1 INTRODUCTION

1.1 Network Security:

Network Security consists of the provisions and policies adopted by a network administrator to prevent and monitor unauthorized access, misuse, modification, or denial of a computer network and network-accessible resources. Network security involves the authorization of access to data in a network, which is controlled by the network administrator. Users choose or are assigned an ID and password or other authenticating information that allows them access to information and programs within their authority.

Network security covers a variety of computer networks, both public and private, that are used in everyday jobs conducting transactions and communications among businesses, government agencies and individuals. Networks can be private, such as within a company, and others which might be open to public access. Network security is involved in organizations, enterprises, and other types of institutions. It does as its title explains: It secures the network, as well as protecting and overseeing operations being done. The most common and simple way of protecting a network resource is by assigning it a unique name and a corresponding password.

The networks are computer networks, both public and private, that are used every day to conduct transactions and communications among businesses, government agencies and individuals. The networks are comprised of "nodes", which are "client" terminals (individual user PCs), and one or more "servers" and/or "host" computers. They are linked by communication systems, some of which might be private, such as within a company and others which might be open to public access. The obvious example of a network system that is open to public access is the Internet, but many private networks also utilize publicly-accessible communications.

Today, most companies' host computers can be accessed by their employees whether in their offices over a private communications network, or from their homes or hotel rooms while on the road through normal telephone line.

Network security involves all activities that organizations, enterprises, and institutions undertake to protect the value and ongoing usability of assets and the integrity and continuity of operations. An effective network security strategy requires identifying threats and then choosing the most effective set of tools to combat them.

1.1.1 Basic Security Assumptions

Several new assumptions have to be made about computer networks because of their evolution over the years:

- Modern networks are very large, very interconnected, and run both ubiquitous protocols (such as IP) and proprietary protocols. Therefore, they are often open to access, and a potential attacker can with relative ease attach to, or remotely access, such networks. Widespread IP internetworking increases the probability that more attacks will be carried out over large, heavily interconnected networks, such as the Internet.
- Computer systems and applications that are attached to these networks are becoming increasingly complex. In terms of security, it becomes more difficult to analyze, secure, and properly test the security of the computer systems and applications; it is even more so when virtualization is involved. When these systems and their applications are attached to large networks, the risk to computing dramatically increases.

1.1.2 Basic Security Requirements

To provide adequate protection of network resources, the procedures and technologies that you deploy need to guarantee three things, sometimes referred to as the CIA triad:

Network Security concerns itself with the following four objectives:

Confidentiality: Providing confidentiality of data guarantees that only authorized users can view sensitive information. (The information cannot be understood by anyone for whom it was unintended)

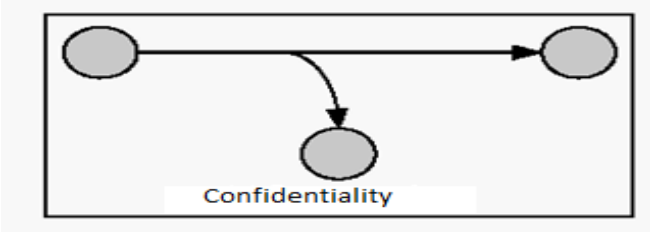

Figure1.1.2.1 confidentiality

Integrity: Providing integrity of data guarantees that only authorized users can change sensitive information and provides a way to detect whether data has been tampered with during transmission; this might also guarantee the authenticity of data. (The information cannot be altered in storage or transit between sender and intended receiver without the alteration being detected)

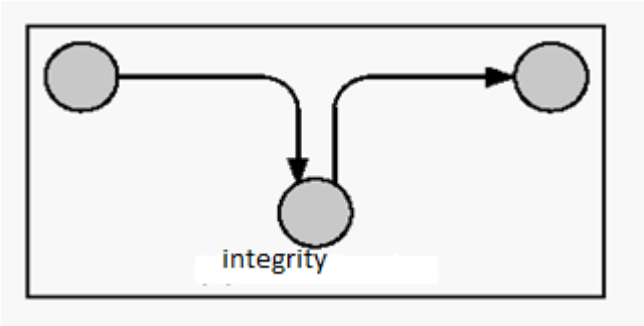

Figure1.1.2.2 Integrity

Availability of systems and data: System and data availability provides uninterrupted access by authorized users to important computing resources and data. (The creator/sender of the information cannot deny at a later stage his or her intentions in the creation or transmission of the information)

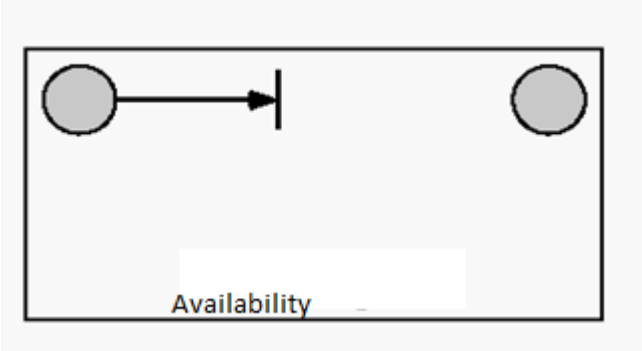

Figure1.1.2.3 Availability

Authentication: The sender and receiver can confirm each other's identity and the origin/destination of the information.

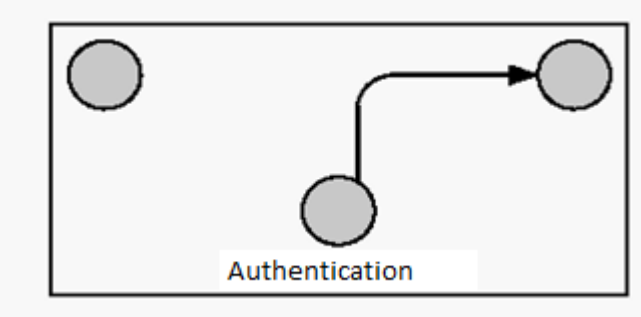

Figure 1.1.2.4 Authentication

Threats to network security include:

Viruses: Computer programs written by devious programmers and designed to replicate themselves and infect computers when triggered by a specific event

Trojan horse programs: Delivery vehicles for destructive code, which appear to be harmless or useful software programs such as games

Vandals: Software applications or applets that cause destruction

Attacks: Including reconnaissance attacks (information-gathering activities to collect data that is later used to compromise networks); access attacks (which exploit network vulnerabilities in order to gain entry to e-mail, databases, or the corporate network); and denial-of-service attacks (which prevent access to part or all of a computer system)

Data interception: Involves eavesdropping on communications or altering data packets being transmitted

Social engineering: Obtaining confidential network security information through nontechnical means, such as posing as a technical support person and asking for people's passwords

Network Security Tools Include:

Software packages: These packages counter most virus threats if regularly updated.

Secure network infrastructure: Switches and routers have hardware and software features that support secure connectivity, perimeter security, intrusion protection, identity services, and dedicated network security hardware and software-Tools such as firewalls and intrusion detection systems provide protection for all areas of the network and enable secure connections.

Virtual private networks: These networks provide access control and data encryption between two different computers on a network. This allows remote workers to connect to the network without the risk of a hacker or thief intercepting data.

Identity services: These services help to identify users and control their activities and transactions on the network. Services include passwords, digital certificates, and digital authentication Keys.

Encryption: Encryption ensures that messages cannot be intercepted or read by anyone other than the authorized persons.

Security management: This is the glue that holds together the other building blocks of a strong security solution.

None of these approaches alone will be sufficient to protect a network, but when they are layered together; they can be highly effective in keeping a network safe from attacks and other threats to security. In addition, well-thought-out corporate policies are critical to determine and control access to various parts of the network.

1.2 Cryptography

Cryptography is an important part of preventing private data from being stolen. Even if an attacker were to break into your computer or intercept your messages they still will not be able to read the data if it is protected by cryptography or encrypted. In addition to concealing the meaning of data, cryptography performs other critical security requirements for data including authentication, repudiation, confidentiality, and integrity.

Cryptography can be used to authenticate that the sender of a message is the actual sender and not an imposter. Encryption also provides for repudiation, which is similar to authentication, and is used to prove that someone actually sent a message or performed an action. For, instance it can used to prove a criminal performed a specific financial transaction.

Cryptography ensures confidentiality because only a reader with the correct deciphering algorithm or key can read the encrypted message. Finally, Cryptography can protect the integrity of information by ensuring that messages have not been altered.

The art of protecting information by transforming it into a unreadable format, called cipher text. Only those who possess a secret key can decipher the message into plain text. Encrypted messages can sometimes be broken by cryptanalysis, also called code breaking, although modern cryptography techniques are virtually unbreakable.

As the Internet and other forms of electronic communication become more prevalent, electronic security is becoming increasingly important. Cryptography is used to protect e-mail messages, credit card information, and corporate data. One of the most popular cryptography systems used on the Internet is Pretty Good Privacy because it's effective and free.

Cryptography systems can be broadly classified into symmetric-key systems that use a single key that both the sender and recipient have, and public-key systems that use two keys, a public key known to everyone and a private key that only the recipient of messages uses.

In cryptography systems there are two types of techniques available. They are

1. Text encryption

2. Image encryption

1.2.1 Text Encryption

Encryption is the process of scrambling the contents of a file or message to make it unintelligible to anyone not in possession of the "key" required to unscramble the file or message. There are two types of encryption: symmetric (private/secret) key and asymmetric(public) key encryption.

1.2.1.1 Symmetric Key Encryption

When most people think of encryption it is symmetric key cryptosystems that they think of. Symmetric key, also referred to as private key or secret key, is based on a single key and algorithm being shared between the parties who are exchanging encrypted information. The same key both encrypts and decrypts messages. The strength of the scheme is largely dependent on the size of the key and on keeping it secret. Generally, the larger the key, the more secure the scheme. In addition, symmetric key encryption is relatively fast.

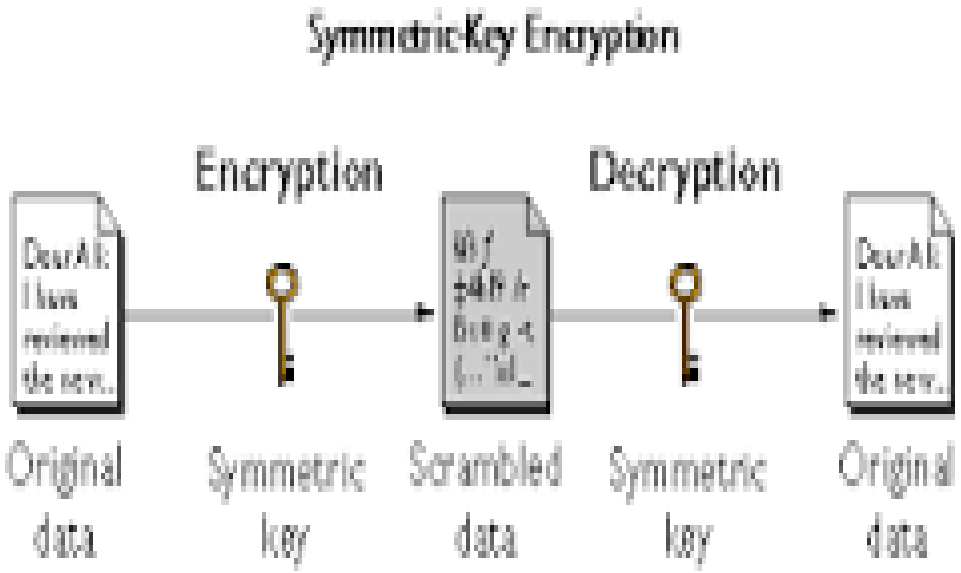

Figure1.2.1.1 Symmetric key Encryption

The main weakness of the system is that the key or algorithm has to be shared. You can't share the key information over an unsecured network without compromising the key. As a result, private key cryptosystems are not well suited for spontaneous communication over open and unsecured networks. In addition, symmetric key provides no process for authentication or non-repudiation. Remember, non-repudiation is the ability to prevent individuals or entities from denying (repudiating) that a message was sent or received or that a file was accessed or altered, when in fact it was. This ability is particularly important when conducting e-commerce. The lists the advantages and disadvantages of symmetric key cryptosystems.

Examples of widely deployed symmetric key cryptosystems include DES, IDEA, Blowfish, RC4, CAST, and SKIPJACK.

The Advantages and Disadvantages of Symmetric Key Cryptography

Advantages

1. Extremely Secure

When it uses a secure algorithm, symmetric key encryption can be extremely secure. One of the most widely-used symmetric key encryption systems is the U.S. Government-designated Advanced Encryption Standard. When you use it with its most secure 256-bit key length, it would take about a billion years for a 10 petaflop computer to guess the key through a brute-force attack. Since, as of November 2012, the fastest computer in the world runs at 17 petaflops, 256-bit AES is essentially unbreakable.

2. Relatively Fast

One of the drawbacks to public key encryption systems is that they need relatively complicated mathematics to work, making them very computationally intensive. Encrypting and decrypting symmetric key data is relatively easy to do, giving you very good reading and writing performance. In fact, many solid state drives, which are typically extremely fast, use symmetric key encryption internally to store data and they are still faster than unencrypted traditional hard drives.

3. Widely Understood

The symmetric key cryptography is simply understandable by people once they known the key.

Disadvantages

1. Sharing Key

The biggest problem with symmetric key encryption is that you need to have a way to get the key to the party with whom you are sharing data. Encryption keys aren't simple strings of text like passwords. They are essentially blocks of gibberish. As such, you'll need to have a safe way

to get the key to the other party. Of course, if you have a safe way to share the key, you probably don't need to be using encryption in the first place. With this in mind, symmetric key encryption is particularly useful when encrypting your own information as opposed to when sharing encrypted information.

2. More Damaged If Compromised

When someone gets their hands on a symmetric key, they can decrypt everything encrypted with that key. When you're using symmetric encryption for two-way communications, this means that both sides of the conversation get compromised. With asymmetrical public-key encryption, someone that gets your private key can decrypt messages sent to you, but can't decrypt what you send to the other party, since that is encrypted with a different key pair.

1.2.1.2 Asymmetric Key Encryption

For centuries, all cryptography was based on the symmetric key cryptosystems. Then in 1976,two computer scientists, Whitfield Diffie and Martin Hellman of Stanford University, introduced the concept of asymmetric cryptography. Asymmetric cryptography is also known as public key cryptography.

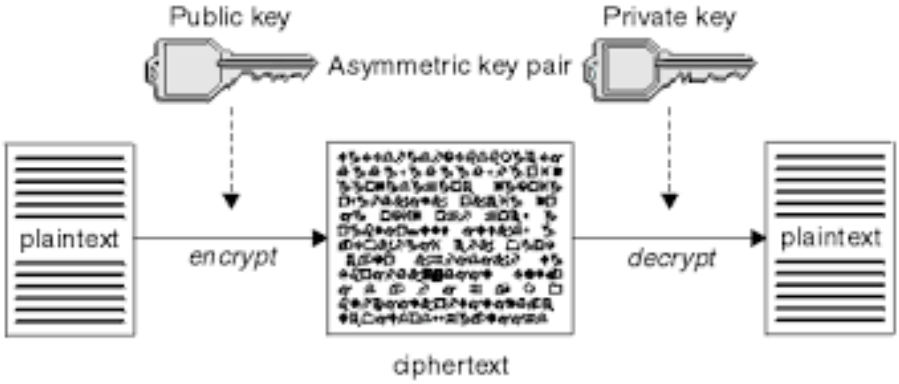

Figure 1.2.1.2 Asymmetric key Encryption

Public key cryptography uses two keys as opposed to one key for a symmetric system. With public key cryptography there is a public key and a private key. The keys' names describe their function. One key is kept private, and the other key is made public. Knowing the public key does not reveal the private key. A message encrypted by the private key can only be decrypted by the corresponding public key. Conversely, a message encrypted by the public key can only be decrypted by the private key.

