CHAPTER 1: INTRODUCTION

1.1 PROJECT OVERVIEW

Steganography[1] and cryptography[3] are the two main methods for information hiding and security today. Information hiding (steganography) aims at hiding the very existence of the secret message itself. It is done by hiding the secret data in any innocuous medium, so that is it not apparent that there is any data hidden in the cover medium. Cryptography is the technique which scrambles the secret information itself so that it cannot be understood without unscrambling it. A cipher is usually applied in case of cryptography. Multimedia files, such as images, audio and video are widely used today. Images are a good medium for hiding data. The more detailed the image, the lesser constraints there are on how much data it can hold without it becoming conspicuous. There are several tools available for hiding data in audio files, but the large size of meaningful audio files then made it less popular than image files as a steganographic medium. But both these mediums have a lot less storage capacity as compared to videos, since videos can be taken as a collection of frames (images) and audio.

In this paper, we have discussed a new method of steganography along with cryptography which will be applied with videos as the cover medium. Multimedia steganography is one of the most recent and secure forms of steganography. Visual steganography is the most widely practiced form of steganography. It started with concealing messages within the lowest bits of noisy images or sound files. We have performed steganography on video files and hidden the message in an encrypted format, thus achieving a multiple cryptographic system. The most commonly used technique is Least Significant Bit steganography (LSB steganography). But instead of traditional LSB encoding, we have used a modified encoding technique which first transforms the video using a Lazy Lifting Wavelet transform and then applies LSB[1] in the subbands of the video that we have gotten.

1.2 OVERVIEW

Steganography[1] is an art of hiding the secret information inside digitally covered information. The hidden message can be text, image, speech or even video and the cover can be chosen from either an image or a video. Here in this paper, the hidden message is text and it is implemented over video file. The traditional well known method uses image as cover which has the limitation of embedding dimension. So, cover should be a video to overcome the limitation of embedding dimension. Nowadays, the use of a video based steganography is common and numbers of steganalysis tools are available to check whether the video is stego-video or not. Most of the tools are checking for information hided by LSB, DCT, Frequency Domain Analysis etc and finds whether the video has hidden or secret data or not. In this paper, LSB and Random Byte Hiding techniques are implemented and MATLAB based implementation is done to simulate the results.

CHAPTER 2: LITERATURE REVIEW

Steganography (Listen) is the art or practice of concealing a message, image, or file within another message, image, or file. The word steganography combines the Ancient Greek words steganos , meaning "covered, concealed, or protected", and graphei (γραφή) meaning "writing". The first recorded use of the term was in 1499 by Johannes Trithemius in his Steganographia, a treatise on cryptography and steganography, disguised as a book on magic. Generally, the hidden messages will appear to be (or be part of) something else: images, articles, shopping lists, or some other cover text. For example, the hidden message may be in invisible ink between the visible lines of a private letter. Some implementations of steganography which lack a shared secret are forms of security through obscurity, whereas key-dependent steganographic schemes adhere to Kerckhoffs's principle.

2.1 TECHNIQUES

2.1.1 Physical

Steganography has been widely used, including in recent historical times and the present day. Known examples include:

Hidden messages within wax tablets — in ancient Greece, people wrote messages on the wood, then covered it with wax upon which an innocent covering message was written.
Hidden messages on messenger's body — also used in ancient Greece. Herodotus tells the story of a message tattooed on the shaved head of a slave of Histiaeus, hidden by the hair that afterwards grew over it, and exposed by shaving the head again. The message allegedly carried a warning to Greece about Persian invasion plans. This method has obvious drawbacks, such as delayed transmission while waiting for the slave's hair to grow, and the restrictions on the number and size of messages that can be encoded on one person's scalp.
In the early days of the printing press, it was common to mix different typefaces on a printed page due to the printer not having enough copies of some letters otherwise. Because of this, a message could be hidden using 2 (or more) different typefaces, such as normal or italic.
During World War II, the French Resistance sent some messages written on the backs of couriers using invisible ink.

Hidden messages on paper written in secret inks, under other messages or on the blank parts of other messages.

Messages written in Morse code on knitting yarn and then knitted into a piece of clothing worn by a courier. Jeremiah Denton repeatedly blinked his eyes in Morse Code during the 1966 televised press conference that he was forced into as an American POW by his North Vietnamese captors, spelling out the word, "T-O-R-T-U-R-E". This confirmed for the first time to the U.S. Military (naval intelligence) and Americans that American POWs were being tortured in North Vietnam.

2.1.2 Digital

Image of a tree with a steganographically hidden image. The hidden image is revealed by removing all but the two least significant bits of each color component and a subsequent normalization. The hidden image is shown below.

Modern steganography entered the world in 1985 with the advent of the personal computers being applied to classical steganography problems.[5] Development following that was very slow, but has since taken off, going by the large number of steganography software available:

Concealing messages within the lowest bits of noisy images or sound files.Concealing data within encrypted data or within random data. The data to be concealed are first encrypted before being used to overwrite part of a much larger block of encrypted data or a block of random data (an unbreakable cipher like the one-time pad generates ciphertexts that look perfectly random if one does not have the private key).

Chaffing and winnowing.

Mimic functions convert one file to have the statistical profile of another. This can thwart statistical methods that help brute-force attacks identify the right solution in a ciphertext-only attack.

Concealed messages in tampered executable files, exploiting redundancy in the targeted instruction set.

Pictures embedded in video material (optionally played at slower or faster speed).

Injecting imperceptible delays to packets sent over the network from the keyboard. Delays in keypresses in some applications (telnet or remote desktop software) can mean a delay in packets, and the delays in the packets can be used to encode data.
Changing the order of elements in a set.
Content-Aware Steganography hides information in the semantics a human user assigns to a datagram. These systems offer security against a non-human adversary/warden.

2.1.3 Network

All information hiding techniques that may be used to exchange steganograms in telecommunication networks can be classified under the general term of network steganography. This nomenclature was originally introduced by Krzysztof Szczypiorski in 2003.[8] Contrary to the typical steganographic methods which utilize digital media (images, audio and video files) as a cover for hidden data, network steganography utilizes communication protocols' control elements and their basic intrinsic functionality. As a result, such methods are harder to detect and eliminate.[9]

Typical network steganography methods involve modification of the properties of a single network protocol. Such modification can be applied to the PDU (Protocol Data Unit),[10][11][12] to the time relations between the exchanged PDUs,[13] or both (hybrid methods).[14]

Moreover, it is feasible to utilize the relation between two or more different network protocols to enable secret communication. These applications fall under the term inter-protocol steganography.[15]

Network steganography covers a broad spectrum of techniques, which include, among others:

Steganophony - the concealment of messages in Voice-over-IP conversations, e.g. the employment of delayed or corrupted packets that would normally be ignored by the receiver (this method is called LACK - Lost Audio Packets Steganography), or, alternatively, hiding information in unused header fields.[16]

WLAN Steganography – the utilization of methods that may be exercised to transmit steganograms in Wireless Local Area Networks. A practical example of WLAN Steganography is the HICCUPS system (Hidden Communication System for Corrupted Networks)[17]

2.1.4 Printed

Digital steganography output may be in the form of printed documents. A message, the plaintext, may be first encrypted by traditional means, producing a ciphertext. Then, an innocuous covertext is modified in some way so as to contain the ciphertext, resulting in the stegotext. For example, the letter size, spacing, typeface, or other characteristics of a covertext can be manipulated to carry the hidden message. Only a recipient who knows the technique used can recover the message and then decrypt it. Francis Bacon developed Bacon's cipher as such a technique.

CHAPTER 3: SYSTEM ANALYSIS

3.1 Existing System

In the Existing system of video steganography LSB algorithm[15] was used in the frames of video. In LSB algorithm, the message bit is taken from the message byte and then that particular bit will be embedded inside the least significant bit of an image or video or audio file. This is done because

l. The message embedded in the least significant bit of an image file will not draw the suspicion of the hacker as the minute difference that would be made in the pixel value of the image file will not be perceived by the normal naked human eye.
2. The message that will be embedded in the LSB of an audio file will not create suspicion to the hacker as that change would not be perceived by the human ear.
3. The same concept works out evenwith video file.
4. This same algorithm can also be used for digital Watermarking.

The file where we actually embed the message is known as cover file. The usual files that are selected as cover file are image, audio, video, text data. But using text data as a cover file will encipher the message than hiding the message. So logically this cannot be told as steganography.

Implementing Steganography in LSB:-

Steganography can be implemented using LSB algorithm in two ways:
(1) Using keys
(2) Without using keys.
Now let us see a small example of how the message will be stored. Consider 8 bytes of a cover file(say bitmap image).Let the 8 bytes be
11111010
10101010
11110000
01010110
10010010

01101110

10000100

11000000

Consider a message byte 11111111.we should embed all these 8 1s into the cover file. Since we are going to use the LSB algorithm we need 8 bytes from cover file to embed a byte of message file. This is done by, covering each and every bit of the message file by the LSB of the cover file.

The final answer would be:

1111101<u>1</u>

1010101<u>1</u>

1111000<u>1</u>

0101011<u>1</u>

1001001<u>1</u>

0110111<u>1</u>

1000010<u>1</u>

1100000<u>1</u>

The 1's with underlining denotes the message bit embedded in the cover file.

In case of LSB without using key , it is nothing but embedding all the message bits in the successive bytes.

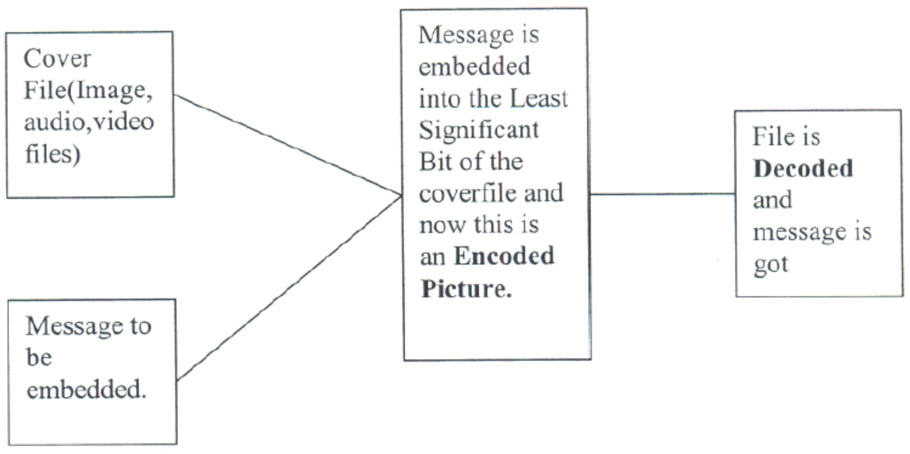

Cover File(Image, audio,video files)
Message to be embedded.
Message is embedded into the Least Significant Bit of the coverfile and now this is an **Encoded Picture.**
File is **Decoded** and message is got

Fig 1: Steganography

3.1.1 Drawbacks of the Existing Systems:

Though LSB has the advantages like easy implementations, it also has its unavoidable drawbacks. Once if the hacker is aware of the presence of message in the cover file, he will first use the LSB algorithm and so it gives very less security to the message. More over as the name refers only LSB[12] of the cover byte is used and other bits are not used as it may create changes in the cover file and it would draw suspicion and tends to easy steganalysis. So if a confidential message is expected to send, then it should guarantee full security and it should make the hacker to consume his/her maximum amount of time hack the content.

3.2 Proposed Technique:

The proposed model is an extension of the previous model. The proposed techniques makes use of the LSB steganography combined with Lazy Wavelets. The lazy wavelet transform is applied to the visual frames, and the data is stored in the coefficients of the visual component. It first transforms the video using a Lazy Lifting Wavelet transform and then applies LSB in the subbands of the video that we have gotten. The length up to which it is stored is hidden using LSB in the audio component. Experimental results show that the proposed technique does not affect the higher and lower ends of the frequency distribution of the signal. Moreover, it has a high payload capacity and low computational requirements. It enhances the security level to new heights.

