process

for i=1:len1

    if(binVector(1,i)==0)

        new(1,i)=cd(1,i)+0.001;

    else

        new(1,i)=cd(1,i);

    end


end


cd1=[new cd(len1+1:end)];

len2=length(cd1);


stego = ilwt(ca,cd1,'haar',level);
```

```
sound(stego,fs)

axes(handles.axes2)

specgram(stego,1024,fs);


handles.stego=stego;


handles.cd=cd;

handles.len1=len1;

handles.key=key;

t=toc;

set(handles.edit10,'string',t)


set(handles.figure1, 'pointer', 'arrow')


global psnr mse len SNR


guidata(hObject, handles);


% --- Executes on button press in pushbutton3.

function pushbutton3_Callback(hObject, eventdata, handles)

stego=handles.stego;

fs=handles.fs;
```

```
stego=stego';

[file path]=uiputfile('*.wav');

audiowrite([path file],stego,fs)


% --- Executes on button press in pushbutton4.

function pushbutton4_Callback(hObject, eventdata, handles)


axes(handles.axes1);cla

axes(handles.axes2);cla

a=[];

set(handles.edit1,'string',a)

set(handles.edit2,'string',a)

set(handles.edit3,'string',a)

set(handles.edit4,'string',a)

set(handles.edit5,'string',a)


set(handles.edit8,'string',a)

set(handles.edit9,'string',a)

set(handles.edit10,'string',a)


function edit6_Callback(hObject, eventdata, handles)
```

```
% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
          end
function edit7_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)

set(handles.figure1, 'pointer', 'watch')
drawnow;
```

```
[file path]=uigetfile('*.wav');
[y fs]=audioread(file);

sound(y(:,1),fs)
axes(handles.axes3)
specgram(y(:,1),1024,fs);
handles.y=y;
handles.fs=fs;
set(handles.figure1, 'pointer', 'arrow')

guidata(hObject, handles);

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)

set(handles.figure1, 'pointer', 'watch')
drawnow;

global psnr mse len SNR
%set(handles.edit14,'string',len)
%set(handles.edit15,'string',SNR)
%set(handles.edit16,'string',psnr)
```

```
%set(handles.edit17,'string',mse)
y=handles.y;


fs=handles.fs;
cd=handles.cd;
len1=handles.len1;
key=handles.key;
tic;


key1=get(handles.edit7,'string');
key1=str2num(key1);



if(key~=key1)% checking if key is not equal
   msgbox('Wrong key enter correct key','box')
   error('Wrong key enter correct key')
end


%level=get(handles.edit6,'string');
level=1;
%level=str2num(level);
```

```
y=y';

[ca2 cd2]=lwt(y(1,:),'haar',level);



for i=1:len1

    if(cd2(1,i)~=cd(1,i))

       recov_sec(1,i)=0;

    else

       recov_sec(1,i)=1;

    end

end

[sec_key]=secret_key_generation_txt(len1,key1);



sec=xor(recov_sec,sec_key);

recov_sec=sec';


textStr = bin2str(recov_sec);

%/////////////////////////////////Decryption/////////////////////////////


pass2=key1;

data_encrypt=textStr;
```

```
txt2=decrypt()
%//////////////////////////////////////////////////////////////////////////


set(handles.figure1, 'pointer', 'arrow')
t=toc;
set(handles.edit13,'string',t)


set(handles.text8,'string',txt2)
handles.textStr=txt2;
guidata(hObject, handles);


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)


textStr=handles.textStr;
fidenc=fopen('Recovered Secrete Text.txt','w');
fprintf(fidenc,'%s',textStr)
fclose(fidenc)




% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
```

```
axes(handles.axes3);cla

a=[];


set(handles.edit6,'string',a)

set(handles.edit7,'string',a)

set(handles.text8,'string',a)

set(handles.edit13,'string',a)

set(handles.edit14,'string',a)

set(handles.edit15,'string',a)

set(handles.edit16,'string',a)

set(handles.edit17,'string',a)

clc;

clear all;


function edit8_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.

function edit8_CreateFcn(hObject, eventdata, handles)


if ispc && isequal(get(hObject,'BackgroundColor'),

get(0,'defaultUicontrolBackgroundColor'))

    set(hObject,'BackgroundColor','white');
```

```
end

function edit9_Callback(hObject, eventdata, handles)



% --- Executes during object creation, after setting all properties.

function edit9_CreateFcn(hObject, eventdata, handles)


if ispc && isequal(get(hObject,'BackgroundColor'),

get(0,'defaultUicontrolBackgroundColor'))

    set(hObject,'BackgroundColor','white');

end


function edit10_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.

function edit10_CreateFcn(hObject, eventdata, handles)


if ispc && isequal(get(hObject,'BackgroundColor'),

get(0,'defaultUicontrolBackgroundColor'))

    set(hObject,'BackgroundColor','white');

end
```

```
function edit13_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.


function edit13_CreateFcn(hObject, eventdata, handles)


if ispc && isequal(get(hObject,'BackgroundColor'),

get(0,'defaultUicontrolBackgroundColor'))

    set(hObject,'BackgroundColor','white');

end


function edit14_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.

function edit14_CreateFcn(hObject, eventdata, handles)


if ispc && isequal(get(hObject,'BackgroundColor'),

get(0,'defaultUicontrolBackgroundColor'))

    set(hObject,'BackgroundColor','white');

end


function edit15_Callback(hObject, eventdata, handles)
```

```
% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)



if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit16_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit17_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
```

```
[file path]=uigetfile('*.wav');

[y fs]=audioread(file);


sound(y(:,1),fs)

axes(handles.axes3)

specgram(y(:,1),1024,fs);

handles.y=y;

handles.fs=fs;

set(handles.figure1, 'pointer', 'arrow')


guidata(hObject, handles);


% --- Executes on button press in pushbutton6.

function pushbutton6_Callback(hObject, eventdata, handles)


set(handles.figure1, 'pointer', 'watch')

drawnow;


global psnr mse len SNR

%set(handles.edit14,'string',len)

%set(handles.edit15,'string',SNR)

%set(handles.edit16,'string',psnr)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```