With the aid of public key cryptography, it is possible to establish secure communications with any individual or entity when using a compatible software or hardware device. For example, if Alice wishes to communicate in a secure manner with Bob, a stranger with whom she has never communicated before, Alice can give Bob her public key. Bob can encrypt his outgoing transmissions to Alice with Alice's public key. Alice can then decrypt the transmissions using her private key when she receives them. Only Alice's private key can decrypt a message encrypted with her public key. If Bob transmits to Alice his public key, then Alice can transmit secure encrypted data back to Bob that only Bob can decrypt. It does not matter that they exchanged public keys on an unsecured network. Knowing an individual's public key tells you nothing about his or her private key. Only an individual's private key can decrypt a message1encrypted with his or her public key. The security breaks down if either of the parties' private keys is compromised. While symmetric key cryptosystems are limited to securing the privacy of information, asymmetric or public key cryptography is much more versatile.

Public key cryptosystems can provide a means of authentication and can support digital certificates. With digital certificates, public key cryptosystems can provide enforcement of non-repudiation. Unlike symmetric key cryptosystems, public key allows for

secure spontaneous communication over an open network. In addition, it is more scalable for very large systems (tens of millions) than symmetric key cryptosystems. With symmetric key cryptosystems, the key administration for

large networks is very complex. Summarizes the advantages and disadvantages of the public key cryptosystems.

The Advantages and Disadvantages of Public Key Cryptography

Advantages

- In asymmetric or public key, cryptography there is no need for exchanging keys, thus eliminating the key distribution problem.
- The primary advantage of public-key cryptography is increased security: the private keys do not ever need to be transmitted or revealed to anyone.
- Can provide digital signatures that can be repudiated

Disadvantages

- A disadvantage of using public-key cryptography for encryption is speed: there are popular secret-key encryption methods which are significantly faster than any currently available public-key encryption method.

Decryption is the process of transforming data that has been rendered unreadable through encryption back to its unencrypted form. In decryption the systems extracts and converts the garbled data and transforms it to texts that are easily understandable not only by the reader but also by the system. The user will encrypt the original text to given more security from unauthorised access parties. . Then the cipher will obtained. So by using this cipher the user decrypt, Then the recovered text will obtained. This process will give more security.

1.2.2 Image Encryption:

Unlike text messages, the multimedia information including image data has some special characteristics like high capacity, redundancy and high correlation among pixels. In some cases image applications require to satisfy their own needs like real time transmission and processing. One of the main goals that must be achieved during the transmission of information over the network is security. Cryptography is the technique that can be used for secure transmission of data. This technique will make the information to be transmitted into an unreadable form by encryption so that only authorized persons can correctly recover the information. The security of image can be achieved by various types of encryption schemes. Different chaos based and non-chaos based algorithms have been proposed. Among this the chaotic based methods are considered to be more promising. The chaotic image encryption can be developed by using properties of chaos including deterministic dynamics and unpredictable behaviour. There are three kinds of encryption techniques namely substitution, transposition or permutation and techniques that include both transposition and substitution. Substitution schemes change the pixel values while permutation schemes just shuffle the pixel values based on the algorithm. In some cases both the methods are combined to improve security. In an image encryption technique based on Arnold cat map and Chen's chaotic system is proposed. In combinations of three permutation techniques is described, in which bit level, pixel level and block level permutations are applied in some order. Image encryption in is an enhancement to AES algorithm by adding a key stream generator. The method in is chaos based using bit level permutation. Permutation at the bit level not only changes the position of the pixel but also alters its value. In a novel image encryption method based on total shuffling scheme is illustrated. In combinations of two logistic maps are used for improving the security of encryption.

1.3 Steganography:

Steganography is the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity. The word steganography is of Greek origin and means "concealed writing" from the Greek words steganos meaning "covered or protected", and graphic meaning "writing".

The first recorded use of the term was in 1499 by Johannes Trithemius in his Steganography, a treatise on cryptography and steganography disguised as a book on magic. Generally, messages will appear to be something else: images, articles, shopping lists, or some other cover text and, classically, the hidden message may be in invisible ink between the visible lines of a private letter. It is high security technique for long data transmission.

The advantage of steganography over cryptography alone is that messages do not attract attention to themselves. Plainly visible encrypted messages no matter how unbreakable will arouse suspicion, and may in themselves be incriminating in countries where encryption is illegal. Therefore, whereas cryptography protects the contents of a message, steganography can be said to protect both messages and communicating parties.

Steganography includes the concealment of information within computer files. In digital steganography, electronic communications may include steganographic coding inside of a transport layer, such as a document file, image file, program or protocol. Media files are ideal for steganographic transmission because of their large size. As a simple example, a sender might start with an innocuous image file and adjust the colour of every 100th pixel

to correspond to a letter in the alphabet, a change so subtle that someone not specifically looking for it is unlikely to notice it.

Some more points about steganography

1) Steganography is the science of hiding secret information in an unsuspicious cover object. The goal of Steganography was defined by Johnson and Jajodia as "the goal of steganography is to avoid drawing suspicion to the transmission of a hidden message. If suspicion is raised, then this goal is defeated.

2) Discovering the communication is the first attack to steganography since it is against the main goal of steganography. But it is possible to use steganography together with cryptography by encrypting the message before embedding it into a cover object. Using steganography and cryptography together to provide better information security is a hot topic

3) Steganography techniques can be categorized as 'fragile' and 'robust' according to the strength of the stego-object against the steganalysis attacks. The stego-object, which does not lose its hidden message and its cover object is still recognizable after being exposed to multiple image processing techniques such as warping, cropping, rotating and blurring is defined as robust. The others are fragile. Their hidden messages are lost under JPEG compression or they are completely destroyed after applying image processing techniques

- **Steganography in Text Files**

Hiding messages in text files is the easiest and oldest but a fragile way of secret communication. Today, it is applied by changing the layout of a document, adding extra spaces and using hidden characters in text. The disadvantage of text steganography is its weakness against attacks. The extra spaces, lines and characters added could easily be detected by opening the

text in a word processor. The hidden message will be lost if the document is reformatted. Additional techniques for hiding messages in text files are given in.

- **Steganography in Image Files**

Images are the most common cover objects used in steganography. Image steganograms can be fragile or robust according to the method applied on the image file. Image steganography methods can be categorized as 'Image domain methods' and 'Transform domain methods'. Image domain methods are easy to apply, but it is possible to create more robust stego-images with transform domain techniques. Some of the popular image domain tools are 'Hide and Seek', 'Mandelsteg', 'Steganos', 'StegoDos', 'STOOLS', and 'White Noise Storm'. Some of the transform domain tools are 'Jpeg-Jsteg', 'JPHide', 'Outguess', 'PictureMarc' and 'SysCop' .

Some of the techniques were briefly explained as follows:

1) Least Significant Bit Algorithm: This is the simplest method applied on image files. First, the message to be hidden is broken into pieces of 1 bit then the least significant bit of each pixel of the cover object is used to store the bits of the secret message. The change in cover object is invisible to naked eyes up to 4th LSB of the image. This technique is unsuccessful and visible to naked eyes, when the bits of the hidden message have more space to be placed than the cover image. The LSB method is not a robust algorithm since it is easily corrupted when it is exposed to image processing techniques.

2) Patchwork Algorithm: It is a more complex method compared to LSB algorithm. First, two random pixels are selected from the image. Then, the brighter of the two is made brighter and the darker one is darker. The contrast change between these two pixels corresponds to a part of the bits of the hidden message. The image remains undetectable under filtering attacks, even in the case of a few hundred changes in pixels.

3) Transform Domain Algorithms: Robust methods use transformation algorithms such as DCT (Discrete Cosine Transformation) or Wavelet Transformation. The message is hidden in significant areas of the cover image using the algorithms which make the stego-object more robust to image processing attacks than the LSB method. A more detailed research on transform domain techniques can be found in.

About The Project:

The project “ Encryption Based Steganography-Modern Approach for Information Security”, The word steganography is of Greek origin and means "concealed writing" from the Greek words steganos meaning "covered or protected", and graphic meaning "writing". In this project we use network security concepts to send the information through network media and use the cryptography technique to encrypt the message to give security to the data. By using the steganography technique to hide the information in image to give more security to the information.

2. LITERATURE SURVEY

2.1 INTRODUCTION

Steganography is the science of hiding secret information in an unsuspicious cover object. The goal of Steganography was defined by Johnson and Jajodia as "the goal of steganography is to avoid drawing suspicion to the transmission of a hidden message. If suspicion is raised, then this goal is defeated." Discovering the communication is the first attack to steganography since it is against the main goal of steganography. But it is possible to use steganography together with cryptography by encrypting the message before embedding it into a cover object. Using steganography and cryptography together to provide better information security is a hot topic.

Steganography techniques can be categorized as 'fragile' and 'robust' according to the strength of the stego-object against the steganalysis attacks. The stego-object, which does not lose its hidden message and its cover object is still recognizable after being exposed to multiple image processing techniques such as warping, cropping, rotating and blurring is defined as robust. The others are fragile. Their hidden messages are lost under JPEG compression or they are completely destroyed after applying image processing techniques.

Steganography is the art and science of hiding communication; a steganographic system thus embeds hidden content in unremarkable cover media so as not to arouse an eavesdropper's suspicion. In the past, people used hidden tattoos or invisible ink to convey steganographic content. Today, computer and network technologies provide easy-to-use communication channels for steganography. Essentially, the information-hiding process in a steganographic system starts by identifying a cover medium's redundant bits (those that can be modified without destroying that medium's integrity). The embedding process creates a stego medium by replacing these redundant bits with data from the hidden message. Modern steganography's goal is to keep its mere presence undetectable, but steganographic system because of their

invasive nature leave behind detectable traces in the cover medium. Even if secret content is not revealed, the existence of it is: modifying the cover medium changes its statistical properties, so eavesdroppers can detect the distortions in the resulting stego medium's statistical properties. The process of finding these distortions is called statistical steganalysis. An introduction to steganography[1]

But in the above process the data is directly embedding into the cover image and the data embed in image pixels in bits in continuous pixels. Once hacker knows one of the cover image and he gets over data very easily. In our project the data is embed in randomly selected pixels by doing some special operations on pixel values and give more security to the data as well as cover image and the quality of the image not reduced

The researches on new hiding methods and cover objects which hidden information is embedded in. It is the result from the researches to embed information in executable files, but when will use the executable file for cover they have many challenges must be taken into consideration which is any changes made to the file will be firstly detected by untie viruses , secondly the functionality of the file is not still functioning. In this paper, a new information hiding system is presented. The aim of the proposed system is to hide information (data file) within image page of execution file (EXEfile) to make sure changes made to the file will not be detected by universe and the functionality of the exe.file is still functioning after hiding process. Meanwhile, since the cover file might be used to identify hiding information, the proposed system considers overcoming this dilemma by using the execution file as a cover file. New Design for Information Hiding with in Steganography Using Distortion Techniques[2]

In the above process the secret information is embedding into an executable file. But it is difficult to get our original secret information from covered executable file. But in our project we retrieve the original secret information from cover image is not difficulty.

Steganography is a process that involves hiding a message in an appropriate carrier for example an image or an audio file. The carrier can then be sent to a receiver without anyone else knowing that it contains a hidden message. This is a process, which can be used for example by civil rights organisations in repressive states to communicate their message to the outside world without their own government being aware of it. Less virtuously it can be used by terrorists to communicate with one another without anyone else's knowledge. In both cases the objective is not to make it difficult to read the message as cryptography does, it is to hide the existence of the message in the first place possibly to protect the courier. The initial aim of this study was to investigate steganography and how it is implemented. Based on this work a number of common methods of steganography could then be implemented and evaluated. The strengths and weaknesses of the chosen methods can then be analysed. To provide a common frame of reference all of the steganography methods implemented and analysed used images. An Evaluation of Image Based Steganography Methods[3]

In the above process doing one of many steganography techniques. But each technique has their own drawbacks for giving security to the information as well as covered image.

Cryptography and Steganography are the two fields available for data security. Cryptography is a technique in which the data is scrambled in an unintelligent gibberish fashion so that it becomes difficult for any malicious user to extract the original message. Only the desired recipient will be having the code for decryption and will be able to extract messages. Cryptography has helped a great deal in data security but it has some disadvantages. The encrypted data will arouse suspicion to malicious users and there is a possibility of it being decrypted or being suppressed. Hence the intended information might not reach its destination effectively. The disadvantages of

Cryptography have lead to the development of Steganography. The most well-known steganographic technique in the data hiding field is least-significant-bits (LSBs) substitution. This method embeds the fixed-length secret bits in the same fixed length LSBs of pixels. Although this technique is simple, it generally causes noticeable distortion when the number of embedded bits for each pixel exceeds three. Several adaptive methods for steganography have been proposed to reduce the distortion caused by LSBs substitution. For example, adaptive methods vary the number of embedded bits in each pixel, and they possess better image quality than other methods using only simple LSBs substitution. However, this is achieved at the cost of a reduction in the embedding capacity. A Comparative Analysis of Image Steganography[4]

In the above process the secret message is embed into image by doing LSB substitution. Once the hacker knows one of the modified pixel he detects all modified pixels in covered image. But in our project we generate random pixels and doing special operations to embed data in pixels to give more security to the secret information.

The proposed method hides the secret message based on searching about the identical bits between the secret messages and image pixels values. It is implemented to hide the secret message. The proposed method is efficient, simple and fast it robust to attack and improve the image quality. In this a new Steganography technique was presented, implemented and analyzed. The proposed method hides the secret message based on searching about the identical bits between the secret messages and image pixels values. The This paper conclude that the proposed method is more efficient, simple, appropriate and accurate than LSB method, it search about the identical then start hiding, hence the change in the image resolution is quite low, as well as it makes the secret message more secure. a new method in image steganography with improved image quality[5]

In the above process the secret information is directly stored in image pixels by using LSB algorithm. There is less security to the data when the

hacker finds the pixel values. But in our project the secret information is converted into cipher text and embed into secret cover image for giving more security.

2.2 CURRENT SYSTEM

In existing system, use LSB(Least Significant Bit) replacement algorithm. In LSB makes the changes in the image resolution quite clear as well as it is easy to attack. LSB hiding technique hide the secret message directly in the least two significant bits in the image pixels, and the pixels take in continuous manner, hence that affect the image resolution, which reduce the image quality and make the image easy to attack. As well as this method is already has been attacked and broken.

2.3 PROBLEM STATEMENT

Steganography is an art of science of covered writing. The secret information is covered with an unsuspicious media. Previously the secret information is directly embed into covered image. But there is less security to the given information.

Now we introduce a novel method for giving more security to the information. In this first we do the encryption for secret information and take a image. now we generate random pixels and do some mathematical operations on pixel values for difficult to know hacker and embed the cipher text in image. It gives more security to the secret information.

2.4 PROPOSED SYSTEM

Proposed algorithm able to make the secret message more secure because we convert the secret message into cipher text by doing encryption process and enhance the quality of the image by doing the steganography process. The proposed method hides the secret cipher text by doing some special operations on pixel values and we change the dark pixel as very dark

and brighter is changed as very brighter pixel. In the proposed system the image quality is not changed and the secret information is securely transmitted from sender to receiver.

2.5 OBJECTIVES

- Steganography is the science of hiding secret information in an unsuspicious cover object.
- In previous days the secret information is directly embed in image pixel values by the use of LSB algorithm
- But there is less security to the information. Then we introduce a novel approach for giving more security to the information which we send to receiver.
- We generate random pixels and embed the secret information after doing the encryption process.
- It gives more security to the data.

2.6 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

A **Requirement** is a feature that the system must have or a constraint that it must satisfy to be accepted by the clients. Requirements Engineering aims at defining the requirements of the system under construction. It includes two main activities namely *Requirements Elicitation* and *Analysis*.