3.2.1 The Visual Cryptographic Steganography Model

In this model, we will encrypt the message using the symmetric key algorithm, and after that hide the data into a video file, which will act like our cover. At the receiving side, we will first extract the hidden data from the received video file, and then decrypt it using the shared key that we already know.

3.2.2 Hiding Procedure

A video file is usually composed of multiple frames. This method uses some frames (or images) of the video to hide the secret message. The secret data is hidden in sequential frames. Each frame is treated as a different image and an image steganography method is applied to them. We use the 2D - Lazy Wavelet Transform[2] on each frame to get four

sub-bands. The data is then hidden in these four subbands using LSB to hide 3 bits in each element of the subband. The length of the data stream which is encoded into the video is stored in the audio using simple LSB.
The proposed method consists of the following phases:

3.2.3 Encrypting the Given Secret Data File

Given the secure data that we want to send, we first apply encryption on it so that the data is converted to a cipher text and is not readable. We have applied the RSA algorithm on our secret data. This provides reasonable amount of security with good encrypting speed performance .

3.2.4 Converting Given Encrypted Cipher Data into a Stream of Bits

Since we will be dealing with the individual bits of the encrypted file, which we will hide inside the cover video, we will convert the given file into a series of bits. We will read the file character by character (since after encryption, the file has been converted to only ASCII characters), and then break the eight bit characters into strings of bits. This bit stream can now be encoded into the video.

3.2.5 Applying Lazy Wavelet Transform on the Frames of the Video

A video consists of frames,each of which can be considered as separate images. On these separate images, we apply image transformation techniques.We use wavelets to transform the given image in the spatial domain into the frequency domain. Values in the multimedia data are stored as integers, but many wavelet transforms return real values, which cause data loss when stored in a multimedia file and then retrieved. To overcome this, we use the Lazy Lifting Scheme[4], by applying an Integer Wavelet Transform. The lifting scheme calculates wavelet transforms in an efficient way, and can easily be converted to an integer transform. We can easily do this by adding some rounding operators.

3.2.6 Hiding Three bits in Transform Coefficients of the Four Sub-bands

Performing the Integer Wavelet Transform on the frames of the video will give us four sub-bands for each frame. We will hide the message in the least significant bits (LSB) of the transform coefficients.With three LSB’s to store the bits in each transform coefficient, we calculate PSNR.

3.2.7 Hiding the Total Length and the Number of Bits in the Last Frame in the Audio using LSB

Since we are storing the data sequentially in the frames, we will store maximum payload in all the frames, which will be fixed by the frame size. But in the last frame in which the message bits will be hidden, a fewer number of bits might be hidden. When we will decode our original message at the receiver end, we will need this number to extract the exact message. Also, the total number of frames in which the data is stored will need to be sent along with the message. Both these numbers will fit into a 4 byte unsigned integer. So we will store these numbers sequentially in the audio using the LSB method given in. The process of extracting the secret data from the steganographed video is just the reverse process of hiding the data in the video.

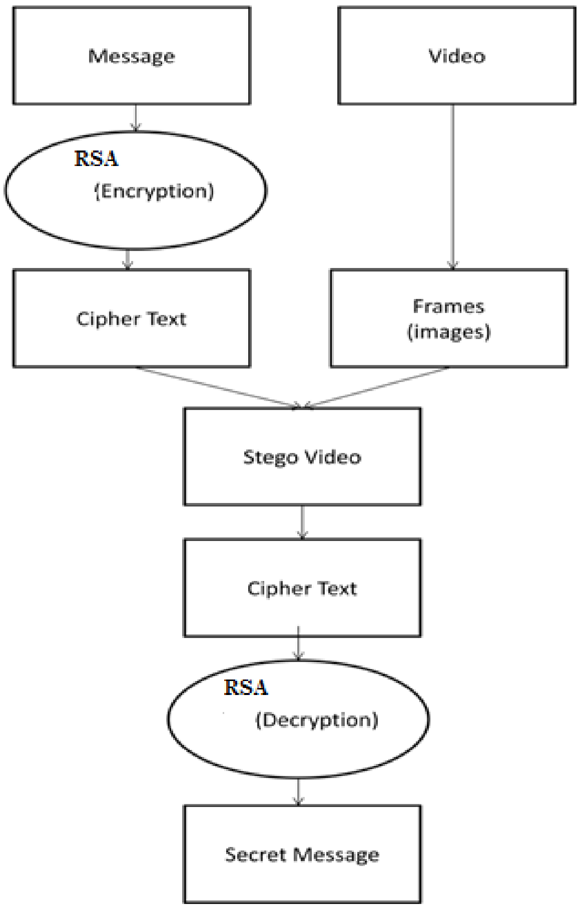


Fig 2: Procedure

3.3 Feasibilty Study

3.3.1 Economic Feasibility

The project is economical . It does not require any software other than MATLAB 2013.

Cost

- RAM: 2 GB $39.00
- Processor: Intel Core i3 or higher $ 140.00

Software Cost

- Windows Operating System 32/64 bit: $ 229.99
- Matlab 2013b 64 bit: $89

3.2.2 Operation Feasibility

The project fits with all the requirements of the proposed system.

- Current work practices and procedures are adequate to support the new system.
- The project involves moderate workforce.
- A team of business analyst, designers, developers, testers, quality engineer and maintenance team is enough to develop and handle project
- Small and medium enterprises will have economic advantages.
- The project assures high security therefore no loss of information.
- The system suggests a strong enterprise model.
- The project does not violate any legal aspect.
- End-users feel confident about the use of such technology.

3.3.3 Technical Feasibility

The project is technically feasible.

- The project makes use of hardware and software specifications as mentioned in next section that are feasibility and easily available at affordable prices.
- The projects works well under given specification with no additional implementations and infrastructure.
- The project can be extended as and when required and can be modified according to the needs of the organisation.

CHAPTER 4: SYSTEM SPECIFICATIONS

4.1 Hardware Requirements

- The minimum hardware requirements for the project are:
- Core I3 2.0 GHz Processor
- Minimum of 2GB RAM
- Minimum of 20 GB HDD
- VGA Display with 640 x 480 screen in High/True color Display mode
- 1.44 MB FDD
- 52X CD ROM Drive
- Monitor
- Keyboard
- Mouse

4.2 Software Requirements

- MATLAB 2013
- Operating System: Windows 7

CHAPTER 5: SOFTWARE DESCRIPTION

MATLAB FEATURES

MATLAB (**mat**rix **lab**oratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems.

In 2004, MATLAB had around one million users across industry and academia. MATLAB users come from various backgrounds of engineering, science, and economics. MATLAB is widely used in academic and research institutions as well as industrial enterprises.

CHAPTER 6: PROJECT DESCRIPTION

6.1 Problem Definition

The problem with the traditional LSB method was that once if the hacker is aware of the presence of message in the cover file, he will first use the LSB algorithm and so it gives very less security to the message. More over as the name refers only LSB of the cover byte is used and other bits are not used as it may create changes in the cover file and it would draw suspicion and tends to easy steganalysis. So if a confidential message is expected to send, then it should guarantee full security.

6.2 Overview of the Project

Visual steganography is the most widely practiced form of steganography. It started with concealing messages within the lowest bits of noisy images or sound files. We have performed steganography on video files and hidden the message in an encrypted format, thus achieving a multiple cryptographic system. The most commonly used technique is Least Significant Bit steganography (LSB steganography). But instead of traditional LSB encoding, we have used a modified encoding technique which first transforms the video using a Lazy Lifting Wavelet transform and then applies LSB in the sub-bands of the video that has been obtained. The proposed approach to video steganography utilizes the visual as well as the audio component. The lazy wavelet transform[4] is applied to the visual frames, and the data is stored in the coefficients of the visual component. The length up to which it is stored is hidden using LSB in the audio component. Experimental results show that the proposed technique does not affect the higher and lower ends of the frequency distribution of the signal. Moreover, it has a high payload capacity and low computational requirements.

6.3 Module Description

6.3.1 Encryption Module For The Text Message

RSA[7] is one of the first practicable public-key cryptosystems and is widely used for secure data transmission. In such acryptosystem, the encryption key is public and differs from the decryption key which is kept secret. In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the factoring problem. RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described the algorithm in 1977.

A user of RSA creates and then publishes the product of two large prime numbers, along with an auxiliary value, as their public key. The prime factors must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime factors can feasibly decode the message.

Key Generation

RSA involves a *public key* and a *private key*. The public key can be known by everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key. The keys for the RSA algorithm are generated the following way:

1. Choose two distinct prime numbers p and q.
   - For security purposes, the integers p and q should be chosen at random, and should be of similar bit-length. Prime integers can be efficiently found using aprimality test.
2. Compute $n = pq$.
   - n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
3. Compute $\varphi(n) = \varphi(p)\varphi(q) = (p - 1)(q - 1) = n - (p + q -1)$, where φ is Euler's totient function.

4. Choose an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$; i.e., e and $\varphi(n)$ are coprime.
5. Determine d as $d \equiv e^{-1} \pmod{\varphi(n)}$; i.e., d is the multiplicative inverse of e (modulo $\varphi(n)$).

- This is more clearly stated as: solve for d given $d \cdot e \equiv 1 \pmod{\varphi(n)}$
- This is often computed using the extended Euclidean algorithm. Using the pseudocode in the *Modular integers* section, inputs a and n correspond to e and $\varphi(n)$, respectively.
- d is kept as the private key exponent.

The *public key* consists of the modulus n and the public (or encryption) exponent e. The *private key* consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and $\varphi(n)$ must also be kept secret.

Encryption

Alice transmits her public key (n, e) to Bob and keeps the private key secret. Bob then wishes to send message M to Alice.

He first turns M into an integer m, such that $0 \le m < n$ by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext $c$corresponding to

$$c \equiv m^e \pmod{n}$$

This can be done quickly using the method of exponentiation by squaring. Bob then transmits c to Alice.

Note that at least nine values of m will yield a ciphertext c equal to m,[note 1] but this is very unlikely to occur in practice.

Decryption

Alice can recover m from c by using her private key exponent d via computing

$$m \equiv c^d \pmod{n}$$

Given m, she can recover the original message M by reversing the padding scheme.

6.3.2 Hiding Module

In this module, the encrypted image or encrypted text is hidden inside the video using lazy wavelet based transform. The resulting video is called stego video. We deal with the individual bits of the encrypted file, which hide inside the cover video, we convert the given file into a series of bits. We read the file character by character (since after encryption, the file has been converted to only ASCII characters), and then break the eight bit characters into strings of bits. This video consists of frames, each of which can be considered as separate images. On these separate images, we apply image transformation techniques. We use wavelets to transform the given image in the spatial domain into the frequency domain. Values in the multimedia data are stored as integers, but many wavelet transforms return real values, which cause data loss when stored in a multimedia file and then retrieved. To overcome this, we use the Lazy Lifting Scheme, by applying an Integer Wavelet Transform. The lifting scheme calculates wavelet transforms in an efficient way, and can easily be converted to an integer transform. We can easily do this by adding some rounding operators. Bit stream can now be encoded into the video. Performing the Integer Wavelet Transform on the frames of the video gives us four sub-bands for each frame. We hide the message in the least significant bits (LSB) of the transform coefficients. With three LSB's to store the bits in each transform coefficient. Since we store the data sequentially in the frames, we store maximum payload in all the frames, which is fixed by the frame size. But in the last frame in which the message bits are to be hidden, a fewer number of bits might be hidden. When we decode our original message at the receiver end, we need this number to extract the exact message. Also, the total number of frames in which the data is stored needs to be sent along with the message. Both these numbers fit into a 4 byte unsigned integer. So we store these numbers sequentially in the audio using the LSB method given in.

6.3.2.1 Applying Lazy Wavelet Transform on the Frames of the Video

A video is comprised of many frames. On each frame we apply a image transformation technique. Wavelet transformation is use to convert the spatial domain into frequency domain but most of the wavelet techniques produce real values, which will result in data loss when is hide and retrieved. So to overcome this we use lazy wavelet scheme, by applying Integer Wavelet Transform which produces integer values. After applying Integer Wavelet Transform we get four subbands.