Requirements Elicitation focuses on describing the purpose of the system. The Client, the Developer, and the Users identify a problem area and define a system that addresses the problem. Such a definition is called **Requirements Specification**. This specification is structured and formalized during analysis to produce an *Analysis Model*.

Functional Requirements:

The functional requirement of the system defines a function of software system or its components. A function is described as a set of inputs, behaviour of a system and output.

The functional requirements are:

- **SENDER :**

Inputs

Image, Original Secret Message

Output

Embedded Image

- **RECEIVER :**

Inputs

Embedded Image, Original image

Output

Secret Message

Non-Functional Requirements

User Interfaces and Characteristics

- **Usability:**

This system provides an easy interface to the Sender as well as the

Receiver. They are able to view the plain text or data in a neat manner.

- **Reliability:**

The system work reliably even if they are large text to be transferred

❖ **Performance:**

It is assured with the getting good response time

❖ **Supportability:**

The system run on any platform having JAVA Net beans provided.

❖ **Implementation:**

The system easily implemented using JAVA Net beans

❖ **Security:**

This system provides security to the image that is being transferred.

Hardware Requirements:

- Processor : Pentium IV or above
- RAM : 512MB or above
- Hard disk : 80 GB Min.
- Monitor : User choice
- Keyboard : Standard.
- Mouse : Standard

Software Requirements

- Operating System : Windows XP and above.
- Front-End/ GUI Tool : JAVA Net Beans
- Programming Language : Java (jdk1.7.0_05)

2.7 ABOUT JAVA

Java Technology:

Java technology is both a programming language and a platform.

The Java Programming Language:

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

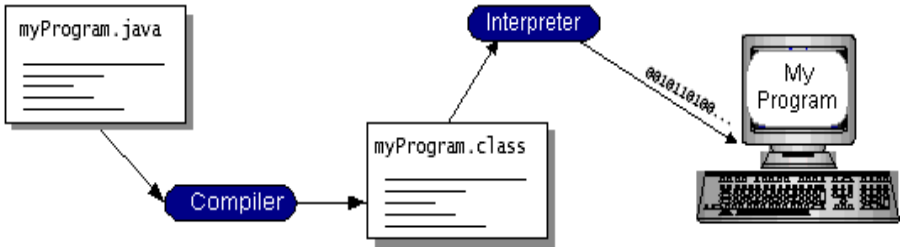

Figure 2.7.1- WORKING OF JAVA

You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

The Java Platform:

A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The *Java Virtual Machine* (JVM)
- The *Java Application Programming Interface*(Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms. The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. The next section, What Can Java Technology Do? highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

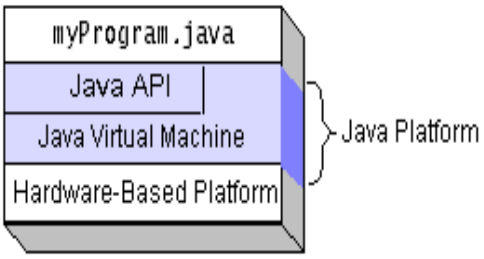

Figure 2.7.2- THE JAVA PLATFORM

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time bytecode compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do? :

The most common types of programs written in the Java programming language are *applets* and *applications*. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components those provide a wide range of functionality. Every full implementation of the Java platform gives you the following features:

The essentials: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
Applets: The set of conventions used by applets.

Networking: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.

Internationalization: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.

Security: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.

Software components: Known as JavaBeans, can plug into existing component architectures.

Object serialization: Allows lightweight persistence and communication via Remote Method Invocation (RMI).

It provides uniform access to a wide range of relational databases. The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.

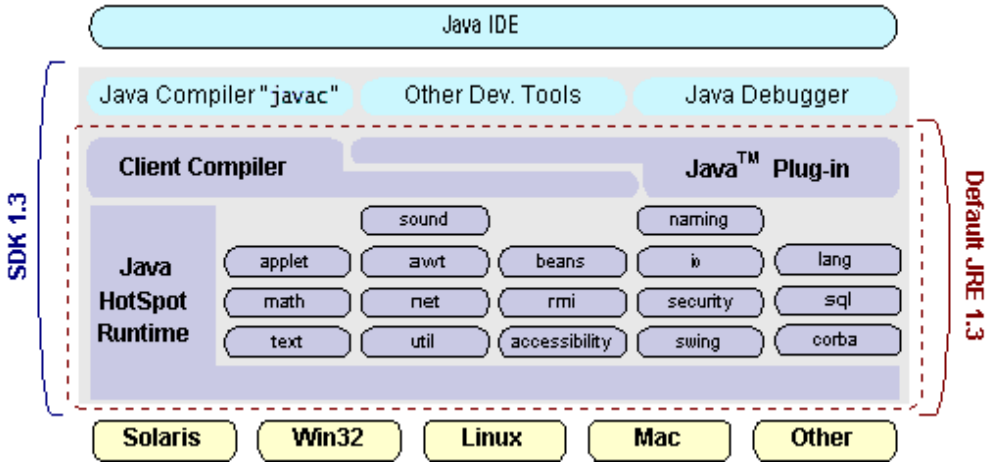

Figure 2.7.3 – JAVA 2 SDK

NetBeans:

NetBeans is an integrated development environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5. It is also an application platform framework for Java desktop applications and others.

The NetBeans IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM.

The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans Platform (including the NetBeans IDE itself) can be extended by third party developers.

NetBeans Platform:

The NetBeans Platform is a reusable framework for simplifying the development of Java Swing desktop applications. The NetBeans IDE bundle for Java SE contains what is needed to start developing NetBeans plugins and NetBeans Platform based applications; no additional SDK is required.

Applications can install modules dynamically. Any application can include the Update Centre module to allow users of the application to download digitally signed upgrades and new features directly into the running application. Reinstalling an upgrade or a new release does not force users to download the entire application again.

The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application. The features of the platform are:

- User interface management (e.g. menus and toolbars)
- User settings management
- Storage management (saving and loading any kind of data)

- Window management
- Wizard framework (supports step-by-step dialogs)
- NetBeans Visual Library
- Integrated development tools

Integrated modules:

NetBeans Profiler:

The NetBeans Profiler is a tool for the monitoring of Java applications: It helps developers find memory leaks and optimize speed. Formerly downloaded separately, it is integrated into the core IDE since version 7.4.

The Profiler is based on a Sun Laboratories research project that was named JFluid. That research uncovered specific techniques that can be used to lower the overhead of profiling a Java application. One of those techniques is dynamic byte code instrumentation, which is particularly useful for profiling large Java applications. Using dynamic byte code instrumentation and additional algorithms, the NetBeans Profiler is able to obtain runtime information on applications that are too large or complex for other profilers. NetBeans also support Profiling Points that let you profile precise points of execution and measure execution time.

GUI design tool:

Formerly known as project Matisse, the GUI design-tool enables developers to prototype and design Swing GUIs by dragging and positioning GUI components. The GUI builder has built-in support for JSR 295 (Beans Binding technology), but the support for JSR 296 (Swing Application Framework) was removed in, 7.1.

NetBeans JavaScript editor:

The NetBeans JavaScript editor provides extended support for JavaScript, Ajax, and CSS. JavaScript editor features comprise syntax highlighting, refactoring, code completion for native objects and functions, generation of JavaScript class skeletons, generation of Ajax callbacks from a template; and automatic browser compatibility checks.

CSS editor features comprise code completion for styles names, quick navigation through the navigator panel, displaying the CSS rule declaration in a List View and file structure in a Tree View, sorting the outline view by name, type or declaration order (List & Tree), creating rule declarations (Tree only), refactoring a part of a rule name (Tree only).

NetBeans IDE Bundle for Java ME:

The NetBeans IDE Bundle for Java ME is a tool for developing applications that run on mobile devices; generally mobile phones, but this also includes entry-level PDAs, and Java Card, among others.

The NetBeans IDE comes bundled with the latest Java ME SDK 3.0 which supports both CLDC and CDC development. One can easily integrate third-party emulators for a robust testing environment. You can download other Java platforms, including the Java Card Platform 3.0, and register them in the IDE.

NetBeans IDE Bundle for PHP:

- Syntax highlighting, code completion, occurrence highlighting, error highlighting.
- Semantic analysis with highlighting of parameters and unused local variables.
- PHP code debugging with xdebug.
- PHP Unit testing with PHPUnit and Selenium.

- Code coverage.
- Symfony framework support.
- Zend Framework support.
- PHP 5.3 namespace and closure support.
- Code Folding for Control Structures.

NetBeans IDE Complete Bundle:

Sun Microsystems also releases a version of NetBeans that includes all of the features of the above bundles. This bundle includes:

- NetBeans Base IDE
- Java SE, JavaFX
- Web and Java EE
- Java ME
- C/C++
- PHPGlassFish
- Apache Tomcat

3. UML MODELING

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.UML was created by Object Management Group and UML 1.0 specification draft was proposed to the OMG in January 1997.

OMG is continuously putting effort to make a truly industry standard.

- UML stands for Unified Modeling Language.
- UML is different from the other common programming languages like C++, Java, and COBOL etc.
- UML is a pictorial language used to make software blue prints.

So UML can be described as a general purpose visual modeling language to visualize, specify, construct and document software system. Although UML is generally used to model software systems but it is not limited within this boundary. It is also used to model non software systems as well like process flow in a manufacturing unit etc.

UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization UML is become an OMG (Object Management Group) standard.

Goals of UML:

A picture is worth a thousand words, this absolutely fits while discussing about UML. Object oriented concepts were introduced much earlier than UML. So at that time there were no standard methodologies to organize and consolidate the object oriented development. At that point of time UML came into picture.

There are a number of goals for developing UML but the most important is to define some general purpose modeling language which all modelers can use and also it needs to be made simple to understand and use.

UML diagrams are not only made for developers but also for business users, common people and anybody interested to understand the system. The system can be a software or non software. So it must be clear that UML is not a development method rather it accompanies with processes to make a successful system.

At the conclusion the goal of UML can be defined as a simple modeling mechanism to model all possible practical systems in today's complex environment.

UML is popular for its diagrammatic notations. We all know that UML is for visualizing, specifying, constructing and documenting the components of software and non software systems. Here the Visualization is the most important part which needs to be understood and remembered by heart.

3.1.1 UML Standard Diagrams:

The elements are like components which can be associated in different ways to make a complete UML pictures which is known as diagram. So it is very important to understand the different diagrams to implement the knowledge in real life systems.

Any complex system is best understood by making some kind of diagrams or pictures. These diagrams have a better impact on our understanding. So if we look around then we will realize that the diagrams are not a new concept but it is used widely in different form in different industries.

We prepare UML diagrams to understand a system in better and simple way. A single diagram is not enough to cover all aspects of the system. So UML defines various kinds of diagrams to cover most of the aspects of a system.

You can also create your own set of diagrams to meet your requirements. Diagrams are generally made in an incremental and iterative way.

There are two broad categories of diagrams and then are again divided into sub-categories:

- Structural Diagrams
- Behavioural Diagrams

Structural Diagrams:

The structural diagrams represent the static aspect of the system. These static aspects represent those parts of a diagram which forms the main structure and therefore stable.

These static parts are represents by classes, interfaces, objects, components and nodes. The four structural diagrams are:

- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

Behavioural Diagram

Any system can have two aspects, static and dynamic. So a model is considered as complete when both the aspects are covered fully.

Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system.

UML has the following five types of behavioral diagrams:

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

3.2 Use Case Diagram

A usecase illustrates a unit of functionality provided by the system. The main purpose of the usecase diagram is to help development teams visualize the functional requirements of a system, including the relationship of "actors" (human beings who will interact with the system) to essential processes, as well as the relationships among different usecases. Use case diagrams generally show groups of usecases either all usecases for the complete system, or a breakout of a particular group of usecases with related functionality (e.g., all security administration related usecases). To show a usecase on a usecase diagram, you draw an oval in the middle of the diagram and put the name of the usecase in the center of, or below, the oval. To draw an actor (indicating a system user) on a use case diagram, you draw a stick person to the left or right of your diagram. Use simple lines to depict relationships between actors and use cases.

3.2.1. Actors

Actor:

Actors are external entities that interact with the system. Actors typically include a user role or another system. They have unique names and descriptions.

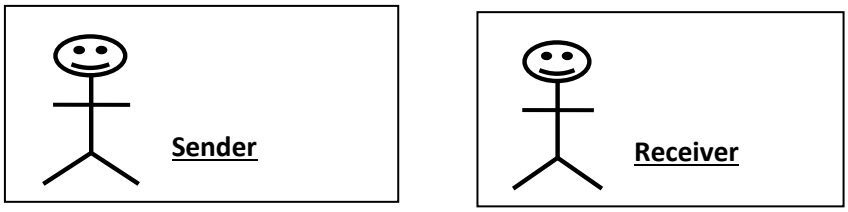

Figure 3.2.1: Actors – Sender and Receiver

Actors Description:

Sender:

Sender will browse the Image from particular Location and browse or type the information which we want send. Now encrypt the information. Now sender will embed the cipher text into the selected cover image

Receiver:

Receiver will browse the embedded image as well as original from particular location. And get all modified pixels and get the cipher text. Now receiver decrypt the cipher text and get original information.

Use Case:

Use cases describe the behaviour of the system as seen from the actor's point of view. It describes the function provided by the system as a set of events that yield a visible result for the actors.

Actors initiate the use cases for accessing system's functionality. When actors and use Cases exchange information, they are said to Communicate. To describe a use case we use a template composed of six fields:

Use Case Name	:	The name of the use case
Participating Actors	:	The actors participating in the particular use case
Entry Condition	:	Condition for initiating the use case.
Flow of events	:	Sequence of steps describing the functioning of use case.
Exit Condition	:	Condition for terminating the use case.
Quality Requirements	:	Requirements that do not belong to the use case but constraint the functionality of the system.

Use case diagrams include four types of relations. They are as follows:

- Communication Relationships
- Inclusion Relationships
- Extension Relationships
- Inheritance Relationships

3.2.2 USE CASE DIAGRAM:

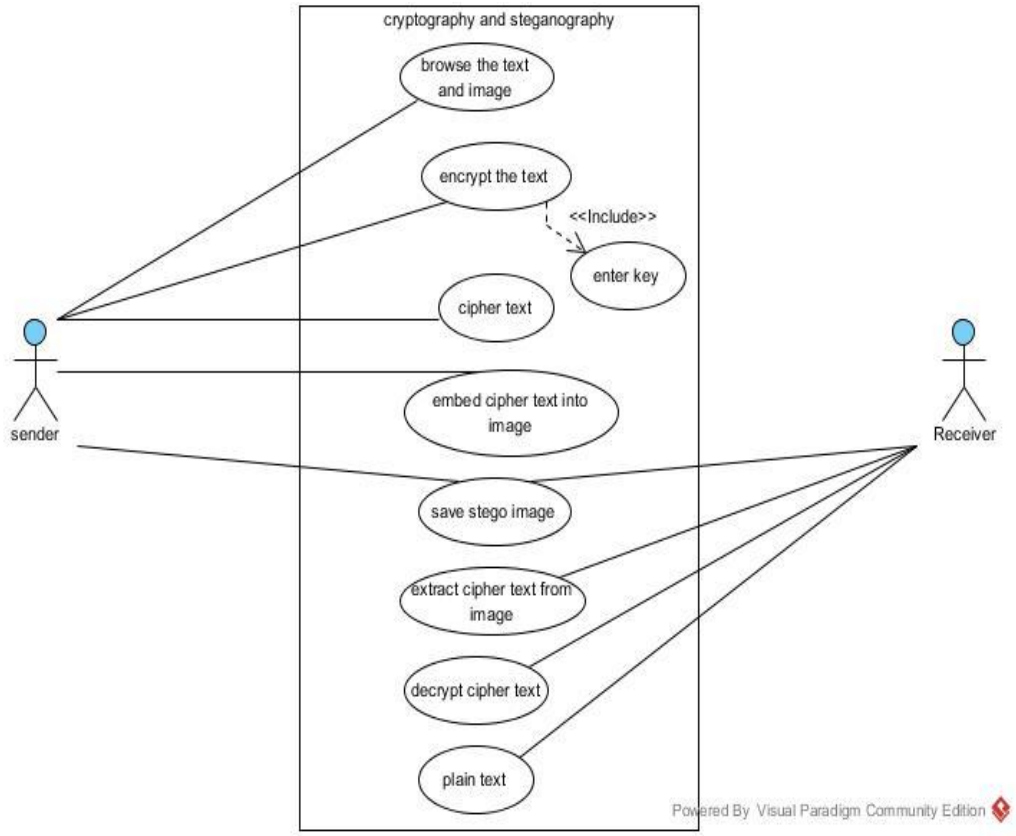

Figure 3.2.2 Use Case Diagram

Description:

In the use case diagram the sender will browse the image and secret message file from the respective source file then encrypt the secret message and embed the cipher text into the cover image. Now we save the stego image and send to Receiver. Now the receiver will browse the Stego image and original cover image. Compare all pixels in to images and get all modified pixels. From that modified pixels Extract cipher text and Decrypt the cipher text and get Original message.

1. Select Image and Message Use Case:

Use Case Name	Select Image, Secret message file
Participating Actors	Sender
Entry Condition	Sender browse the image and secret message from a particular condition
Flow of Events	1.Browse the image and secret message from particular location 2 Display the image and secret message on the screen
Exit Condition	1.Save the embed image on the particular Location

Table 3.2.1 Select Image and Message Use Case

2. Encryption Use Case:

Use Case Name	Encryption
Participating Actors	Sender
Entry Condition	Browse the image and secret message file
Flow of Events	1.Select the image and secret message file by using browse method 2.Encrypt the secret message and embed the cipher text into image. 3.Save the stego-image in a particular location.
Exit Condition	Obtain the Stego-image.

Table 3.2.2 Encryption Use Case

3. Decryption Use Case:

Use Case Name	Decryption
Participating Actors	Receiver
Entry Condition	Browse the Stego-image and original cover image from particular location
Flow of Events	1.Browse the stego-image and original image. 2.In decryption process we have to browse two images and check each pixel and get all modified pixels. Now receiver get cipher text. 3.Decrypt the cipher text and get original secret message.
Exit Condition	Obtain the original secret message.

Table 3.2.3 Decryption Use Case

Scenario:

A Scenario is an instance of a use case describing a concrete set of actions. Scenarios are used as examples for illustrating common cases: their focus is on understandability.

A Use Case Scenario is a description that illustrates, step by step, how a user is intending to use a system, essentially capturing the system behavior from the user's point of view. A use case scenario can include stories, examples, and drawings. Use cases are extremely useful for describing the problem domain in unambiguous terms and for communicating with the potential users of a system.

Scenario Name : The name of the Scenario

Participating Actors : The Instances of the actors participating in the Scenario.

Flow of events : Sequence of steps describing the events in Scenario.

Scenario-1:

Scenario Name	Encryption
Participating Actors	Sender
Flow of Events	In this phase sender encrypt the secret message. The cipher text is embed into image by doing some special operations and send the data embed image to receiver.

Table 3.2.4 Encryption Scenario

Scenario-2:

Scenario Name	Decryption
Participating Actors	Receiver
Flow of Events	In this phase receiver browse two images and check all pixels and get modified pixels and get the cipher text. Now receiver decrypt the cipher text and get original secret message.

Table 3.2.5 Decryption Scenario

3.3 About Class Diagram:

Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation. Classes are composed of three things: a name, attributes, and operations. Class diagrams also display relationships such as containment, inheritance, associations and others. The association relationship is the most common relationship in a class diagram. The association shows the relationship between instances of classes. The multiplicity of the association denotes the number of objects that can participate in the relationship.

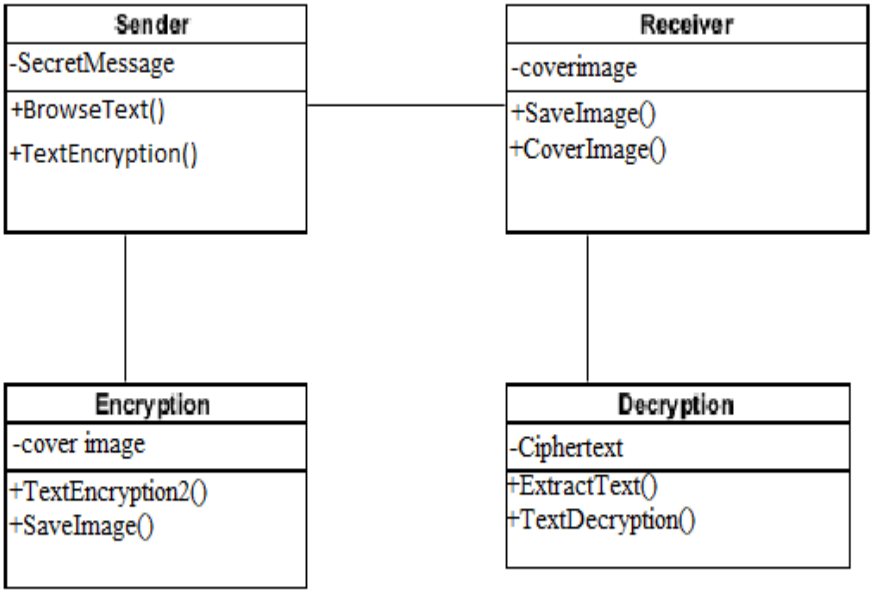

Figure 3.3.1 Class diagram

3.4 About Sequence Diagrams

Interaction between objects can be described by means of sequence diagrams. An object interacts with another object by sending messages. The reception of a message by an object triggers the execution of an operation, which in turn may send messages to other objects. Arguments may be passed along with a message and are bound to the parameters of the executing operation in the receiving object. Typically, we use a sequence diagram to the event flow of a use case, identify the objects that participate in the use case, and assign pieces of the use case behaviour to the objects in the form of services.

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.

Encryption: In this phase we take a secret information and encrypt the information. Select a cover image, and embed the cipher text into the cover image. Save the stego-image and send to receiver.

Decryption: In this phase we take stego-image as well as original image and check each pixel and get all modified pixels. Now we get cipher text and decrypt the cipher text and get original information.

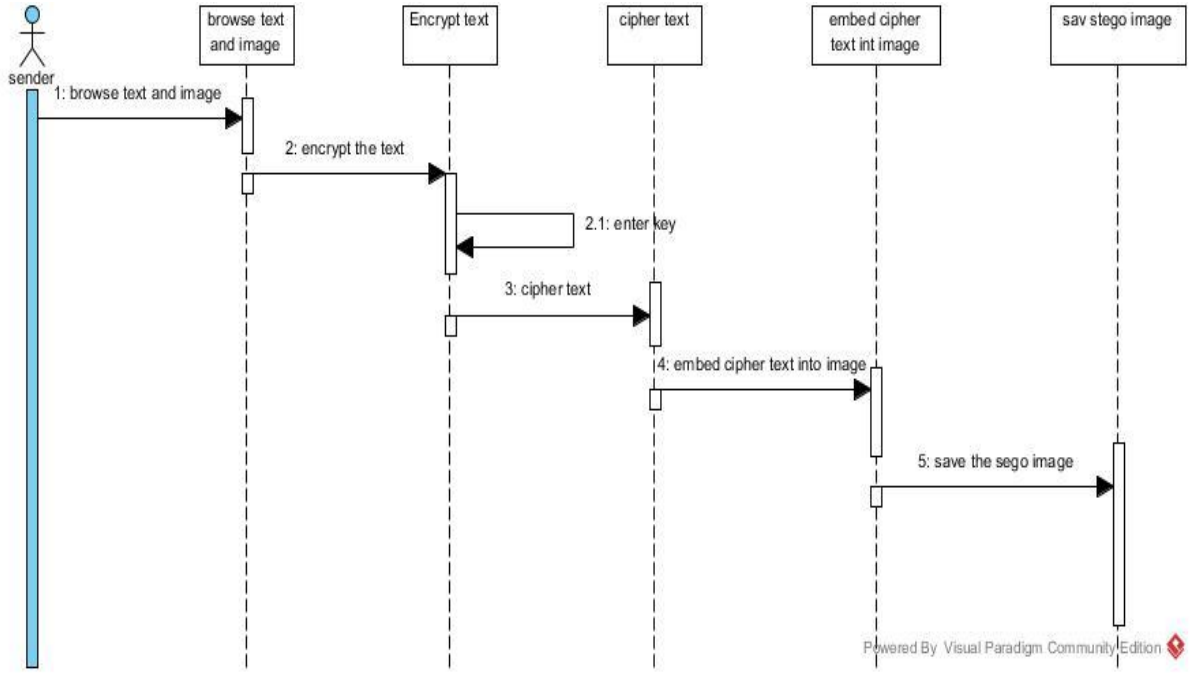

Figure 3.4.1 Sequence diagram for Sender

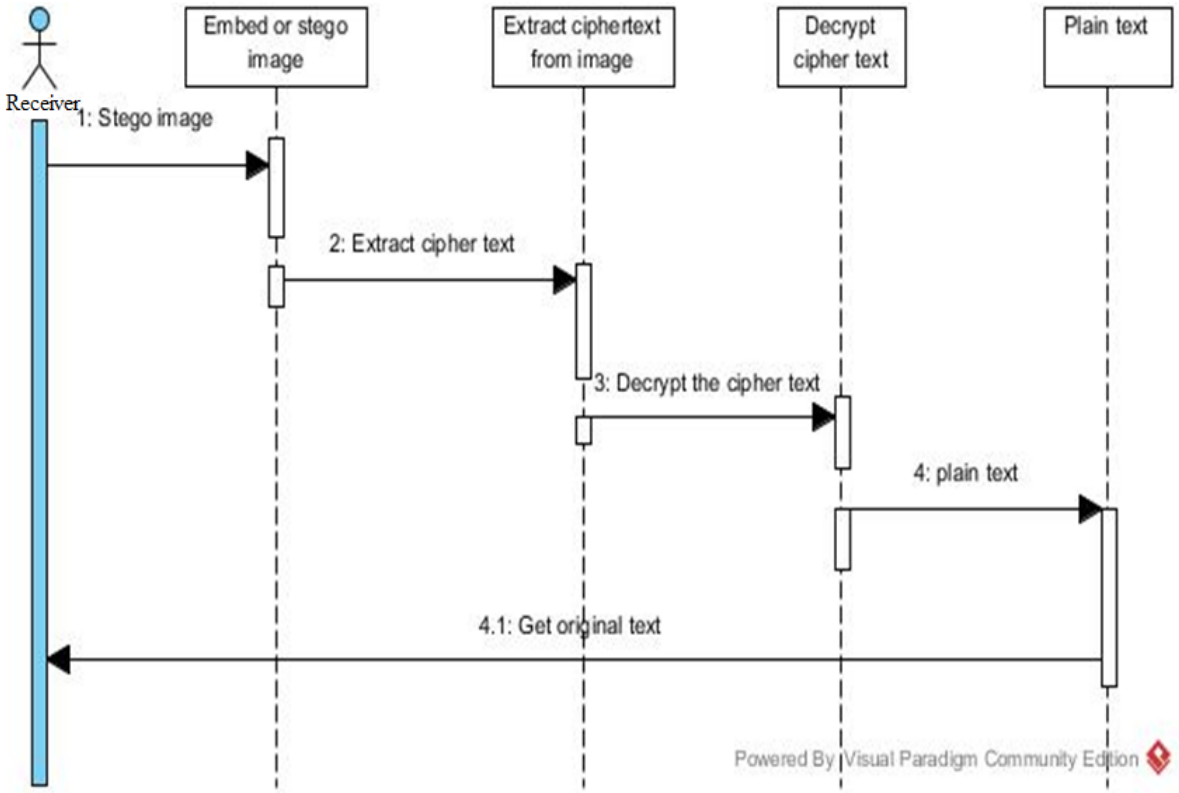

Figure 3.4 2 sequence diagram for Receiver

3.5 About State-Chart Diagram:

State diagrams are used to describe the behaviour of a system. State diagrams describe all of the possible states of an object as events occur. Each diagram usually represents objects of a single class and tracks the different states of its objects through the system. Not all classes will require a state diagram and state diagrams are not useful for describing the collaboration of all objects in a use case. State diagrams have very few elements. The basic elements are rounded boxes representing the state of the object and arrows indicting the transition to the next state. The activity section of the state symbol depicts what activities the object will be doing while it is in that state. All state diagrams being with an initial state of the object.

Encryption: In this phase we take a secret information and encrypt the information. Select a cover image, and embed the cipher text into the cover image. Save the stego-image and send to receiver.

Decryption: In this phase we take stego-image as well as original image and check each pixel and get all modified pixels. Now we get cipher text and decrypt the cipher text and get original information.

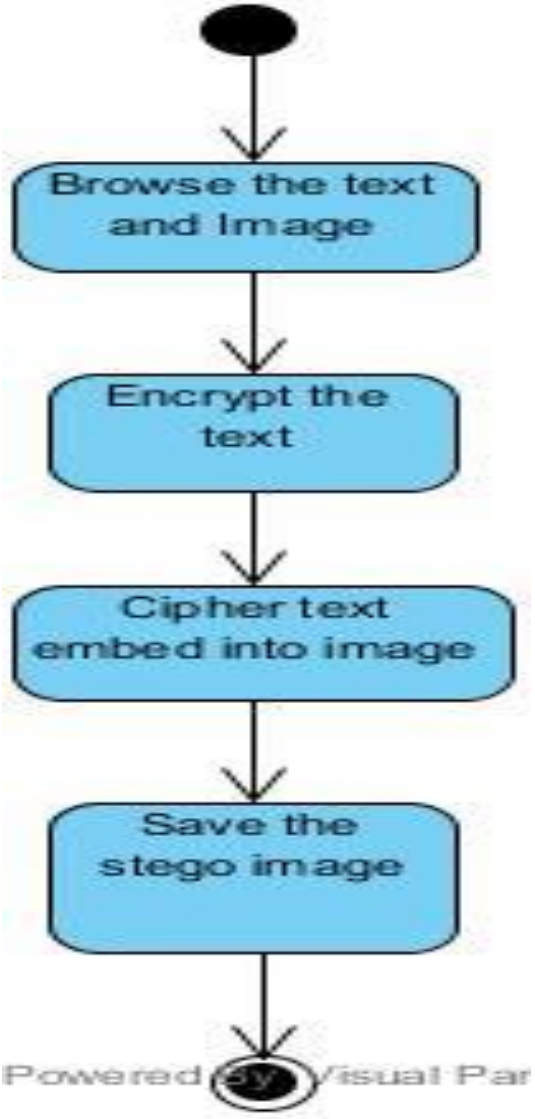

Figure3.5.1 State diagram for Sender

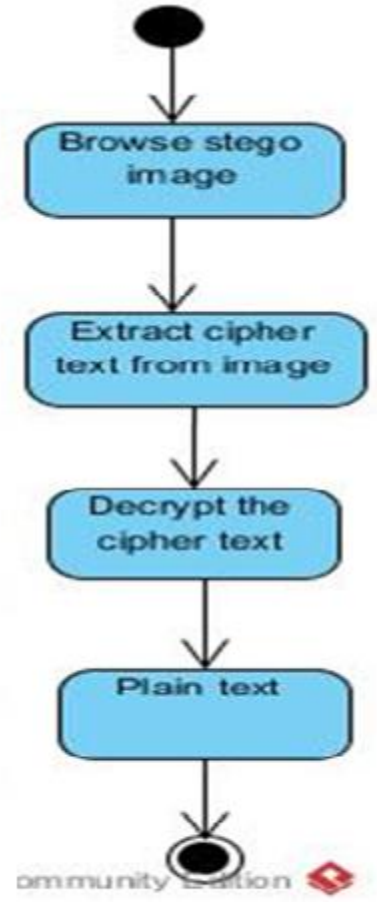

Figure3.5.2 State diagram for Receiver

3.6 Component Diagram:

Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces or collaboration .So Component diagrams represent the implementation view of a system.