6.3.2.2 Hiding bits in the Four Sub-bands

For each subbands we find out RGB components, and now we start encoding data in RGB components in sequential order of each subbands of each frame using LSB technique

6.3.2.3 Hiding the Total Length and the Number of Bits in the Last Frame in the Audio using LSB

Since we are storing the data sequentially in the frames, we will store maximum payload in all the frames, which will be fixed by the frame size. But in the last frame in which the message bits will be hidden, a fewer number of bits might be hidden. When we will decode our original message at the receiver end, we will need this number to extract the exact message. Also, the total number of frames in which the data is stored will need to be sent along with the message. So we will store these numbers sequentially in the audio using the LSB method.

6.3.2.4 Algorithm for Information Hiding

Frame: I

Encrypted message: m

Step 1: Extract all frames from video

Step 2: Select 1st Frame I from Video

Step 3: Apply Lazy wavelet scheme to produce 4 subbands (cA cH cV cD).

Step 4: Extract RGB component from choosen sub band

Step 5: Select 1st pixel P from RGB Component

Step 6: Get 24 bit corresponding pixel value

Step 7: Perform $m_n \oplus P_n(k)$ where k=1,2,3 and n=1……n-1

Step 8: Repeat step 3, 4, 5 for all frames Step 9: Construct the video from all encoded frames.

Step 10: Transmit Video, secret Key through secure channel

6.3.2.5 Extraction

The process of extracting the secret data from the stegano-graphed video is just reverse process of hiding the data in video.

6.3.2.6 Algorithm for Information retrieval

Step 1: Extract all frames from video

Step 2: Select 1st Frame I from Video

Step 3: Apply Lazy wavelet scheme to produce 4 sub bands (cA cH cV cD).

Step 4: Extract RGB component from chosen subband

Step 5: Select 1st pixel P from RGB Component

Step 6: Get 24 bit corresponding pixel value

Step 7: Perform msg=$p_n(k)$ where k=1,2,3 and n=1….n-1

Step 8: Repeat step 3,4,5 for all frames and save message. Step 9: Apply AES decryption to convert cipher message to
plain text (we get the secret message)

6.4 DATA FLOW DIAGRAM

6.4.1 LEVEL 0

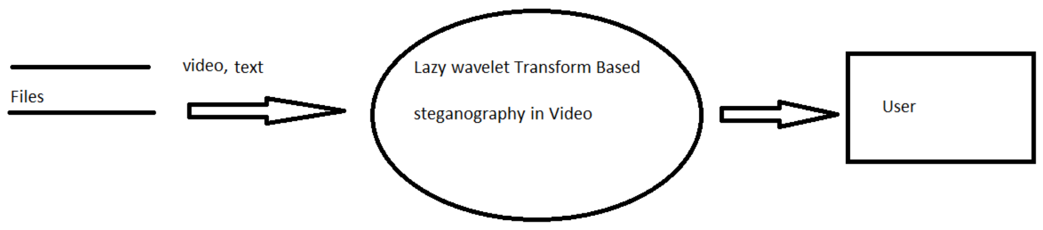


Fig 3: Level 0 DFD

6.4.2 LEVEL 1: HIDE MESSAGE

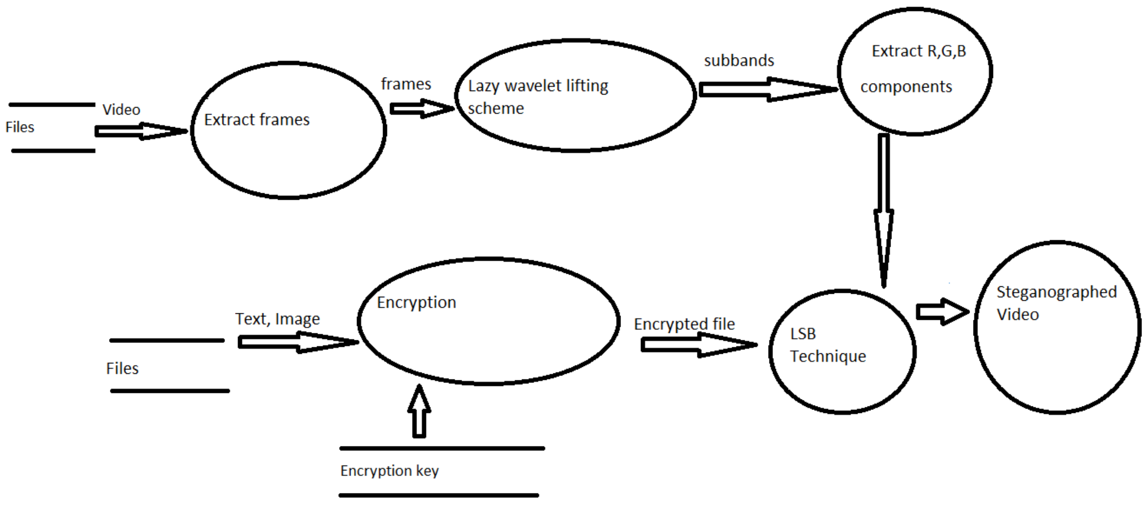


Fig 4: Level 1 DFD

6.4.3 LEVEL 1: EXTRACT MESSAGE

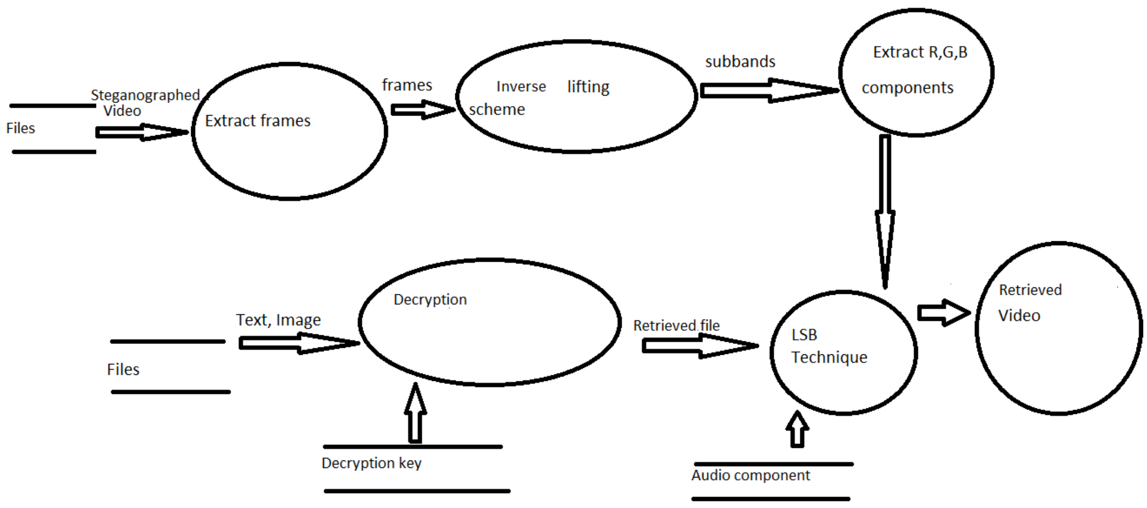


Fig 5: Level 1 DFD

CHAPTER 7: SYSTEM TESTING

7.1 UNIT TESTING

The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you expect. Each unit is tested separately before integrating them into modules to test the interfaces between modules. Unit testing has proven its value in that a large percentage of defects are identified during its use.

The most common approach to unit testing requires drivers and stubs to be written. The driver simulates a calling unit and the stub simulates a called unit. The investment of developer time in this activity sometimes results in demoting unit testing to a lower level of priority and that is almost always a mistake. Even though the drivers and stubs cost time and money, unit testing provides some undeniable advantages. It allows for automation of the testing process, reduces difficulties of discovering errors contained in more complex pieces of the application, and test coverage is often enhanced because attention is given to each unit.

7.1.1 RSA ENCRYPTION

Table 1

Input	**Public Key**	**Private Key**	**Expected Output**	**Observed Output**
Abcde	7,143	103,143	; ,d)	; ,d)
Qwerty	19,221	91,221	_fe)AO	_fe)AO
Q12u23	31,667	159,667	??T?T”	??T?T”
56glj	7,15	7,15	<:p]-	<:p]-
hwi12	17,253	13,253	|04b?	|044b?

7.1.2 RSA DECRYPTION

Table 2

Input	**Public Key**	**Private Key**	**Expected Output**	**Observed Output**
; ,d)	7,143	103,143	abcde	abcde
_fe)AO	19,221	91,221	qwerty	qwerty
??T?T"	31,667	159,667	Q12u23	Q12u23
<:p]-	7,15	7,15	56glj	56glj
|04b?	17,253	13,253	hwi12	hwi12

CHAPTER 8: RESULTS AND DISCUSSIONS

This section deals with the result analysis of proposed technique. The above algorithms have been successfully implemented and the results are shown in the table below.