During design phase software artifacts (classes, interfaces etc) of a system are arranged in different groups depending upon their relationship. Now these groups are known as components. Finally, component diagrams are used to visualize the implementation.

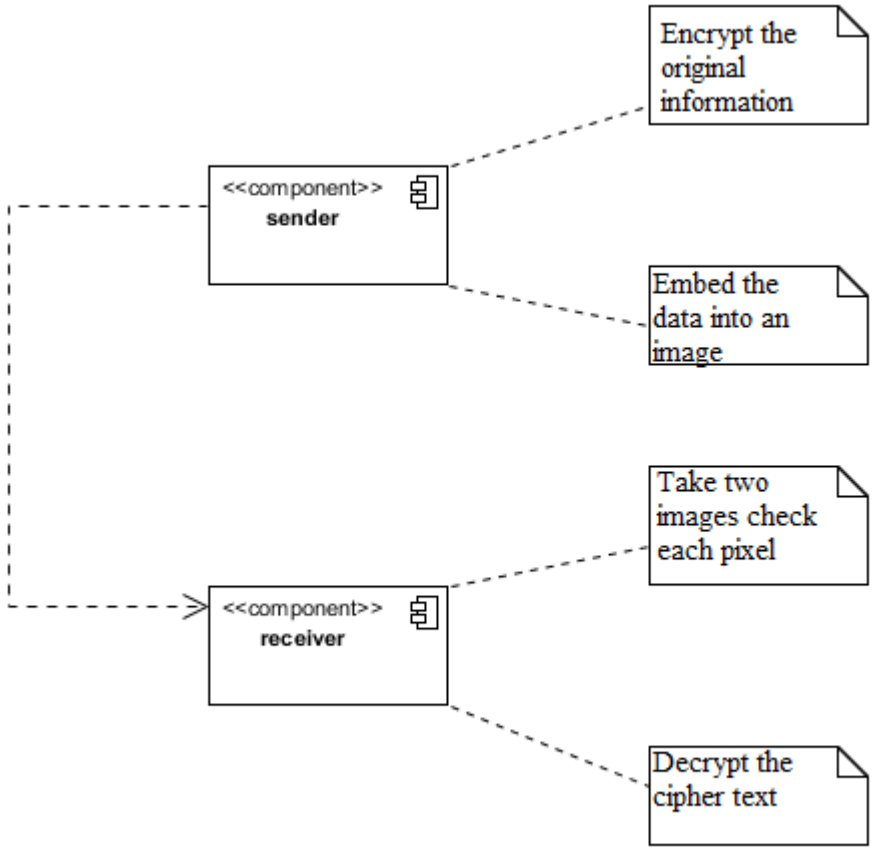

Figure 3.6.1 Component diagram

3.7 About Deployment Diagram:

Deployment diagrams are a set of nodes and their relationships. These nodes are physical entities where the components are deployed.

Deployment diagrams are used for visualizing deployment view of a system. This is generally used by the deployment team.

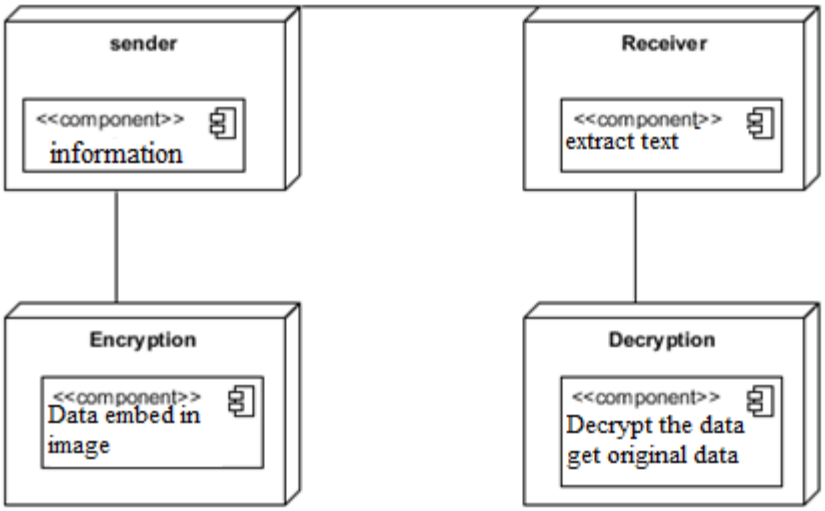

Figure 3.7.1 Deployment diagram

4.DESIGN

4.1 Design and description of Algorithm:

Steganography is the art and science of hiding communication; a steganographic system thus embeds hidden content in unremarkable cover media so as not to arouse an eavesdropper's suspicion. In the past, people used hidden tattoos or invisible ink to convey steganographic content. Today, computer and network technologies provide easy-to-use communication channels for steganography. Essentially, the information-hiding process in a steganographic system starts by identifying a cover medium's redundant bits (those that can be modified without destroying that medium's integrity). The embedding process creates a stego medium by replacing these redundant bits with data from the hidden message. Modern steganography's goal is to keep its mere presence undetectable, but steganographic system because of their invasive nature leave behind detectable traces in the cover medium. Even if secret content is not revealed, the existence of it is: modifying the cover medium changes its statistical properties, so eavesdroppers can detect the distortions in the resulting stego medium's statistical properties. The process of finding these distortions is called statistical steganalysis.

The proposed model has the following process

First take the secret information and encrypt the secret information. Take a cover image and embed the cipher text into cover image. Save the stego image and send it to the Receiver.

Receiver take the stego image as well as original cover image and check all the pixels and get all modified pixels. And extract cipher text from stego image. Decrypt the cipher text and get original secret information.

Algorithm:

Encryption:

Step 1: Select secret information and image.

Step 2: Divide the information into three parts.

Step 3: Enter six digit key divide into three parts

Step 4: Do the operation for each part of key mod6+1

Step 5: Add first part result of key to first part of information and subtract second part result of key to second part of information and third part result of key to third part of information.

Step 6: Now we generate random pixels and sort the pixels in ascending order

Step 7: Now we take all pixels RGB values and any one value of RGB values greater than 150 then that pixel change as brighter pixel by adding RGB values to cipher text.

Step 8: And all of RGB values less than 150 then that pixel is change as darker pixel by subtracting RGB values from cipher text .

Step 9: Embed the all values into cover image.

Decryption:

Step 1: Take to two images stego image as well as original cover image.

Step 2: Check all pixels and get all modified pixels

Step 3: Now we get difference of original image RGB values to the stego image RGB values.

Step 4: Now we get cipher text from stego image.

Step 5: Now we enter six digit key to decrypt the cipher text

Step 6: Now we get original information.

4.2 Example:

Encryption:

step 1:enter secret message

GVP

and ASCII values are

G=71

V=86

P=80

step 2:enter six digit key

123456

divide into three parts and do mod6+1 to each part

12%6+1=1

34%6+1=5

56%6+1=3

step 3:Add 1 to first part, subtract 5 to second part and add 3 to third part

we get

72

81

83

cipher text is HQJ

step 4: Generate random pixels

193,161

254,231

68,251

step 5: sort the pixels

68,251

193,161

254,231

step 6: Now we take RGB values for all pixels

68,251 193,161 254,231

R=50	R=60	R=40
G=70	G=150	G=30
B=155	B=170	B=50

step 7:Now we do

R=50+72=122%256=122	R=60+81=141%256=141
G=70+72=142%256=142	G=150+81=231%256=231
B=155+72=227%256=227	B=170+81=251%256=251

R=40-83=-42	G=30-83=-53	B=50-83=-33
255-43=212	255-53=202	255-33=222

step 8:Now we save the stego image and send it to Receiver.

Figure 4.1 Original image

Figure 4.2 Data embedded image

Decryption:

step 1: Browse the stego image and original image.

step 2: Check all pixels and get all modified pixels.

step 3: Now we calculate the difference of RGB values of original values to the RGB values of stego image.

step 4:The difference in pixels is calculated as

72

81

83

step 5: Corresponding cipher text is HQJ

step 6: Now decrypt the cipher text with the same key 123456

step 7:Now we get original message GVP

If we use LSB algorithm there is no security to the embedded data. LSB algorithm fails many times in giving security to the data. And it stores less data as compared to proposed model. In our Proposed model the two images look like same. This proposed model is more secure and robust. No changes in Quality.

5. CODING

The goal of the coding or programming phase is to translate the design of the system produced during the design phase into code in a given programming language, which can be executed by a computer and that performs the computation specified by the design.

The coding phase affects both testing and maintenance. The goal of coding is not to reduce the implementation cost but the goal should be to reduce the cost of later phases. In other words the goal is not to simplify the job of programmer. Rather the goal should be to simplify the job of the tester and maintainer.

5.1. Coding Approach:

There are two major approaches for coding any software system. They are Top-Down approach and bottom up approach.

Bottom-up Approach can best suit for developing the object-oriented systems. During system design phase of reduce the complexity. We decompose the system into an appropriate number of subsystems, for which objects can be modelled independently. These objects exhibit the way the subsystems perform their operations .Once objects have been modelled they are implemented by means of coding. Even though related to the same system as the objects are implemented of each other the Bottom-Up approach is more suitable for coding these objects.

In this approach, we first do the coding of objects independently and then we integrate these modules into one system to which they belong. In this project, Top-Down approach is followed.

5.2 Information Handling:

Any software system require some amount of information during its operation selection of appropriate data structures can help us to produce the code so that objects of the system can better operate with the available information decreased complexity.

5.3 Programming Style:

Programming style deals with act of rules that a programmer has to follow so that the characteristics of coding such as Traceability, Understand ability, Modifiability, and Extensibility can be satisfied.

5.4 Verification and Validation

Verification is the process of checking the product built is right. Validation is the process of checking whether the right product is built. During the Development of the system Coding for the object has been thoroughly verified from different aspects regarding their design, in the way they are integrated and etc. The various techniques that have been followed for validation discussed in testing the current system.

Form level validation:

Validations of all the inputs given to the system at various points in the forms are validated while navigating to the next form. System raises appropriate custom and predefined exceptions to alert the user about the errors occurred or likely to occur.

Field level validation:

Validations at the level of individual controls are also applied whenever necessary. System pops up appropriate and sensuous dialogs whenever necessary.

5.5. SAMPLE CODE

Code for Encryption and Embedding

```
package encryptionandstegnography;

import static encryptionandstegnography.TextEncryption.image;

import java.awt.Color;

import java.awt.image.BufferedImage;

import java.io.BufferedWriter;

import java.io.File;

import java.util.Random;

import javax.imageio.ImageIO;

import javax.swing.ImageIcon;

import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

public class TextEncryption2 extends javax.swing.JFrame {

String pixelPos[] = new String[3000];

   String orderedPixelPos[] = new String[3000];

   public TextEncryption2() {

      initComponents();

   }

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

      // TODO add your handling code here:

      System.out.println("height:"+TextEncryption.height+"
width:"+TextEncryption.width);
```

```
    String pixelPosString = "";
    for(int i=0;i<TextEncryption.encryptedText.length();i++)
    {
        pixelPos[i] = generateRandomDistinctPixel(pixelPos,i);
        if(i%5 == 0)
        {
            pixelPosString+="\n"+pixelPos[i];
        }
        else
            pixelPosString+="\t"+pixelPos[i];
    }
    System.out.println("Random pixels are:");
    for(int i=0;i<=TextEncryption.encryptedText.length()-1;i++)
    {
        System.out.println(i+" -> "+pixelPos[i]);
    }
    jTextArea1.setText(pixelPosString);
}
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int distinctRows[]=new int[100];
    int distinctCols[]=new int[100];
    int rowCount =0,colCount =0;
    for(int i=0;i<TextEncryption.encryptedText.length();i++)
```

```
{
   String pix[] = pixelPos[i].split(",");
   if(rowCount == 0)
   {
      distinctRows[rowCount] = Integer.parseInt(pix[0].trim());
      rowCount++;
   }
   else
   {
      boolean found = false;
      for(int j=0;(j<rowCount);j++)
      {
         if(distinctRows[j]==Integer.parseInt(pix[0].trim()))
         {
            found = true;
            break;
         }
      }
      if(!found)
      {
         distinctRows[rowCount++] = Integer.parseInt(pix[0].trim());
      }
   }
   if(colCount == 0)
```

```
    {
        distinctCols[colCount] =  Integer.parseInt(pix[1].trim());
        colCount++;
    }
    else
    {
        boolean found = false;
        for(int j=0;(j<colCount);j++)
        {
            if(distinctCols[j]==Integer.parseInt(pix[1].trim()))
            {
                found = true;
                break;
            }
        }
        if(!found)
        {
            distinctCols[colCount++] = Integer.parseInt(pix[1].trim());
        }
    }
}
System.out.println("distinct Rows:");
for(int i=0;i<=rowCount-1;i++)
{
```

```
    System.out.print("\t "+distinctRows[i]);
}
System.out.println("distinct Cols:");
for(int i=0;i<=colCount-1;i++)
{
    System.out.print("\t "+distinctCols[i]);
}
int count =0;
String orderedPixelsString = "";
//need to sort those distict Rows and distinct columns
int minpos=0;
for(int i=0;i<=rowCount-1;i++)
{
    int min = distinctRows[i];
    minpos = i;
    for(int j=i+1;j<=rowCount-1;j++)
    {
        if(min > distinctRows[j])
        {
            min = distinctRows[j];
            minpos = j;
        }
    }
    distinctRows[minpos] = distinctRows[i];
```

```
    distinctRows[i] = min;
    System.out.println("r: "+min);
}
for(int i=0;i<=colCount-1;i++)
{
    int min = distinctCols[i];
    minpos = i;
    for(int j=i+1;j<=colCount-1;j++)
    {
        if(min > distinctCols[j])
        {
            min = distinctCols[j];
            minpos = j;
        }
    }
    distinctCols[minpos] = distinctCols[i];
    distinctCols[i] = min;
    System.out.println("c: "+min);
}
for(int i=0;i<=rowCount-1;i++)
{
    for(int j=0;j<=colCount-1;j++)
    {
        String pixelStr = distinctRows[i]+","+distinctCols[j];
```

```
            boolean found = false;
            for(int k=0;k<=TextEncryption.encryptedText.length()-1;k++)
            {
               if(pixelPos[k].trim().equals(pixelStr))
               {
                  found = true;
                  break;
               }
            }
            if(found)
            {
               orderedPixelPos[count++] = pixelStr;
               if(count%5 == 0)
                     orderedPixelsString+= "\n"+pixelStr;
               else
                     orderedPixelsString+= "\t"+pixelStr;
            }
         }
      }
      jTextArea2.setText(orderedPixelsString);
   }
   private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
      //BufferedImage modifiedImage = new
BufferedImage(m,m,BufferedImage.TYPE_INT_RGB);
```

```
for(int i=0;i<TextEncryption.encryptedText.length();i++)
{
    String pixelString = orderedPixelPos[i];
    int charVal = (int)TextEncryption.encryptedText.charAt(i);
    String pixelPos[] = pixelString.split(",");
    int pixelWidthPos = Integer.parseInt(pixelPos[0]);
    int pixelHeightPos = Integer.parseInt(pixelPos[1]);
    Color color = new Color(image.getRGB(pixelWidthPos,
pixelHeightPos));
    int red = color.getRed();
    int green = color.getGreen();
    int blue = color.getBlue();
    if((red>150) || (green>150) || (blue>150))
    {
       red = (red+charVal)%256;
       green = (green+charVal)%256;
       blue = (blue+charVal)%256;
    }
    else
    {
       red = (red-charVal);
       if(red<=0)
          red = 255 + red;
       green = (green-charVal);
       if(green<=0)
```

```
                green = 255 + green;
            blue = (blue-charVal);
            if(blue<=0)
                blue = 255 + blue;
        }
        int val = (65536 * red + 256 * green + blue);
        image.setRGB(pixelWidthPos, pixelHeightPos, val);
    }
    jLabel4.setIcon(new ImageIcon(image));
}
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
     try
    {
        JFileChooser fc=new JFileChooser();
        int returnVal=fc.showSaveDialog(this);
        if(returnVal==JFileChooser.APPROVE_OPTION)
        {
          java.io.File saved_file=fc.getSelectedFile();
          String file_name=saved_file.toString();
         //  JOptionPane.showMessageDialog(null, file_name);
           BufferedWriter bw;
           File file;
            file = new File(saved_file.getAbsolutePath());
```

```
            ImageIO.write(image,"png",file);
            JOptionPane.showMessageDialog(null, "Image saved
successfully");
             // bw.write("toadd");
             //  FileWriter fw=new FileWriter(saved_file.getAbsoluteFile());
            //bw=new BufferedWriter(fw);
           }
       }
       catch(Exception e){
          JOptionPane.showMessageDialog(null,"Exception raised in saving
Image");
       }
   }
   private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
       // TODO add your handling code here:
       this.dispose();
       new TextHomeScreen().call();
   }
   public String generateRandomDistinctPixel(String pixelPos[], int
lengthOfAry)
   {
       Random randomGenerator = new Random();
       boolean isDistinct = false;
       String pixelPosition = "";
       do
```

```
    {
        int randRow = randomGenerator.nextInt(TextEncryption.height);
        int randCol = randomGenerator.nextInt(TextEncryption.width);
        boolean found = false;
        for(int i=0;i<=(lengthOfAry-1);i++)
        {
            if(pixelPos.length != 0)
            {
            if(pixelPos[i].trim().equals((randRow+","+randCol).trim()))
            {
                found = true;
            }
            }
        }
        if(found) continue;
        else
        {
            pixelPosition = (randRow+","+randCol).trim();
            break;
        }
    }while(true);
    return pixelPosition;
}
```

code for Extract data from cover image

```
package encryptionandstegnography;
import static encryptionandstegnography.TextDecryption.hidingImg;
import static encryptionandstegnography.TextDecryption.textHidingImg;
import static encryptionandstegnography.TextDecryption.m;
import static encryptionandstegnography.TextEncryption.encryptedText;
import static encryptionandstegnography.TextEncryption.key;
import java.awt.Color;
import javax.swing.JOptionPane;
public class TextDecryption2 extends javax.swing.JFrame {
      String encryptedString="";
   int key=0;
   public TextDecryption2() {
      initComponents();
   }
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
      int modPixAry[][] = new int[1000][2];
         int count = 0;
            for (int i = 0; i < m; i++)
            {
               for (int j = 0; j < m; j++)
               {
                  Color color = new Color(textHidingImg.getRGB(i, j));
```

```
                int red1=color.getRed();
                int green1=color.getGreen();
                int blue1=color.getBlue();
                Color color2 = new Color(hidingImg.getRGB(i, j));
                int red2=color2.getRed();
                int green2=color2.getGreen();
                int blue2=color2.getBlue();
                System.out.println("red1:"+red1+"
red2:"+red2+"green1:"+green1+"  green2:"+green2+"blue1:"+blue1+"
blue2:"+blue2);
                if((red1!=red2) || (green1!=green2) || (blue1!=blue2))
                {
                  System.out.println("red1:"+red1+"
red2:"+red2+"green1:"+green1+"  green2:"+green2+"blue1:"+blue1+"
blue2:"+blue2);
                  modPixAry[count][0]=i;
                  modPixAry[count][1]=j;
                  count++;
                }
              }
            }
            String modPicStr = "";
            encryptedString = "";
            for(int i=0;i<=count-1;i++)
            {
               modPicStr += modPixAry[i][0]+","+modPixAry[i][1]+"  ";
```

```
if(i%3 == 0)
   modPicStr += "\n";
Color color = new Color(hidingImg.getRGB(modPixAry[i][0],
modPixAry[i][1]));
int red1 = color.getRed();
int green1 = color.getGreen();
int blue1 = color.getBlue();

Color color2 = new
Color(textHidingImg.getRGB(modPixAry[i][0], modPixAry[i][1]));
int red2 = color2.getRed();
int green2 = color2.getGreen();
int blue2 = color2.getBlue();
int difference=0;
 if((red1>150) || (green1>150) || (blue1>150))
{
   if(red2<red1)
   {
      difference = (256 - red1)+red2;
   }
   else
   {
      difference = red2 - red1;
   }
```

6. TESTING

Testing is the process of finding differences between the expected behaviour specified by system models and the observed behaviour of the system. Testing is a critical role in quality assurance and ensuring the reliability of development and these errors will be reflected in the code so the application should be thoroughly tested and validated.

Unit testing finds the differences between the object design model and its corresponding components. Structural testing finds differences between the system design model and a subset of integrated subsystems. Functional testing finds differences between the use case model and the system.

Finally performance testing, finds differences between non-functional requirements and actual system performance. From modelling point of view, testing is the attempt of falsification of the system with respect to the system models. The goal of testing is to design tests that exercise defects in the system and to reveal problems.

6.1. Testing Activities:

Testing a large system is a complex activity and like any complex activity. It has to be broke into smaller activities. Thus incremental testing was performed on the project i.e., components and subsystems of the system were tested separately before integrating them to form the subsystem for system testing.

6.1.1 Unit Testing:

Unit testing focuses on the building blocks of the software system that is the objects and subsystems. There are three motivations behind focusing on components. First unit testing reduces the complexity of overall test activities allowing focus on smaller units of the system, second unit testing makes it easier to pinpoint and correct faults given that

few components are involved in the test. Third unit testing allows parallelism in the testing activities, that is each component are involved in the test. Third unit testing allows parallelism in the testing activities that is each component can be tested independently of one another.

In this system each module of segment display, encryption and decryption are treated as individual units and are tested individually.

The following are some unit testing techniques.

- **Equivalence testing**: It is a black box testing technique that minimizes the number of test cases. The possible inputs are partitioned into equivalence classes and a test case is selected for each class.
- **Boundary testing:** It is a special case of equivalence testing and focuses on the conditions at the boundary of the equivalence classes. Boundary testing requires that the elements be selected from the edges of the equivalence classes.
- **Path testing:** It is a white box testing technique that identifies faults in the implementation of the component the assumption here is that exercising all possible paths through the code at least once. Most faults will trigger failure. This acquires knowledge of source code.

In this system, there are two screens one represents sender form and one for receiver. Sender can follow plain text and browse image. Sender Encrypt the plain text and generate random pixels from cover image. And do some mathematical operations on pixel RGB values and embed the cipher text into cover image. Sender can directly go to send image step without doing previous steps. In same manner, if once receiver gets Stego-image he can directly go to receive image step.

6.1.2 Integration Testing:

Integration testing defects faults that have not been detected. During unit testing by focusing on small groups on components two or more components are integrated and tested and once tests do not reveal any new faults, additional components are added to the group. This procedure allows testing of increasing more complex parts on the system while keeping the location of potential faults relatively small. I have used the following approach to implements and integrated testing.

Top-down testing strategy unit tests the components of the top layer and then integrated the components of the next layer down. When all components of the new layer have been tested together, the next layer is selected. This was repeated until all layers are combined and involved in the test.

In this system**,** we first perform individual testing on the modules, Encryption and Decryption which are individually verified are integrated for making a perfect project.

6.1.3 Validation Testing

The systems completely assembled as package, the interfacing have been uncovered and corrected, and a final series of software tests are validation testing. The validation testing is nothing but validation success when system functions in a manner that can be reasonably expected by the customer. The system validation had done by series of Black-box test methods.

In this system, validations are performed on each individual control. In encryption and decryption, if key is not entered then the tool will handle all those situations efficiently. Also if any of the input is not given or any wrong click occurs then system will generate appropriate error messages.

6.1.4 System Testing:

System testing of software is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. The following are some system testing activities.

- **Usability testing:**

It is a technique used to evaluate a product by testing it on users. It finds differences between the functional requirements and the system. Improved pixel sieve method can be easily used by any user with a minimal knowledge of visual cryptography and windows operating system.

- **Performance testing:**

It covers a broad range of engineering or functional evaluations to meet measurable performance characteristics. The coding of the system involves a simple but effective method that can perform well even the user submits image with high resolution and maximum size.

- **Installation testing:**

It is a kind of quality assurance work in the software industry that focuses on what customers will need to do to install and set up the new software successfully. The system is installed in the target environment.

Finally, we test whether the application is able to encrypt and successfully decrypt the data. This involves testing all the individual systems of the application. This is achieved through System testing.

6.2 Testing Types

- **Unit testing:**

It finds the differences between the object design model and its corresponding components. In this test each component is tested independent of the other thus allowing parallelism in testing activity.

Ex: Individual units like getting keys for encryption, loading the images for generating the Steg0-image are tested individually.

- **Structural testing:**

It finds differences between the system design model and a subset of integrated subsystems.

- **Functional testing:**

It finds differences between the use case model and the system.

- **Performance testing:**

It finds differences between non-functional requirements and actual system performance.

6.3 Testing Plan

Testing accounts for 45 - 75% of the typical project effort. It is also one of the most commonly underestimated activities on a project. A test plan is a document that answers the basic questions about your testing effort. It needs to be initiated during the requirements gathering phase of your project and should evolve into a roadmap for the testing phase.

- Test Planning enables a more reliable estimate of the testing effort up front.
- It allows the project team time to consider ways to reduce the testing effort without being under time pressure.

- Test Plan helps to identify problem areas and focuses the testing team's attention on the critical paths.
- Test plan reduces the probability of implementing non-tested components.

6.4 Test Case Report

Test Case-1: Initiating _Test

Test Case id: 01

Test Case Name: Welcome Screen Test

Test Case Type: Black Box Testing

Description	**Expected Value**	**Observed Value**	**Result**
Sender selects "Proceed" button.	Screen Home should be displayed.	Home screen should be opened.	Successful

Table 6.3.1: Test Case-1: Initiating-Test

Test Case-2 : Encryption and Embedding

Test Case id : 02

Test Case Name : Encrypt and Embed

Test Case Type : White Box Testing

Description	Expected Value	Observed Value	Result
User clicks on 'Encrypt' button and converts message to cipher text	System should convert message to cipher text	Message is converted to cipher text	Successful
User clicks on 'BROWSE IMAGE' button &selects image	System should display image	Image is displayed	Successful
User clicks on 'BROWSE IMAGE' button and does not select any image	Error message should be displayed saying " that select image"	Message box is displayed saying " please select image"	Successful

Table 6.3.2: Test Case-2: Encryption And Embedding

Test Case-3 : Enter key

Test Case id : 03

Test Case Name : Enter key

Test Case Type : White Box Testing

Description	Expected Value	Observed Value	Result
Sender enters 6digit key and click “Encrypt” button.	The message should be Encrypted.	The message is encrypted.	Successful
Sender enter 6 characters and click on "Encrypt" button	"enter only digits" message should be displayed	"Enter only digits" message is displayed	Successful

Table 6.3.3: Test Case-3: Enter key

Test Case-4 : Extraction and Decryption

Test Case id : 04

Test Case Name : Retrieving Secret Message

Test Case Type : White Box Testing

Description	Expected Value	Observed Value	Result
User clicks on 'DATA HIDDEN IMAGE' button and selects received image	System should display image	Image is displayed	Successful
User clicks on 'BROWSE ORIGINAL IMAGE' button .	System should display image	Image is displayed	Successful
User clicks on 'BROWSE IMAGE' button and does not select any Stego-image	Error message should be displayed saying " that select Stego-image"	Message box is displayed saying " please select image"	Successful
User extracts cipher text from Image	Extracted cipher text should be displayed	Cipher text is displayed	Successful
User Decrypt the Message	Cipher text should be decrypted	Cipher text is decrypted	Successful

Table 6.3.4: Test Case-4: Extraction and Decryption

Test Case-5 : Back Tracking

Test Case id : 05

Test Case Name : Backtrack Test

Test Case Type : Black Box Testing

Description	Expected Value	Observed Value	Result
User clicks on 'HOME' button it goes TO HOME PAGE window	'HOME PAGE' window should be displayed	HOME PAGE window is displayed	Successful

Table 6.3.5: Test Case-5: Back Tracking

Test Case-6 : **Extraction and Decryption**

Test Case id : 06

Test Case Name : Decrypted Stego-image Test

Test Case Type : Black Box Testing

Description	Expected Value	Observed Value	Result
User clicks on 'Extract' button after browsing Stego-image and original image,	System should display the cipher text	The cipher text is displayed	Successful
User clicks on 'Extract' button without browsing Stego image	Error message should be displayed saying " please browse Data hidden image"	Message box is displayed saying " please browse encrypted image "	Successful

Table 6.3.6: Test Case-6: Extraction and Decryption

7.RESULT

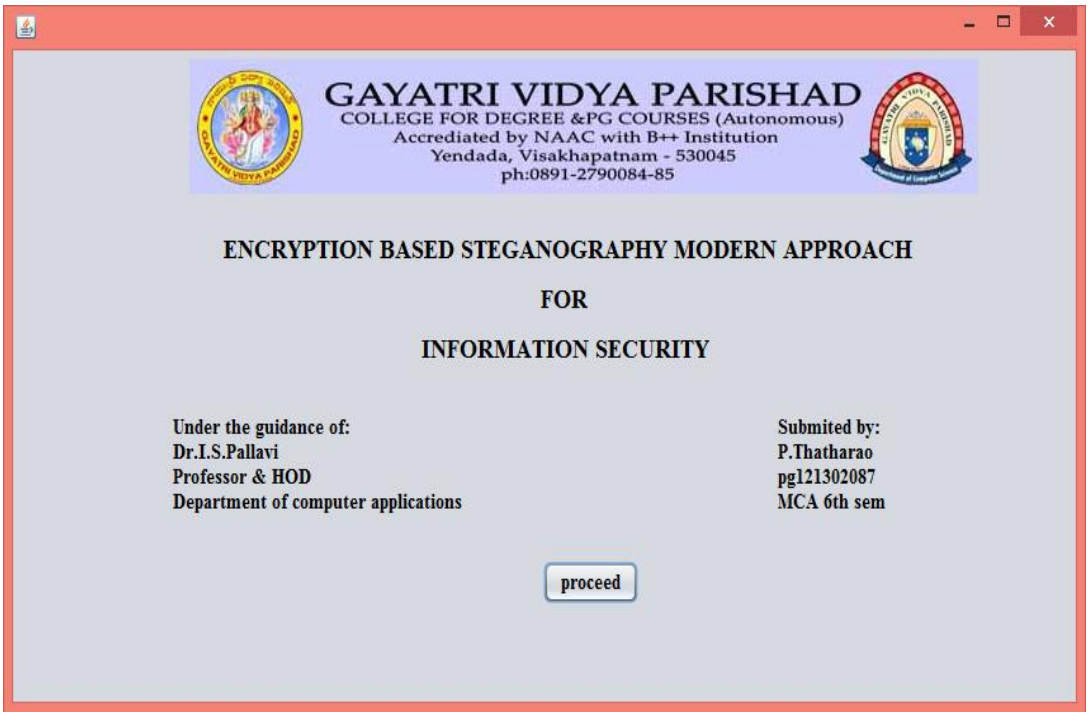

Figure 7.1 Starting screen

The project execution starting from this page. This project has title of the project. When we click on Proceed button the project execution is starts.