8.1 Video data

1) Vipmen.avi

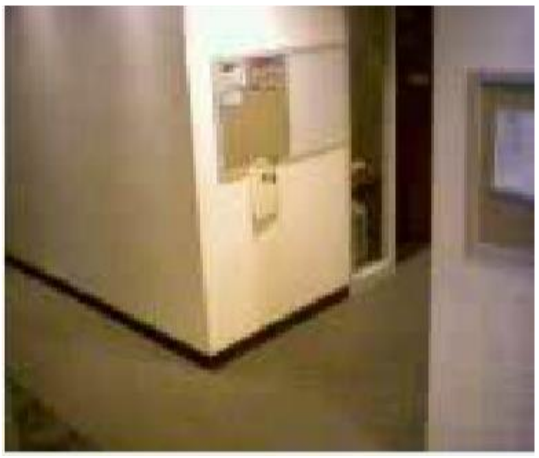


Fig. 6(a) Before Hiding

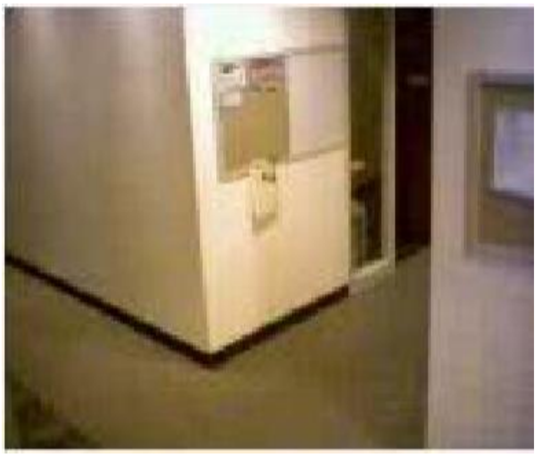


Fig. 6(b) After Hiding

2) Haky_car.avi

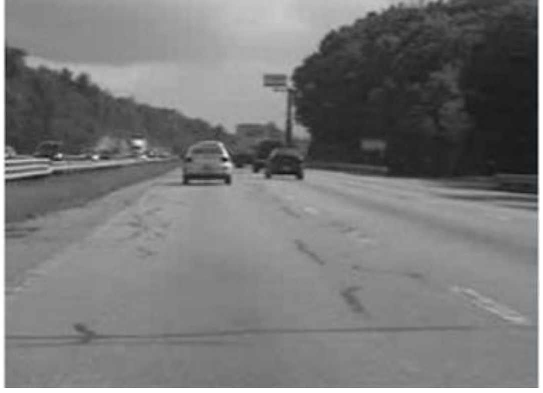


Fig. 7(a) Before Hiding

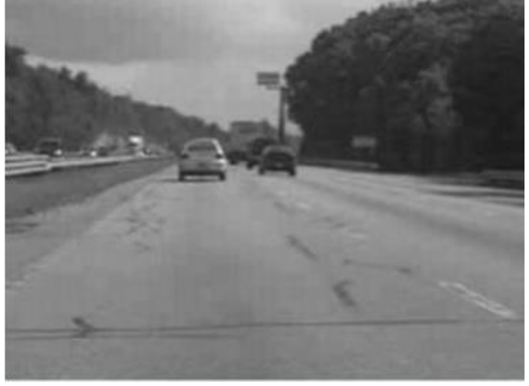


Fig. 7(b) After Hiding

3) Vipmosaicking.avi

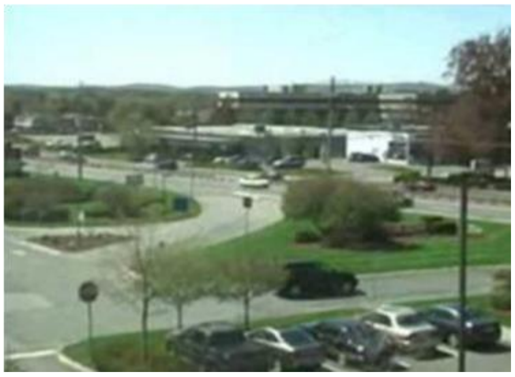


Fig. 8(a) Before Hiding

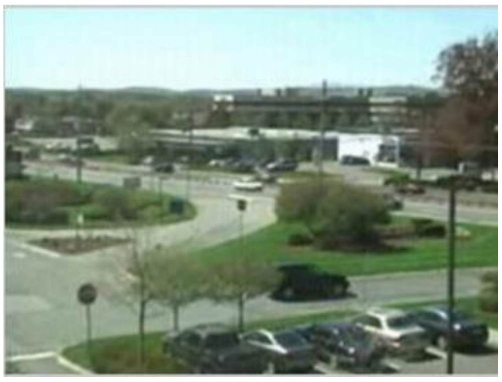


Fig. 8(b) After Hiding

4) Viptraffic.avi

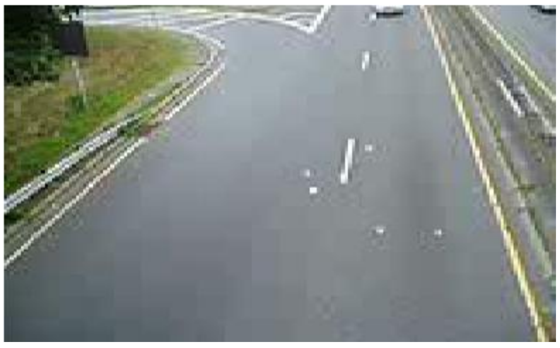


Fig. 9(a) Before Hiding

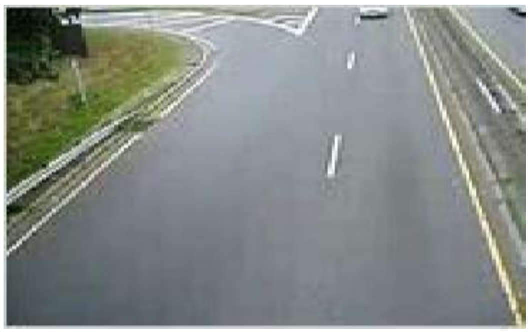


Fig. 9(b) After Hiding

8.2 Result Analysis

Table 3

Video	**MSE**	**RMSE**	**PSNR**
Vipmen.avi	0.7436	0.8623	49.4172
Haky_car.avi	0.7609	0.8723	49.3173
Vipomosaicking.avi	0.7601	0.8718	49.3221
Viptraffic.avi	0.7417	0.8612	49.4285

8.3 HISTOGRAM FOR FRAME

1 . Vipmen Video

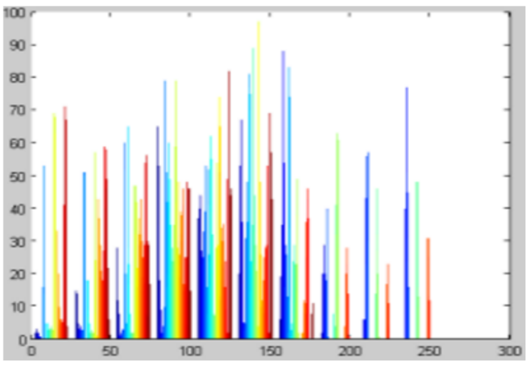


Fig.10(a) Before Hiding

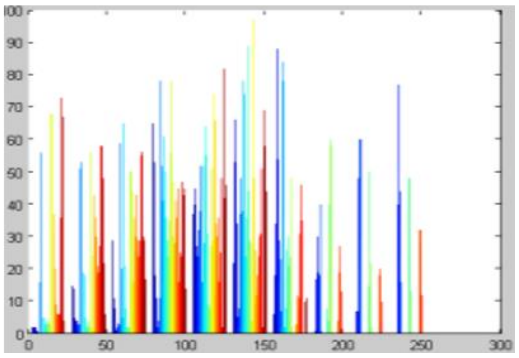


Fig.10(b) After Hiding

2. Haky_car Video

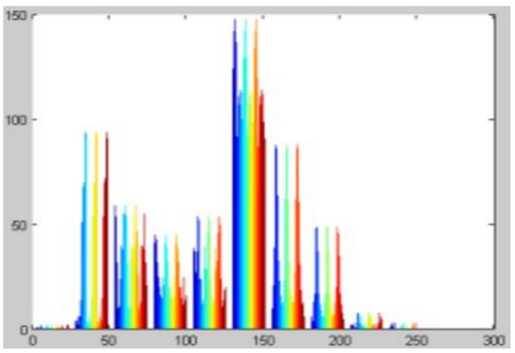


Fig. 11(a) Before Hiding

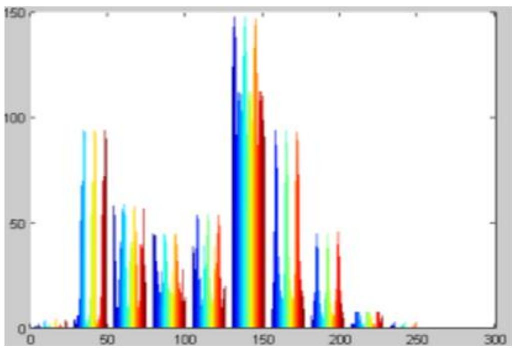
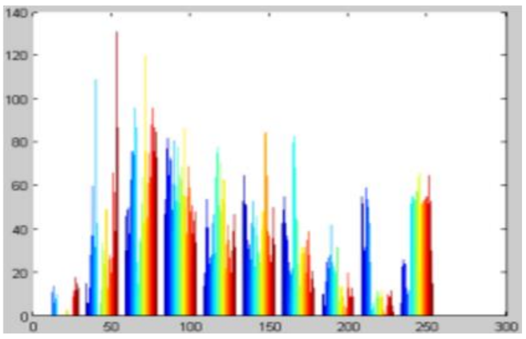


Fig.11 (b) After Hiding

3. Vipmosaicking Video



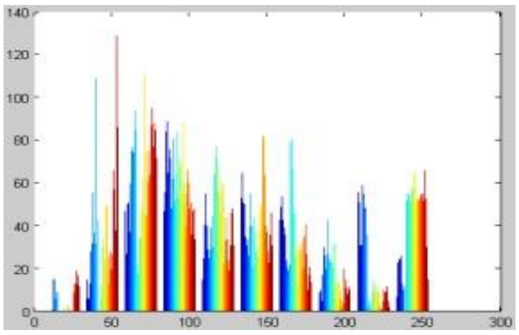
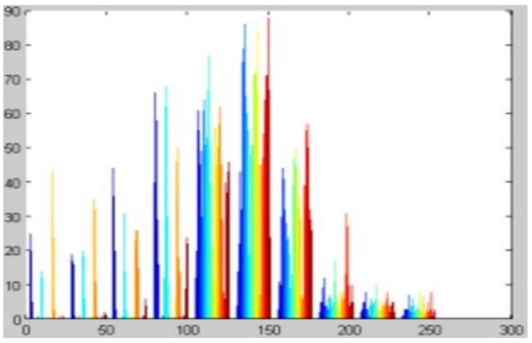


Fig. 12(a) Before Hiding

Fig. 12(b) After Hiding

4 . Viptraffic Video



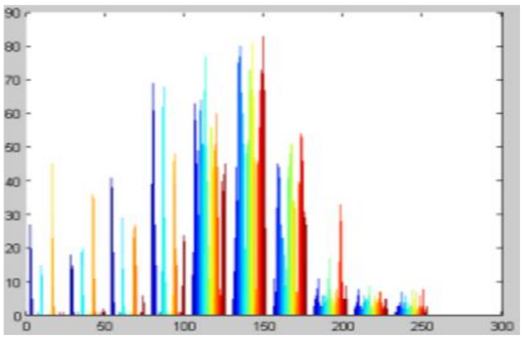


Fig. 13(a) Before Hiding

Fig. 13(b) After Hiding

CHAPTER 9: CONCLUSIONS and FUTURE SCOPE

Steganography is the art of hiding information in digital media in order to conceal the existence of the information. The paper provides a good method of steganography in video by using RSA algorithm, Lazy Wavelet Scheme and LSB technique. The data is hidden in video and the length is hidden in audio component using LSB technique, and the changes which are done in both the components is not recognizable. The proposed technique provides two layer securities by cryptography and Steganography. The technique provides a good capacity to store a high load message.

Experiments were conducted to demonstrate that video-Frame stenography provides a good trade-off between encryption, steganography robustness, flexibility, and real-time processing. The Lazy Wavelet Scheme is applied on each frame to derive sub bands. The proposed algorithm is robust since the payload is embedded into the transform cover image indirectly. The algorithm can be tested in future with some more transform techniques to improve the performance.

The proposed technique can be use in copyright control of materials, medical records, TV broadcasting, financial companies data safe circulation, smart Id cards and banking.

CHAPTER 10: APPENDIX

10.1 Coding

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

% % %Decryption
global select;
% taking input for text file to decypt the text file
if select==1
prompt = {'Enter the decyption key',' '};
dlg_title = 'Input';
num_lines = 1;
def = {'103', '143'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
% d = (answer{1});

d = str2num(answer{1})
Pk = str2num(answer{2})

% taking input from text file
[filename pathname]=uigetfile({'.'},'Browse the text file');
fulpathname=strcat(pathname,filename);
file=fulpathname;
 fileID = fopen(file);
  cipher = fscanf(fileID,'%c');
x=length(cipher);
for j= 1:x
   message(j)= crypt(double(cipher(j)),Pk,d);
end
% disp('Decrypted ASCII of Message:');
% disp(message);
% disp(['Decrypted Message is: ' message]);
% Saving the decypted message file
[file,path] = uiputfile('original_message.txt','Save decrypted file');
fulpathname=strcat(path,file);
file=fulpathname;
 fileID = fopen(file,'w');
 fwrite(fileID,[message]);
   msgbox('Decryption Completed,Original message is retrieved, File is successfully saved
');
end
% taking input for image file as to decrypt the image
if select ==2
    prompt = {'Enter the decryption key'};
dlg_title = 'Input';
num_lines = 1;
def = {'ekansh'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
d = (answer{1});

% d = str2num(answer{1})
% Pk = str2num(answer{2})

% to select the image to decrypt
[filename pathname]=uigetfile({'.'},'Browse the image to decrypt');
fulpathname=strcat(pathname,filename);
file=fulpathname;
% global M;
M = imread(file);
% outputFullFileName='C:\Users\Ekansh\Desktop\gray.png';
% imwrite(M, outputFullFileName, 'png');
msgbox('Decryption process has been started, it would take some time');
% M = input('\nEnter the message: ','s');
```

```
M1=M(:,:,1);
M2=M(:,:,2);
M3=M(:,:,3);
cipher_text='';
[m n]=size(M1);
key=d;
l=length(key);
l1=1;
% k=1;
for i=1:m
for j= 1:n
%   for j1=1:n1

%    cipher(k)= crypt(double(M1(i,j)),Pk,e);

M1(i,j)= bitxor(double(key(l1)),M1(i,j));
l1=l1+1;
if(l1>l)
    l1=1;
end
%   k=k+1;
% end
end
end



% cipher_text=[cipher_text,cipher];
[m n]=size(M2);
l1=1;

for i=1:m
for j= 1:n
```

```
%   for j1=1:n1
%    cipher(k)= crypt(double(M1(i,j)),Pk,e);
M2(i,j)= bitxor(double(key(l1)),M2(i,j));
l1=l1+1;
if(l1>l)
    l1=1;
end
%   k=k+1;
% end
end
end
% k=k-1;
% cipher_text=[cipher_text,cipher];
[m n]=size(M3);
l1=1;
for i=1:m
for j= 1:n
%   for j1=1:n1

%    cipher(k)= crypt(double(M1(i,j)),Pk,e);
%   k=k-1;
M3(i,j)= bitxor(double(key(l1)),M3(i,j));
l1=l1+1;
if(l1>l)
    l1=1;
end
%   k=k+1;
% end
end
end
M=cat(3,M1,M2,M3);
 [file,path] = uiputfile('original_image.png','Save original_image');
```

```
outputFullFileName=strcat(path,file);

% outputFullFileName='C:\Users\Ekansh\Desktop\original_image.png';
imwrite(M, outputFullFileName, 'png');
end
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% clc;
% disp('Implementation of RSA Algorithm');
clear all;
% close all;

prompt = {'Enter 1st prime number:','Enter 2nd prime number:'};
dlg_title = 'Input';
num_lines = 1;
def = {'11','13'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
p = str2num(answer{1});
q = str2num(answer{2});
[Pk,Phi,d,e] = intialize(p,q);


% taking input from text file
[filename pathname]=uigetfile({'.txt'},'Browse the text file');
fulpathname=strcat(pathname,filename);
file=fulpathname;
 fileID = fopen(file);
  M = fscanf(fileID,'%c');

msgbox('Encrption process has been started, it would take some time');
% M = input('\nEnter the message: ','s');
x=length(M);
```

```
c=0;
for j= 1:x
    for i=0:122
        if strcmp(M(j),char(i))
            c(j)=i;
        end
    end
end

% disp('ASCII Code of the entered Message:');
% disp(c);



% % %Encryption
for j= 1:x
   cipher(j)= crypt(c(j),Pk,e);
end
% disp('Cipher Text of the entered Message:');
% disp(cipher);

% Saving the encypted file
[file,path] = uiputfile('secret_message.txt','Save encrypted file');
fulpathname=strcat(path,file);

 file=fulpathname;
 fileID = fopen(file,'w');
 fwrite(fileID,[cipher]);

% saving the private key

[file,path] = uiputfile('private_key.txt','Save Decryption key');
fulpathname=strcat(path,file);
```

```
 file=fulpathname;
 fileID = fopen(file,'w');
 p1=num2str(d);
 p2=num2str(Pk);
  secretkey=' ';
  secretkey=[secretkey,p1];
  secretkey=[secretkey,' '];
  secretkey=[secretkey,p2];

 fwrite(fileID,secretkey);

 msgbox('Encryption Completed, File is saved successfully');
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



function [Pk,Phi,d,e] = intialize(p,q)
clc;
disp('Intaializing:');
Pk=p*q;
Phi=(p-1)*(q-1);
%Calculate the value of e
x=2;e=1;
while x > 1
    e=e+1;
    x=gcd(Phi,e);
end
%Calculate the value of d
i=1;
r=1;
```

```
while r > 0
    k=(Phi*i)+1;
    r=rem(k,e);
    i=i+1;
end
d=k/e;
clc;
% disp(['The value of (N) is: ' num2str(Pk)]);
% disp(['The public key (e) is: ' num2str(e)]);
% disp(['The value of (Phi) is: ' num2str(Phi)]);
% disp(['The private key (d)is: ' num2str(d)]);


function a = dec2bin(d)
i=1;
a=zeros(1,65535);
while d >= 2
    r=rem(d,2);
    if r==1
        a(i)=1;
    else
        a(i)=0;
    end
    i=i+1;
    d=floor(d/2);
end
if d == 2
    a(i) = 0;
else
    a(i) = 1;
end
x=[a(16) a(15) a(14) a(13) a(12) a(11) a(10) a(9) a(8) a(7) a(6) a(5) a(4) a(3) a(2) a(1)];
```

```
function mc = crypt(M,N,e)
e=dec2bin(e);
k = 65535;
c  = M;
cf = 1;
cf=mod(c*cf,N);
for i=k-1:-1:1
    c = mod(c*c,N);
    j=k-i+1;
     if e(j)==1
         cf=mod(c*cf,N);
     end
end
mc=cf;
% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)

% Output folder
global outputFolder;
msgbox('Hiding procedure has been started');
outputFolder = fullfile(cd, 'frames');
if ~exist(outputFolder, 'dir')
mkdir(outputFolder);
end

global mov;

%getting no of frames
 % to access global variable
numberOfFrames = mov.NumberOfFrames;
numberOfFramesWritten = 0;
global usedFrame;
```

```
usedFrame=0;   %to count the no of frames use to hide

for frame = 1 :numberOfFrames

thisFrame = read(mov, frame);
outputBaseFileName = sprintf('%d.png', frame);
outputFullFileName = fullfile(outputFolder, outputBaseFileName);
imwrite(thisFrame, outputFullFileName, 'png');
progressIndication = sprintf('Wrote frame %4d of %d.', frame,numberOfFrames);
disp(progressIndication);
numberOfFramesWritten = numberOfFramesWritten + 1;
end
progressIndication = sprintf('Wrote %d frames to folder "%s"',numberOfFramesWritten,
outputFolder);
disp(progressIndication);
 numberOfFramesWritten=0;
% output folder is same as input folder

% Output folder for stego video
outputFolder = fullfile(cd, 'stego frames');
if ~exist(outputFolder, 'dir')
mkdir(outputFolder);
end



msg1='';
global select
if select==1   % to hide text in video
% to read data from  text file
display('text')
% taking input from text file
[filename pathname]=uigetfile({'.'},'Browse the text file to hide');
```

```
fulpathname=strcat(pathname,filename);
file=fulpathname;
 fileID = fopen(file);
msg1 = fscanf(fileID,'%c');

end
if select ==2 % to hide the image in video
display('image');
    [filename pathname]=uigetfile({'.'},'Browse the image to hide');
fulpathname=strcat(pathname,filename);
file=fulpathname;
M = imread(file);

M1=M(:,:,1);
M2=M(:,:,2);
M3=M(:,:,3);

cipher_text='';
[m n]=size(M1);


 for i=1:m
for j= 1:n
msg1=[msg1,double(M1(i,j))];
end
end
for i=1:m
for j= 1:n
msg1=[msg1,double(M2(i,j))];
end
end
for i=1:m
for j= 1:n
msg1=[msg1,double(M3(i,j))];
end
end
end

% msg1=str2num(msg1);
% msg1=double(msg1);
%  msg=dec2base(msg1, 2);

 msg=dec2base(msg1, 2,14);

%  msg=dec2bin(msg1) % decimal is converted into 7 bits binary
[m n]=size(msg)% to calculate the size of matrix
a=1;
 b=7; % as msg data is represent in 16 bits and 1st 8 bits are of no use(as the are 0)

% start hiding data
add2=0;  % initializing frame url part
for frame=1:numberOfFrames

    if(a<m && b<n)
        display('Data is hiding in new frame');
 add1='';
 outputFolder = fullfile(cd, 'frames');
 add1=[add1,outputFolder];
   add1=[add1,'\'];
%         add1='C:\Users\Ekansh\Desktop\ekansh\project\frames\';
    add2=add2+1;% to increment the frame no so that new frame can be access each time
   add3=num2str(add2);
    add4='.png';
    add=[add1,add3,add4]
```

```
imdata = imread(add);
 liftscheme = liftwave('lazy','int2int');
 [cA cH cV cD]=lwt2(imdata,liftscheme);

 global pixelused;
pixelused=0; % to count the no of pixels use to hide in last frame

% hiding data in cA
R = cA(:,:,1); % Get the RED matrix
G = cA(:,:,2); % Get the GREEN matrix
B = cA(:,:,3); % Get the BLUE matrix

% hiding data in R of cA
 [m1 n1]=size(R);
 R1=R;
  display('hiding data in R of cA ')
 for i=1:m1
 for j=1:n1
     if(a<=m)
         if(b<n)
            c=msg(a,b);
            c = str2num(c);
            d=R(i,j);
            R(i,j)= bitset(d, 2,c ) ;
            b=b+1;
            c=msg(a,b);
            c = str2num(c);
            d=R(i,j);
            R(i,j)= bitset(d, 1,c);
            b=b+1;
       pixelused=pixelused+1;
```

```
         else
             if(a~=m)
             a=a+1;
             b=7;
              c=msg(a,b);
              c = str2num(c);
              d=R(i,j);
            R(i,j)= bitset(d, 2,c ) ;
            b=b+1;
            c=msg(a,b);
            c = str2num(c);
            d=R(i,j);
            R(i,j)= bitset(d, 1,c);
            b=b+1;
             pixelused=pixelused+1;
             end
         end
       end
 end
 end
% to check whether data is left for hiding
% hiding data in G of cA
if(a<=m|| b<=n)
 display('hiding data in G of cA ')
G = cA(:,:,2); % Get the GREEN matrix

 [m1 n1]=size(G);
 G1=G;

 for i=1:m1
 for j=1:n1
     if(a<=m)
```

```
          %display('hhhhhhhhhhhhhhhhhhhhhhhh')
         if(b<n)
            c=msg(a,b);
            c = str2num(c);
            d=G(i,j);
            G(i,j)= bitset(d, 2,c ) ;
            b=b+1;
            c=msg(a,b);
            c = str2num(c);
            d=G(i,j);
            G(i,j)= bitset(d, 1,c);
            b=b+1;
             pixelused=pixelused+1;
         else  if(a~=m)
             a=a+1;
             b=7;
              c=msg(a,b);
              c = str2num(c);
              d=G(i,j);
            G(i,j)= bitset(d, 2,c ) ;
            b=b+1;
            c=msg(a,b);
            c = str2num(c);
            d=G(i,j);
            G(i,j)= bitset(d, 1,c);
            b=b+1;
             pixelused=pixelused+1;
             end
         end
      end
  end
 end
```

```
end

% to check whether data is left for hiding
% hiding data in B of cA
if(a<=m|| b<=n)
 display('hiding data in B of cA ')
 B = cA(:,:,3); % Get the BLUE matrix
 [m1 n1]=size(B);
 B1=B;
 for i=1:m1
 for j=1:n1
     if(a<=m)
         %display('hhhhhhhhhhhhhhhhhhhhhhhh')
         if(b<n)
            c=msg(a,b);
            c = str2num(c);
            d=B(i,j);
            B(i,j)= bitset(d, 2,c ) ;
            b=b+1;
            c=msg(a,b);
            c = str2num(c);
            d=B(i,j);
            B(i,j)= bitset(d, 1,c);
            b=b+1;
             pixelused=pixelused+1;
         else if(a~=m)
             a=a+1;
             b=7;
              c=msg(a,b);
              c = str2num(c);
              d=B(i,j);
            B(i,j)= bitset(d, 2,c ) ;
            b=b+1;
            c=msg(a,b);
            c = str2num(c);
            d=B(i,j);
            B(i,j)= bitset(d, 1,c);
            b=b+1;
             pixelused=pixelused+1;
             end
         end
        end
 end
 end
end

% to check whether data is left for hiding
% hiding data in B of cH
if(a<=m|| b<=n)
display('hiding data in B of cH ')
 B = cH(:,:,3); % Get the BLUE matrix
 [m1 n1]=size(B);
 B1=B;
 for i=1:m1
 for j=1:n1
     if(a<=m)
         if(b<n)
            c=msg(a,b);
            c = str2num(c);
            d=B(i,j);
            B(i,j)= bitset(d, 2,c ) ;
            b=b+1;
            c=msg(a,b);
            c = str2num(c);
            d=B(i,j);
            B(i,j)= bitset(d, 1,c);
            b=b+1;
             pixelused=pixelused+1;
         else if(a~=m)
             a=a+1;
             b=7;
              c=msg(a,b);
              c = str2num(c);
              d=B(i,j);
            B(i,j)= bitset(d, 2,c ) ;
            b=b+1;
            c=msg(a,b);
            c = str2num(c);
            d=B(i,j);
            B(i,j)= bitset(d, 1,c);
            b=b+1;
             pixelused=pixelused+1;
             end
         end
          end
 end
 end
end