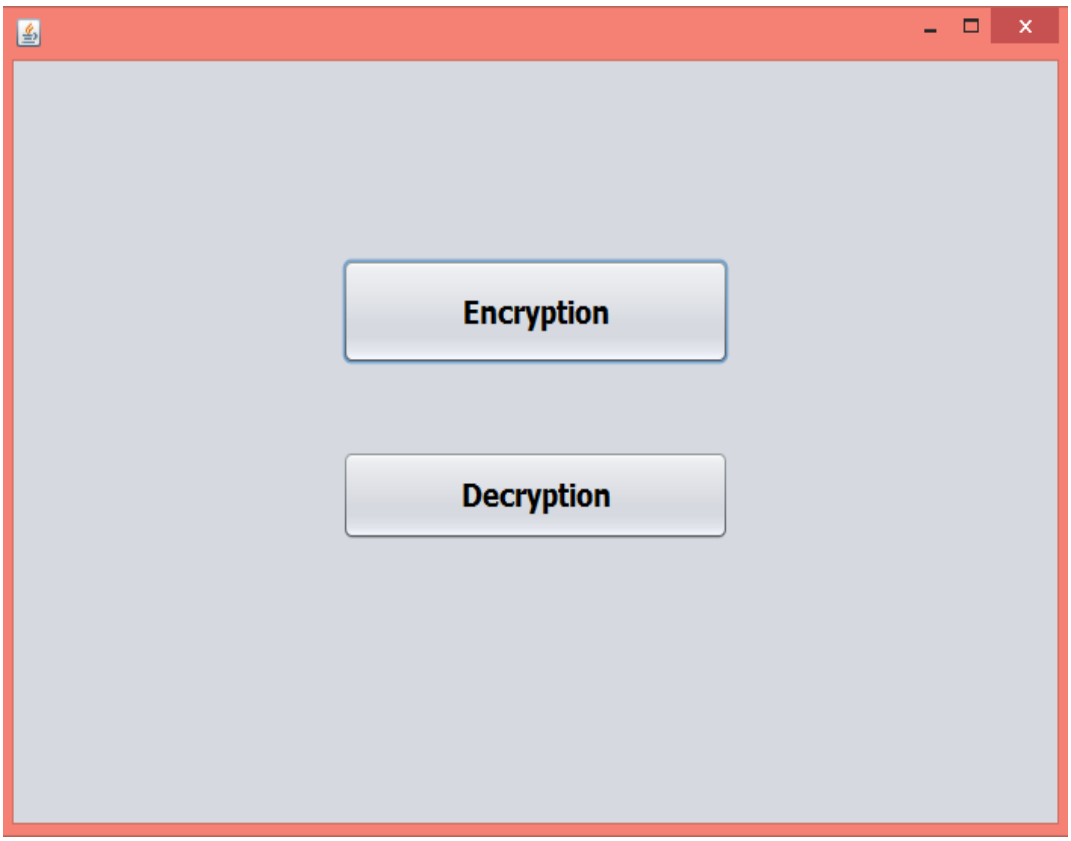

Figure 7.2 Home screen

When we click on Proceed button the above screen will be displayed. when we click on encryption, the Encryption process will be done. When we click on Decryption button, the Decryption process will be done.

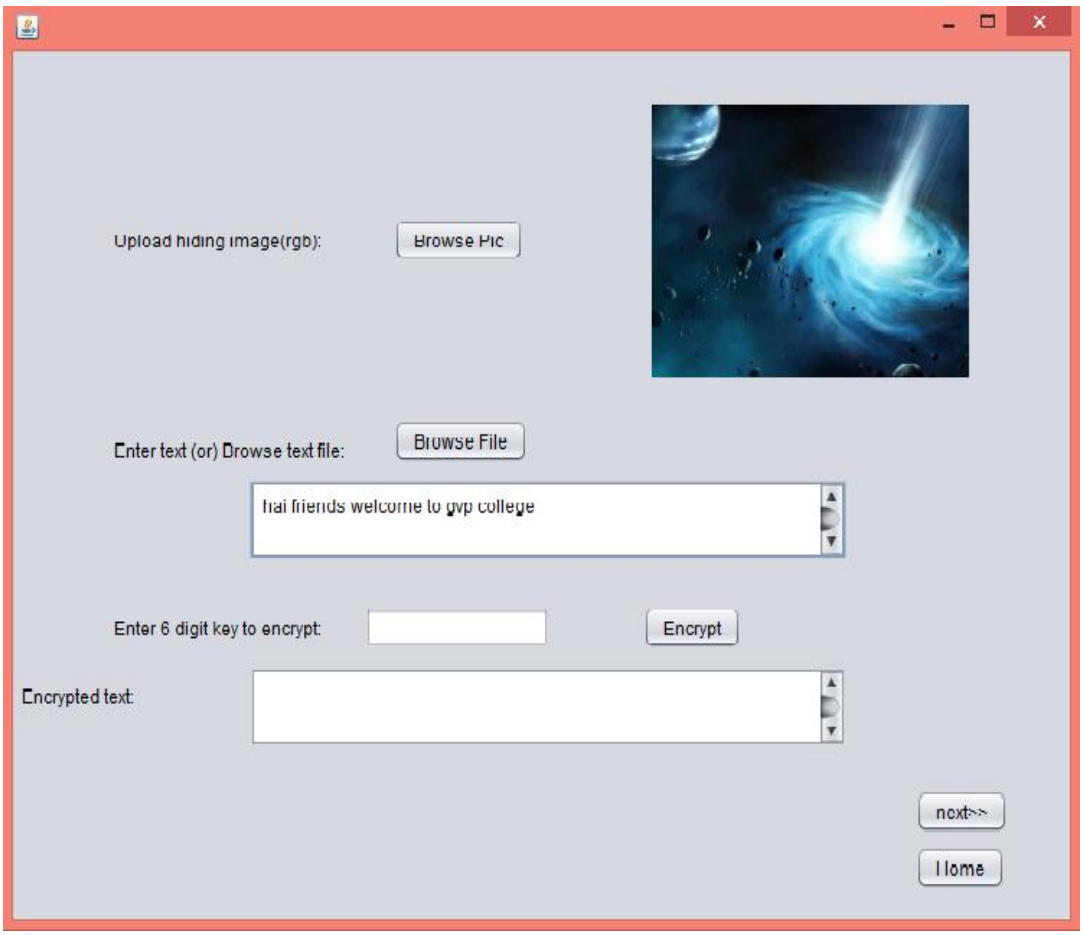

Figure 7.3Browse image and text screen

When we click on Encryption button the encryption should be done. In that process we have to browse an image and secret message which we want to send. After browsing image and message the above screen will be displayed.

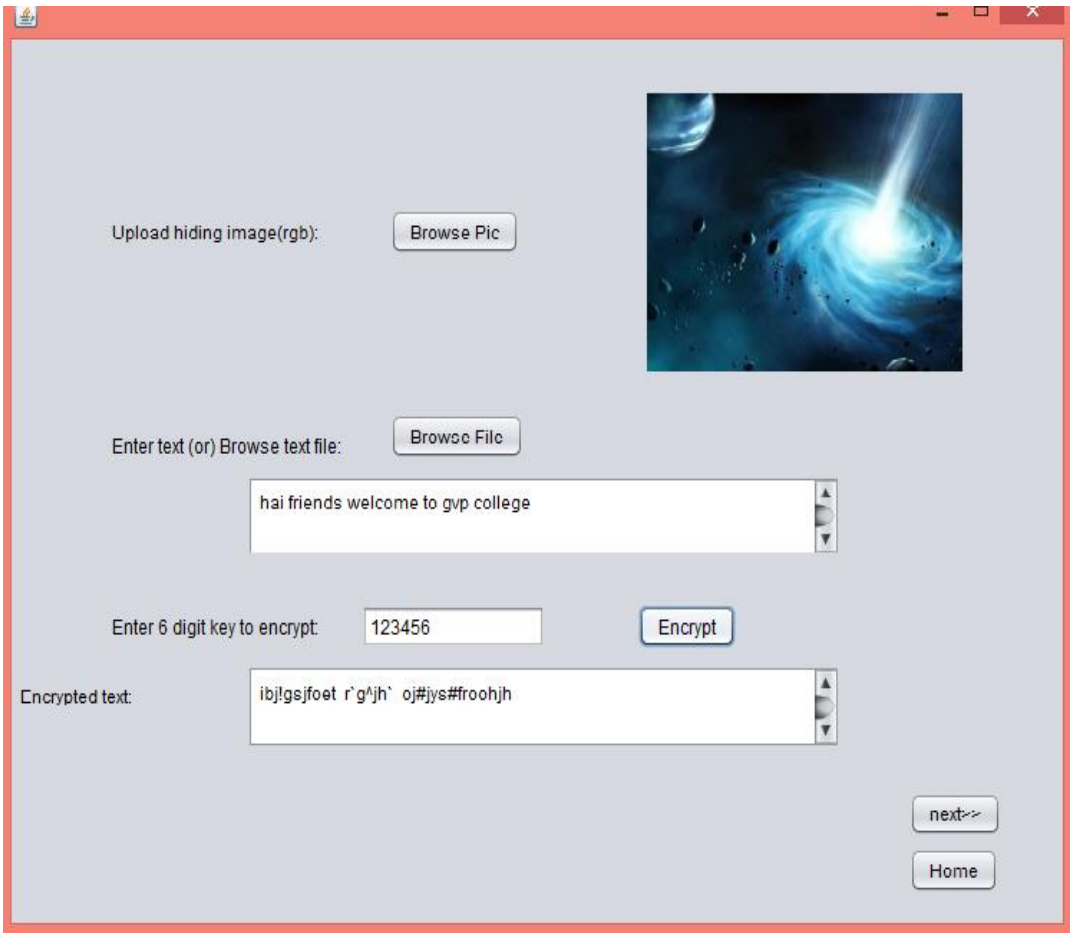

Figure 7.4 Text Encryption screen

After browse image and message and give 6digit key to encrypt the secret message. After we click on Encrypt button the cipher text will be form and above screen will be displayed. And click on Next button.

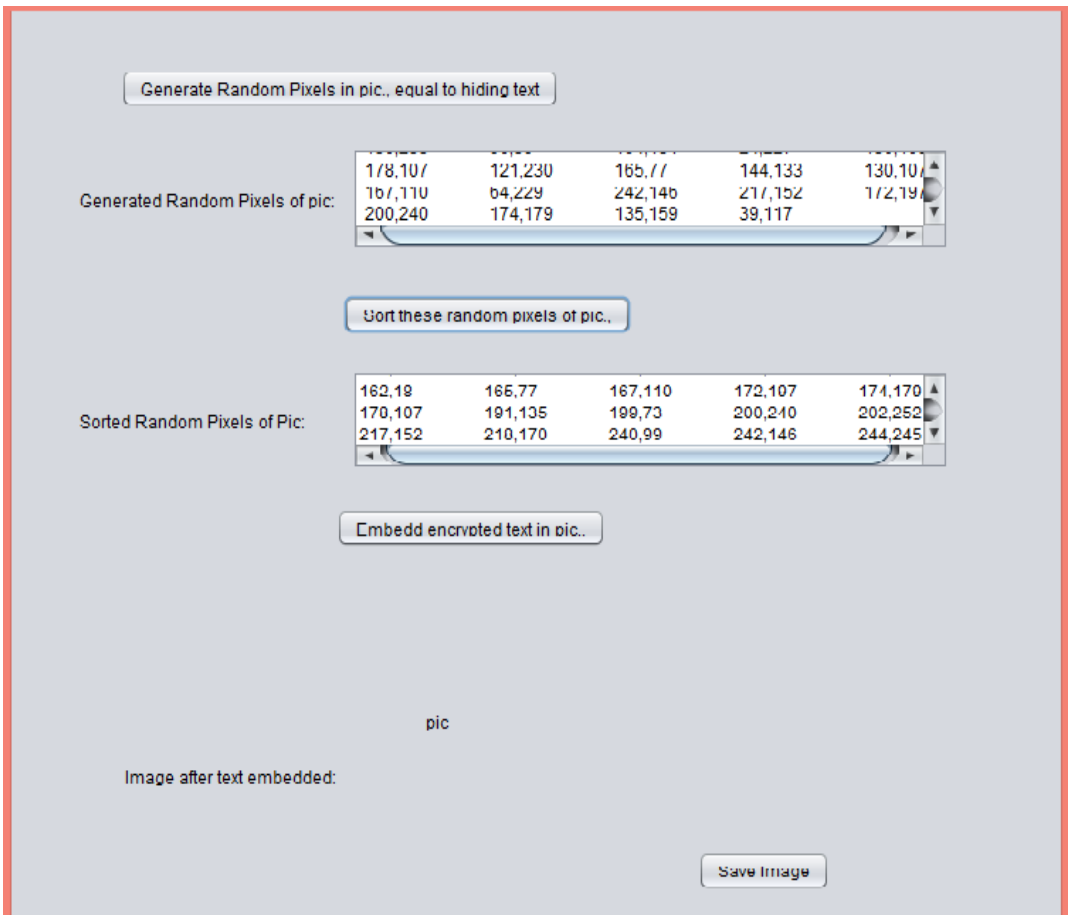

Figure 7.5 Random pixel generation and sorting pixels screen

After Encrypt the secret message we generate random pixels for cover image and sorting the image random pixels. We get the above screen.

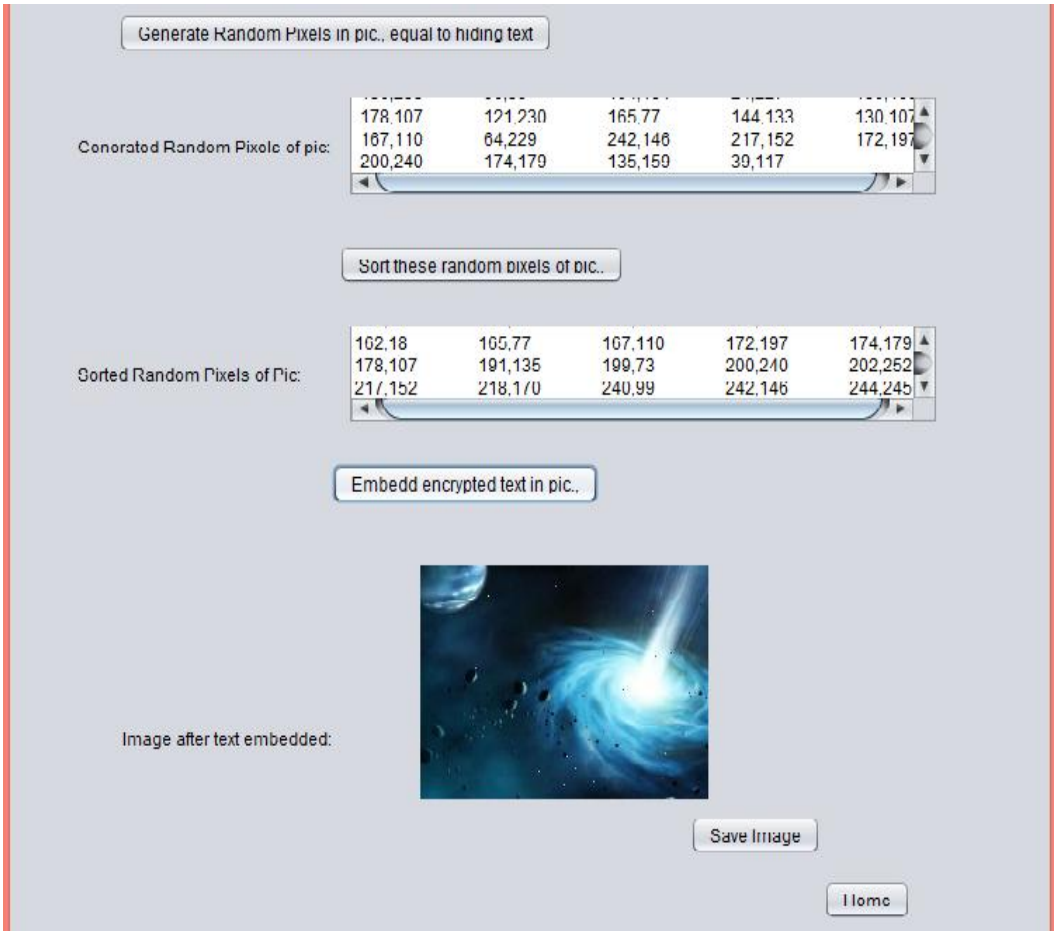

Figure 7.6 Embed image screen

After generating random pixels and we embed the cipher text into the cover image. And save the stego-image and send it to receiver. The above screen will be displayed.