% to recover CD band from RGB
CD = cat(3, R, G, B);
%x=ilwt2(imdata,liftscheme);
X=ilwt2(CA,CH,CV,CD,liftscheme);
 X=uint8(X);
 % imshow(X);
%imwrite(X,'modified_image.png')
```

```
 %thisFrame = read(mov, stegoframe);
outputBaseFileName = sprintf('%d.png', frame);
outputFullFileName = fullfile(outputFolder, outputBaseFileName);
imwrite(X, outputFullFileName, 'png');
progressIndication = sprintf('Wrote frame %4d of %d.', frame,numberOfFrames);
disp(progressIndication);
numberOfFramesWritten = numberOfFramesWritten + 1; usedFrame=usedFrame+1
     end
    if b==15
        b=7;
    end
end
pixelused
% workingDir = tempname;
 outputFolder = fullfile(cd); %to change the directory i.e. back to    one folder
workingDir=outputFolder;
% workingDir='C:\Users\Ekansh\Desktop\ekansh\project\final ppt';
% Read and Sort the Image Sequence
imageNames = dir(fullfile(workingDir,'frames','*.png'));
imageNames = {imageNames.name}';

%  First, match any file names that contain a sequence of numeric digits. Convert the strings
to doubles.
imageStrings = regexp([imageNames{:}],'(\d*)','match');
imageNumbers = str2double(imageStrings);

% Sort the frame numbers from lowest to highest. The sort function returns an index matrix
that indicates how to order the associated files.

[~,sortedIndices] = sort(imageNumbers);
sortedImageNames = imageNames(sortedIndices);
```

```
%  Construct a VideoWriter object, which creates a Motion-JPEG AVI file by default.
global fulpathname1;
inputVideo = VideoReader(fulpathname1);
outputVideo = VideoWriter(fullfile(workingDir,'eku.avi'));
outputVideo.FrameRate = inputVideo.FrameRate;
open(outputVideo);

% Loop through the image sequence, load each image, and then write it to the video.
for ii = 1:numberOfFrames     %no of frames
   img = imread(fullfile(workingDir,'frames',sortedImageNames{ii}));

   writeVideo(outputVideo,img);
end
video.MultimediaFileWriter
% Finalize the video file.
close(outputVideo);
% extracting message from R component of cA1

 R2 = cA1(:,:,1); % Get the RED matrix

 [m1 n1]=size(R2);
 a=1;
 b=1;
 k=1;
% c(m1,n1);

 for i=1:m1
 for j=1:n1


           pixelval=R2(i,j);
          % pixelbin=dec2bin(pixelval,8);
         c(k,b)=bitget(pixelval,2);
```

```
          b=b+1;
          c(k,b)=bitget(pixelval,1);
          b=b+1;
          if(b==9)
             b=1;
             k=k+1;
          end
end
 end
str = num2str(c);
% str=cellstr(str);
 str=char(bin2dec(str));
 s=size(str);
for i=1:s
    message=[message,str(i)];
    % message=strcat(message,str(i))
 End
```

10.2 Screenshot

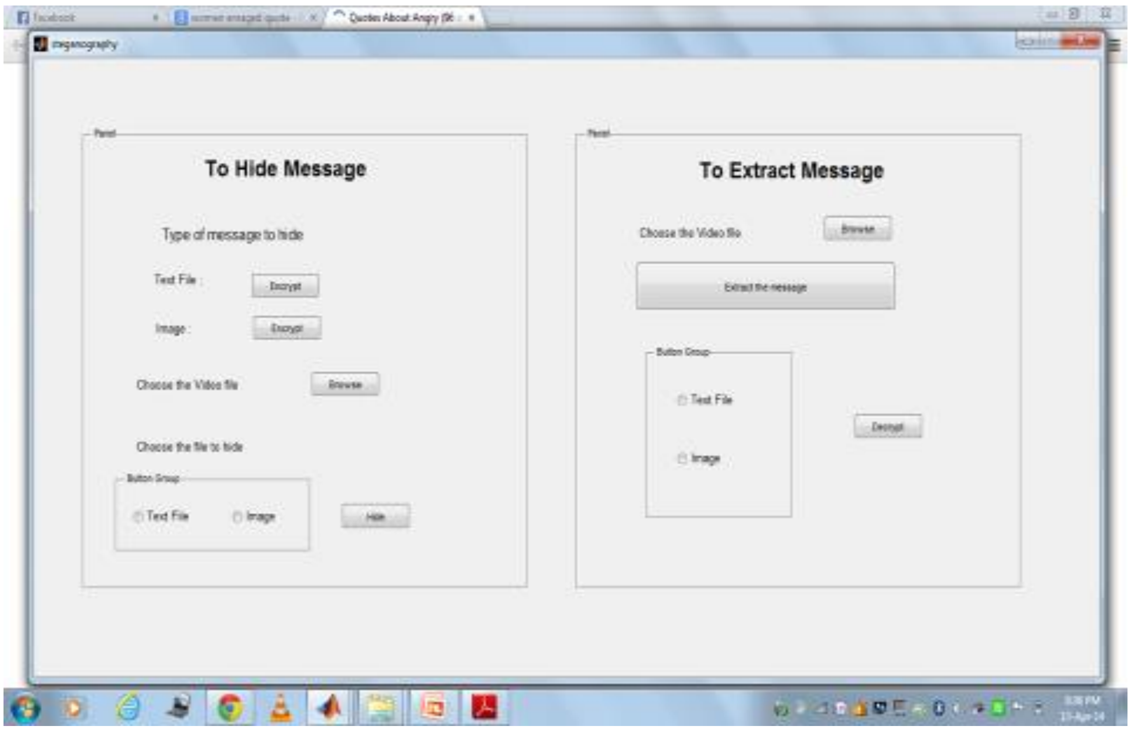
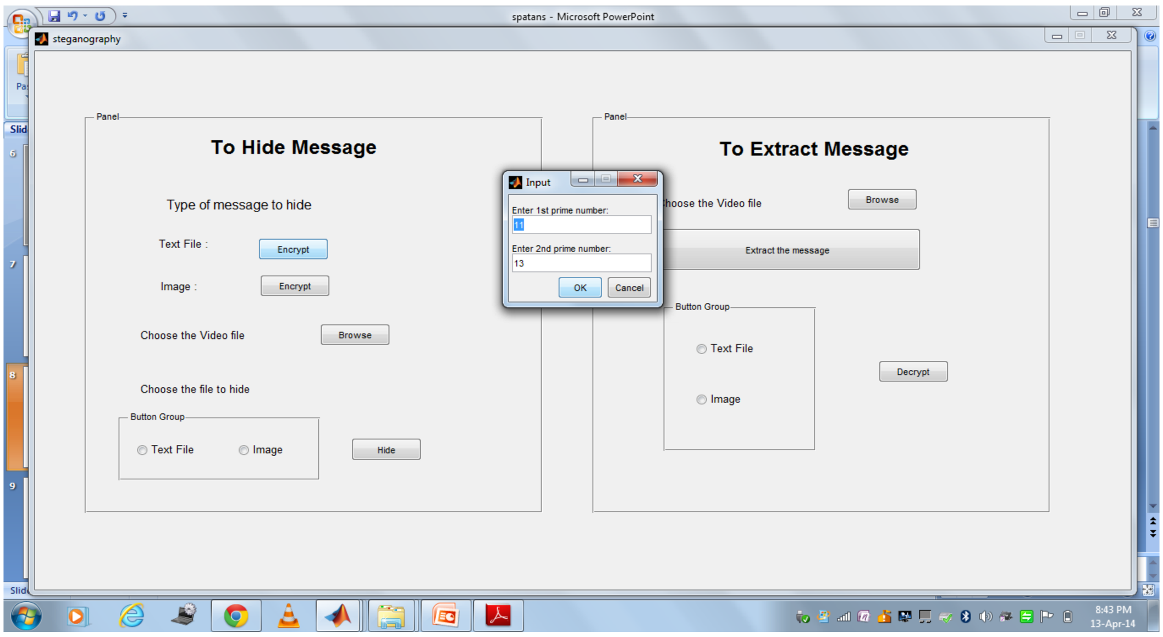


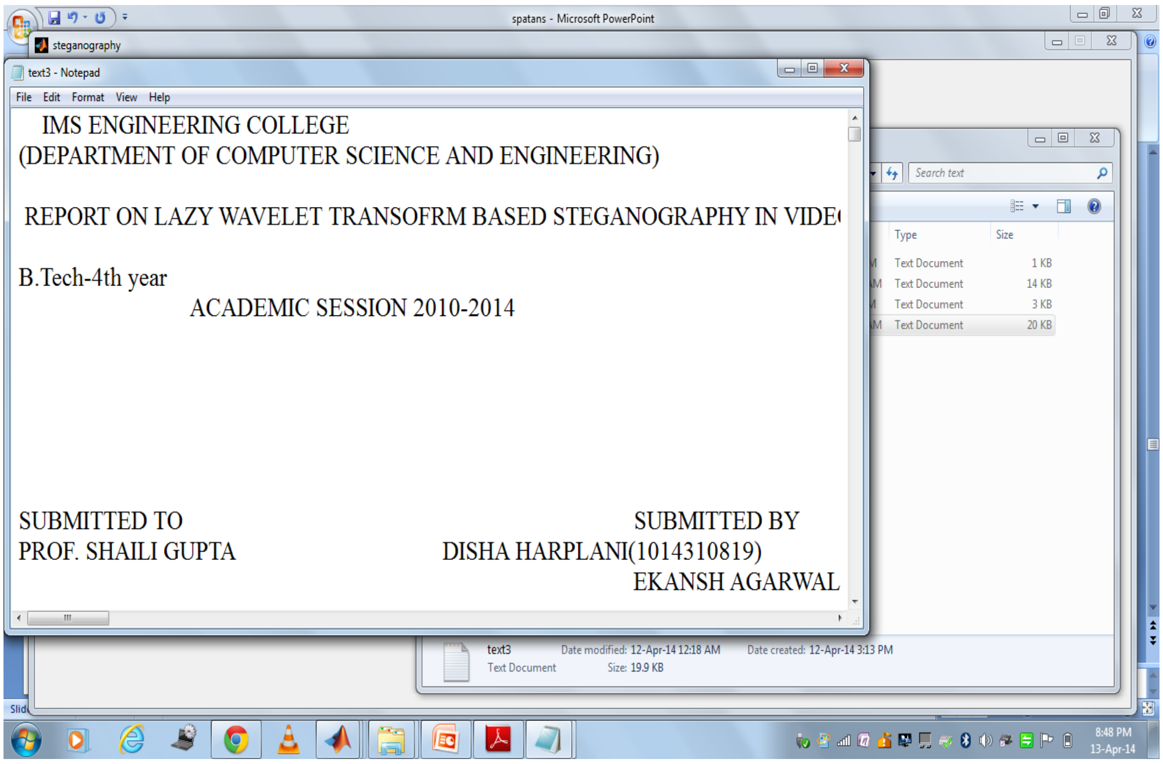
Fig 14: Home page



Enter 1st prime number: 11

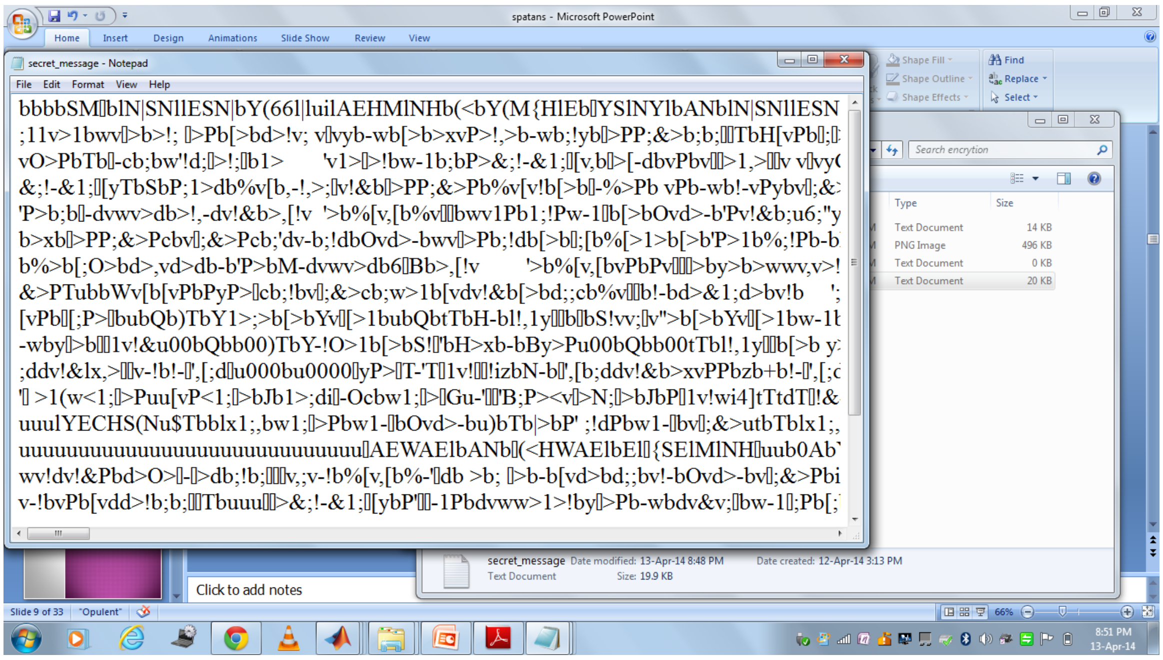
Enter 2nd prime number: 13

Fig 15: Encryption



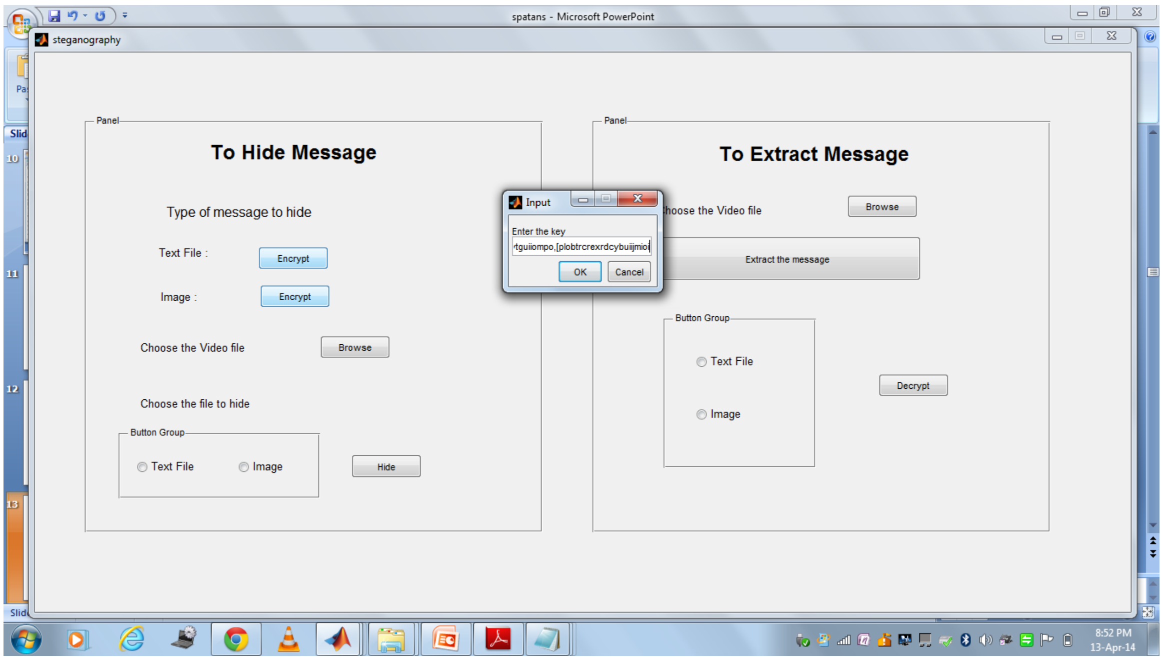
```
   IMS ENGINEERING COLLEGE
(DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING)

 REPORT ON LAZY WAVELET TRANSOFRM BASED STEGANOGRAPHY IN VIDE

B.Tech-4th year
                    ACADEMIC SESSION 2010-2014





SUBMITTED TO                                                    SUBMITTED BY
PROF. SHAILI GUPTA                                   DISHA HARPLANI(1014310819)
                                                                EKANSH AGARWAL
```

Fig 16: Input Text



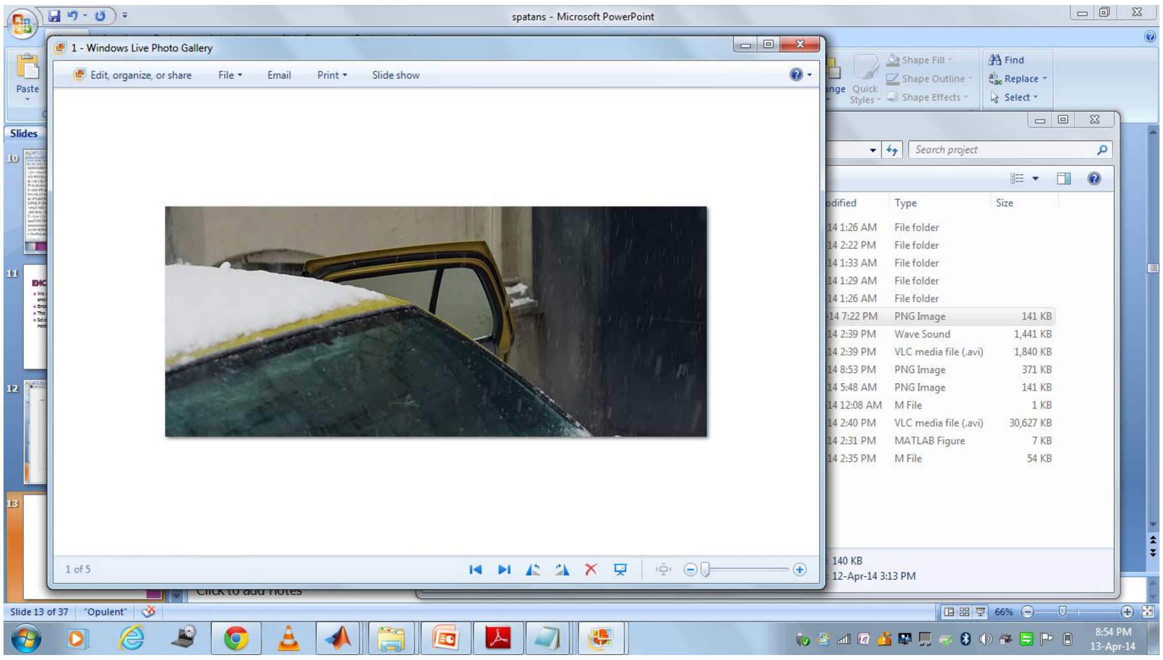
```
bbbbSM▯bIN|SNIlESN|bY(66l|luilAEHMlNHb(<bY(M{HlEb▯YSlNYlbANbIN|SNIlESN
;11v>1bwv▯>b>!; ▯>Pb[>bd>!v; v▯vyb-wb[>b>xvP>!,>b-wb;!yb▯>PP;&>b;b;▯▯TbH[vPb▯;▯
vO>PbTb▯-cb;bw'!d;▯>!;▯b1>      'v1>▯>!bw-1b;bP>&;!-&1;▯[v,b▯>[-dbvPbv▯▯>1,>▯▯v v▯vy(
&;!-&1;▯[yTbSbP;1>db%v[b,-!,>;▯v!&b▯>PP;&>Pb%v[v!b[>b▯-%>Pb vPb-wb!-vPybv▯;&
'P>b;b▯-dvwv>db>!,-dv!&b>,[!v  '>b%[v,[b%v▯▯bwv1Pb1;!Pw-1▯b[>bOvd>-b'Pv!&b;u6;"y
b>xb▯>PP;&>Pcbv▯;&>Pcb;'dv-b;!dbOvd>-bwv▯>Pb;!db[>b▯;[b%[>1>b[>b'P>1b%;!Pb-b
b%>b[;O>bd>,vd>db-b'P>bM-dvwv>db6▯Bb>,[!v       '>b%[v,[bvPbPv▯▯▯>by>b>wwv,v>!
&>PTubbWv[b[vPbPyP>▯cb;!bv▯;&>cb;w>1b[vdv!&b[>bd;;cb%v▯▯b!-bd>&1;d>bv!b    ';
[vPb▯[;P>▯bubQb)TbY1>;>b[>bYv▯[>1bubQbtTbH-bl!,1y▯▯b▯bS!vv;▯v">b[>bYv▯[>1bw-1t
-wby▯>b▯▯1v!&u00bQbb00)TbY-!O>1b[>bS!▯'bH>xb-bBy>Pu00bQbb00tTbl!,1y▯▯b[>b y
;ddv!&lx,>▯▯v-!b!-▯',[;d▯u000bu0000▯yP>▯T-'T▯1v!▯▯!izbN-b▯',[b;ddv!&b>xvPPbzb+b!-▯',[;c
'▯ >1(w<1;▯>Puu[vP<1;▯>bJb1>;di▯-Ocbw1;▯>▯Gu-'▯▯'B;P><v▯>N;▯>bJbP▯1v!wi4]tTtdT▯!&
uuulYECHS(Nu$Tbblx1;,bw1;▯>Pbw1-▯bOvd>-bu)bTb|>bP' ;!dPbw1-▯bv▯;&>utbTblx1;,
uuuuuuuuuuuuuuuuuuuuuuuuuuuuuu▯AEWAElbANb▯(<HWAElbEl▯{SElMlNH▯uub0Ab'
wv!dv!&Pbd>O>▯-▯>db;!b;▯▯▯v,;v-!b%[v,[b%-'▯db >b; ▯>b-b[vd>bd;;bv!-bOvd>-bv▯;&>Pbi
v-!bvPb[vdd>!b;b;▯▯Tbuuu▯▯>&;!-&1;▯[ybP'▯▯-1Pbdvww>1>!by▯>Pb-wbdv&v;▯bw-1▯;Pb[;
```

Fig 17: Encrypted Text



Enter the key: tguiiompo,[plobtrcrexrdcybuiijmioi

Fig 18: Image Encyption



odified	Type	Size
14 1:26 AM	File folder	
14 2:22 PM	File folder	
14 1:33 AM	File folder	
14 1:29 AM	File folder	
14 1:26 AM	File folder	
14 7:22 PM	PNG Image	141 KB
14 2:39 PM	Wave Sound	1,441 KB
14 2:39 PM	VLC media file (.avi)	1,840 KB
14 8:53 PM	PNG Image	371 KB
14 5:48 AM	PNG Image	141 KB
14 12:08 AM	M File	1 KB
14 2:40 PM	VLC media file (.avi)	30,627 KB
14 2:31 PM	MATLAB Figure	7 KB
14 2:35 PM	M File	54 KB

140 KB
12-Apr-14 3:13 PM

Fig 19: Input Image

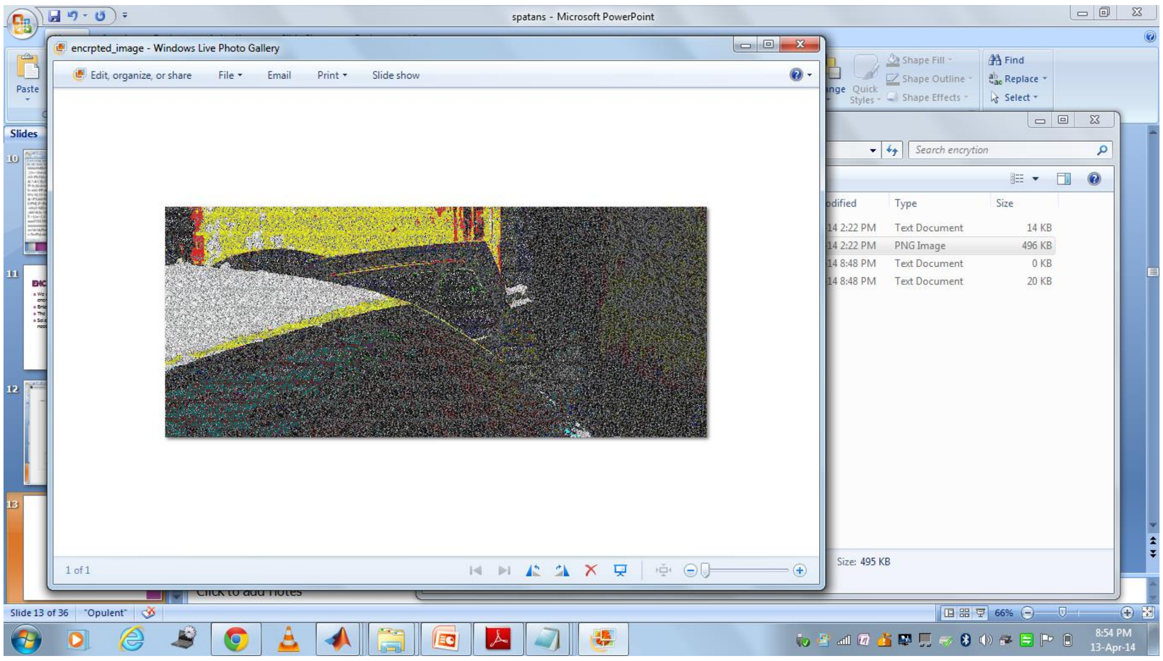


Fig 20: Encrypted Image

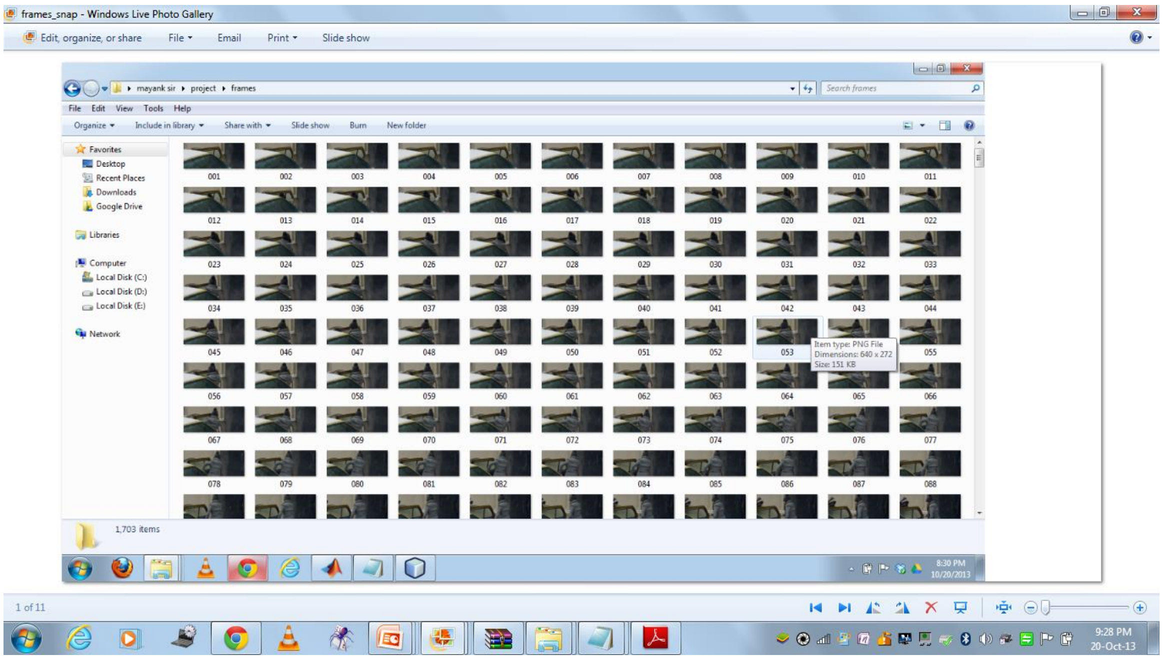
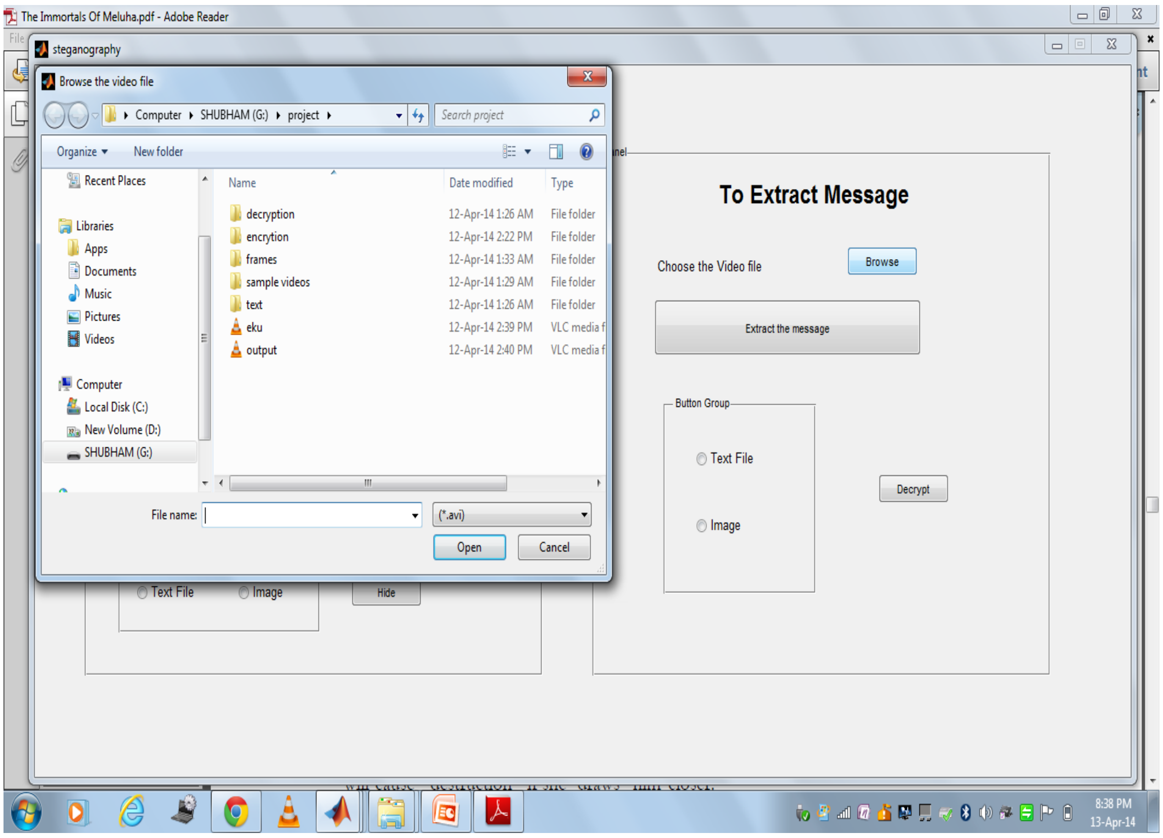


Fig 21: Video Frames



Name	Date modified	Type
decryption	12-Apr-14 1:26 AM	File folder
encrytion	12-Apr-14 2:22 PM	File folder
frames	12-Apr-14 1:33 AM	File folder
sample videos	12-Apr-14 1:29 AM	File folder
text	12-Apr-14 1:26 AM	File folder
eku	12-Apr-14 2:39 PM	VLC media f
output	12-Apr-14 2:40 PM	VLC media f

Fig 22: Browse video file

CHAPTER 11: REFERENCES

[1] H S Manjunatha Reddy. Wavelet based Non LSB Steganography. Department of ECE, Global Academy of Technology, Bangalore-98, India

[2] Khushman Patel, Kul Kauwid Rora, Kamini Singh, Shekhar Verma. Lazy Wavelet Transform Based Steganography in Video. Dept of CSE, IIIT Allahabad, Allahabad (U.P.)

[3] Cryptography. From Wikipedia http://en.wikipedia.org/ wiki / Cryptography

[4] G. Doerr and J.L. Dugelay, "Security pitfalls of framebyframe approaches to video," IEEE Trans. Sig. Proc., vol. Supplement on Secure Media, no. 52, pp. 29552964, 2004.

[5] Symmetric key cryptography. From Wikipedia http://en.wikipedia.org/wiki/Symmetric_key_cryptograp hy

[6] Public key cryptography. From Wikipedia http://en.wikipedia.org /wiki/Public-key_cryptography

[7] RSA from http://en.wikipedia.org/wiki/RSA

[8] Ashish T. Bhole, Rachna Patel. Steganography over Video File using Random Byte Hiding and LSB Technique Department of Computer Engineering, SSBT's COE & T, Bambhori, Jalgaon, India

[9] T. Natramizhnangai and R. SenthilRajan A Novel Steganography Method based on Integer Wavelet Transform and Genetic Algorithm.

[10] Ming Chen, "Analysis of Current Steganography Tools: Classifications & Features," in International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2006, pp. 384 – 387, 2006

[11] D. Artz, "Digital Steganography: Hiding Data within Data," IEEE Internet Computing Journal, June 2001.

[12] Mayank Arya Chandra Ravindra Purwar, Navin Rajpal
"A Novel Approach of Digital Video Encryption" International Journal of Computer Applications (0975 – 8887) Volume 49– No.4, July 2012

[13] Alligood, K. T., Sauer, T., Yorke, J. A. : Chaos: an introduction to dynamical systems. Springer, Heidelberg (1997).

[14] Devaney, R. L. :An introduction to chaotic dynamical systems, 2nd edn.West- view Press, San Francisco
(2003).

[15] Yang, T., Wu, C. W., Chua, L.O. : Cryptography based on chaotic systems. IEEE Transactionson Circuits and Systems-I : Fundamental Theory and Applications 44, 469…472 (1997).

[16] Solak, E. : Cryptanalysis of observer based discrete-time chaotic encryption schemes. International Journal of Bifurcation and Chaos 15 (2), 653…658 (2005).

[17] He, J., Qian, H., Zhou, Y., Li, Z.: Cryptanalysis and improvement of a block cipher based on multiple chaotic systems. Mathematical Problems in Engineering 2010,1…14 (2010).

[18] Mohd. Arif Siddique, Mayank Arya Chandra, Kunwar Babar Ali. "Improved Graphical Password Authentication using Dynamic Grid and Image Zoom" IJAER Volume 6, Number 18 (2011)

[19] Zeghid, M., Machhout, M., Khriji, L., Baganne, A., et al.: A modified AES based algorithm for image encryption. International Journal of Computer Science and Engineering 1 (1), 70…75(2007).

[20] Hongmei Tang, Gaochan Jin, Cuixia Wu and Peijiao Song, A New Image Encryption and Steganography Scheme, IEEE International Conference on Computer and Communications Security,2009, 60-63.

[21] R O El Safy, H H Zayed and A El Dessouki, An Adaptive Steganographic Technique Based on Integer Wavelet Transform, IEEE International Conference on Networking and Media Convergence, 2009, 111-117.

[22] Wang Z, Chang C, Lin C, Li M (2009) A reversible information hiding scheme using left-right and updown Chinese character representation. J Syst Softw 82:1362 –1369