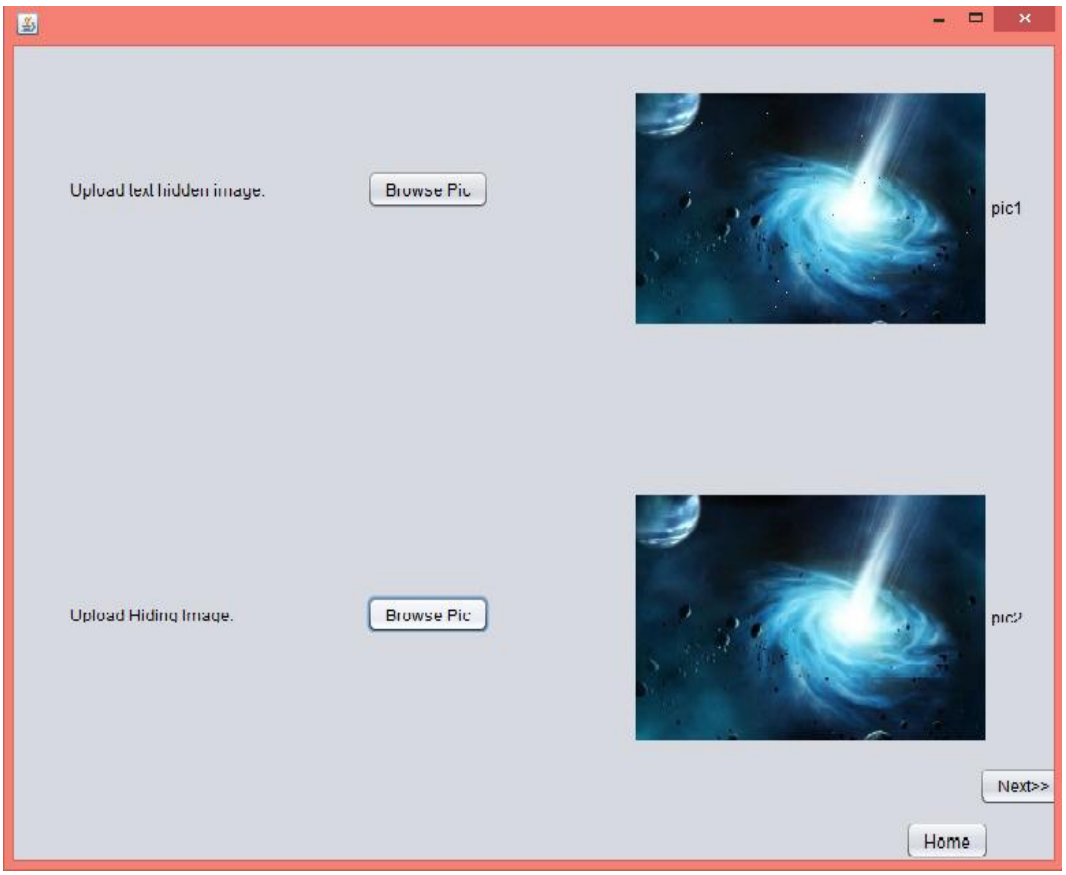

Figure 7.7 Browse Stego-image and Original image screen

Receiver can browse the stego-image and original image for check all pixels. Browse two images are displayed on above screen

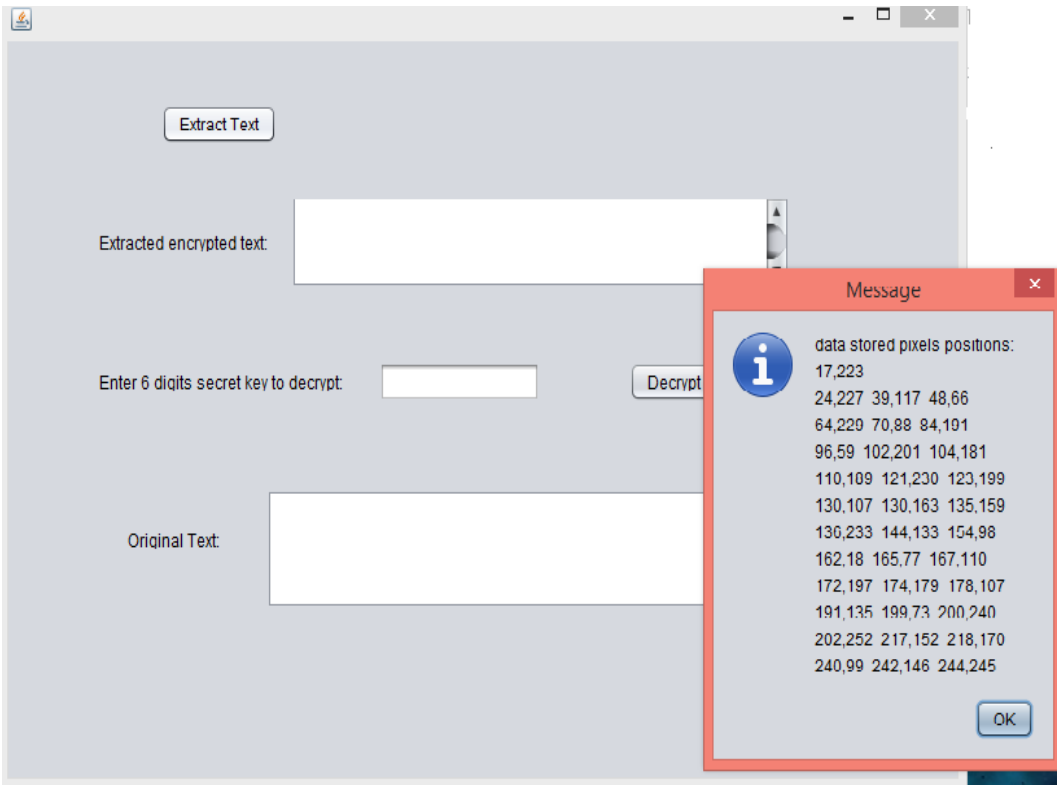

Figure 7.8 Get modified pixels screen

After browse two images compare all pixels and get all modified pixels. From that all modified pixels we extract cipher text. The modified pixels shown in above screen.

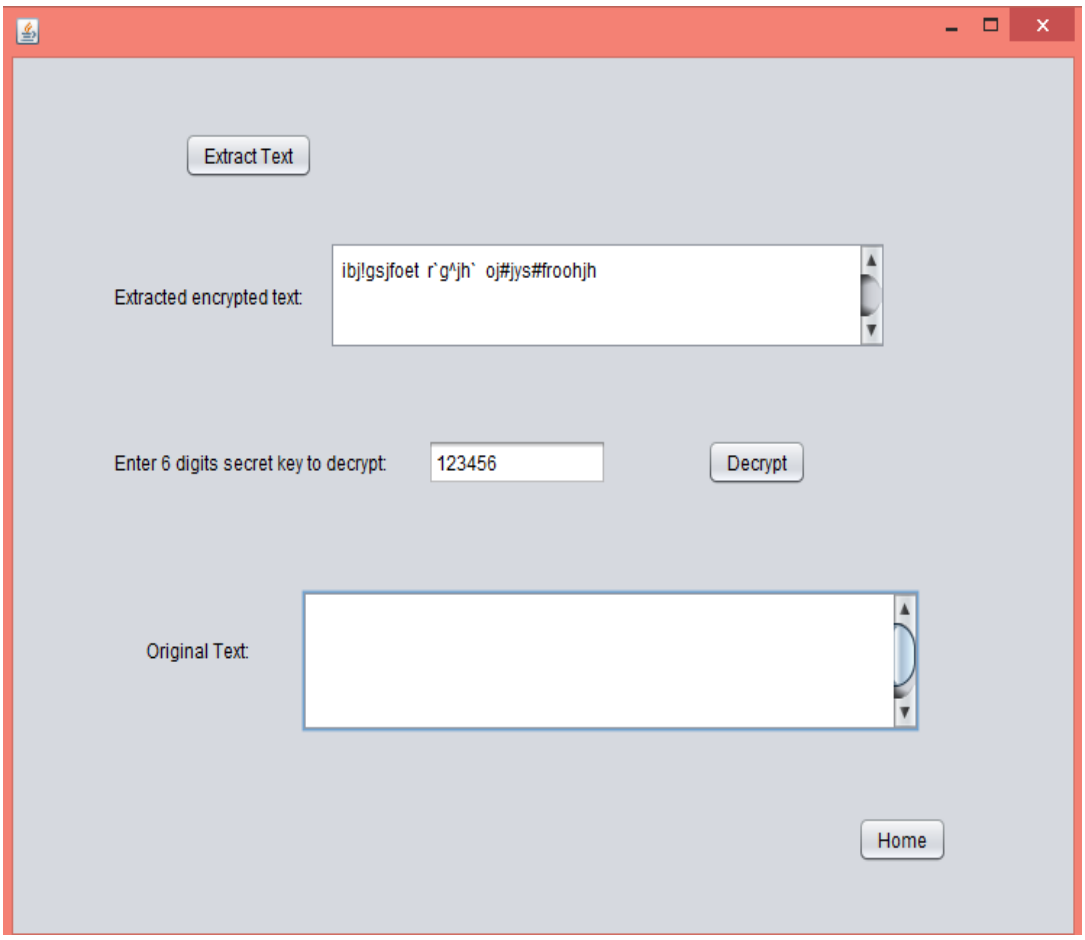

Figure 7.9 Extract cipher text from pixels screen

After getting all modified pixels we get cipher text. And we give 6 digit key to decrypt the cipher text. The 6 digit key same as used in encryption. The above screen displays cipher text and 6 digit key.

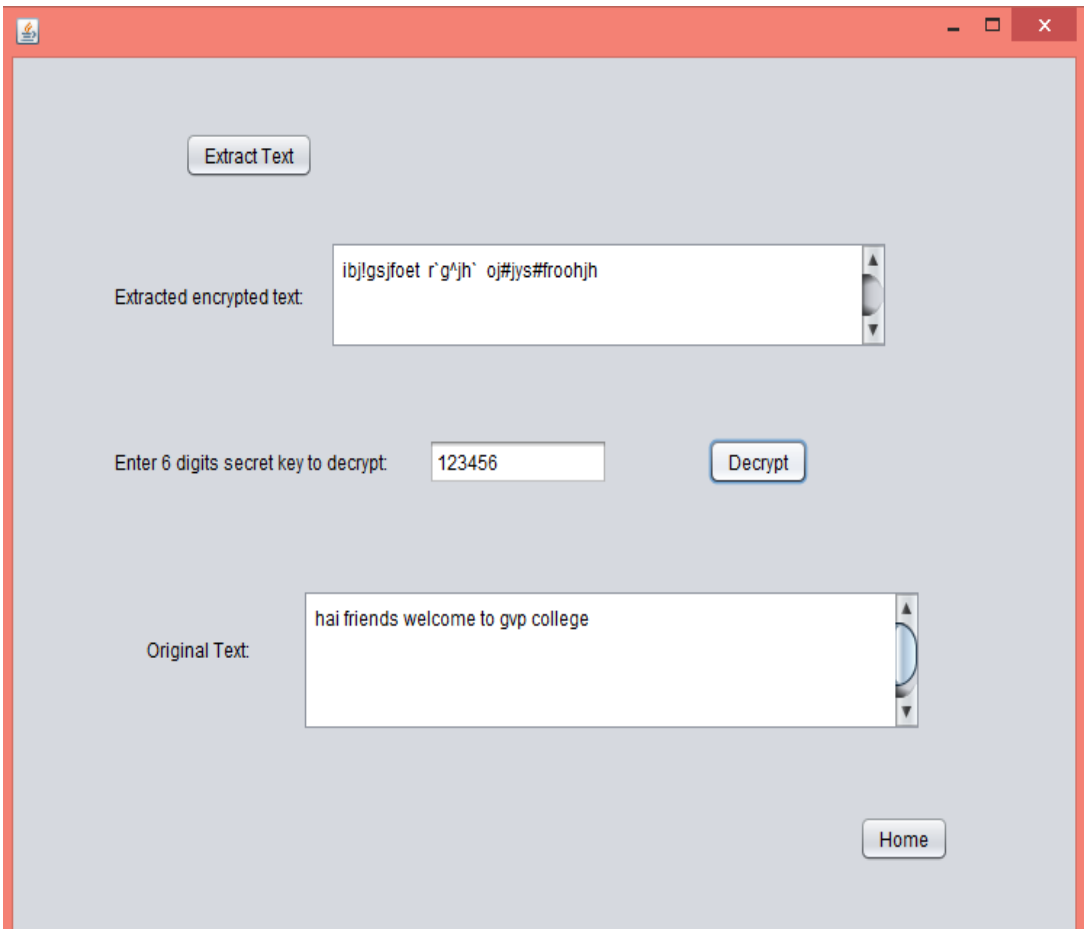

Figure 7.10 Get original message from cipher text

After getting cipher text from modified pixels, we give 6 digit key to decrypt the cipher text and we get original secret message.

8. CONCLUSION

Digital Steganography is an interesting scientific area which falls under the umbrella of security systems. Image steganography is a considerably new dimension in the field of information hiding. In this paper a secure image steganography technique is proposed to hide data bits. The experimental results show that the technique produces good quality stego images, give security to the data and embed more data in cover image as compared to LSB algorithm model.

9. REFERENCES

9.1 Research References

[1] N. Provos, P. Honeyman " An introduction to steganography," IEEE Security & Privacy Magazine, Vol. 1, Issue 3, pp. 32-44, 2003.

[2] H. A. Jalab, A. A. Zaidan, B. B. Zaidan, "New Design for Information Hiding with in Steganography Using Distortion Techniques," International Journal of Engineering and Technology (IJET)), ISSN: 1793-8236, Vol 2, No. 1, pp. 72-77, 2010.

[3] K. Bailey, K. Curran, " An evaluation of image based

steganography methods using visual inspection and automated detection techniques," Multimedia Tools and Applications, Vol. 30 , Issue 1, pp. 55-88, 2006.

[4] Cheddad, J. Condell, K. Curran, & P. Kevitt, (2010). Digital image Steganography- survey and analysis of current methods. Signal Processing, 90, 727–752.

[5] C. K. Chan, L. M. Cheng, "Hiding data in image by simple LSB substitution", pattern recognition, Vol. 37, No. 3, 2004, pp. 469-474.

[6] R. Z. Wang, C. F. Lin and I. C. Lin, "Image Hiding by LSB substitution and genetic algorithm", Pattern Recognition, Vol. 34,No. 3, pp. 671-683, 2001.

[7] D. Sandipan, A. Ajith, S. Sugata, An LSB Data Hiding Technique Using Prime Numbers, The Third International Symposium on

Information Assurance and Security, Manchester, UK,IEEE CS press, 2007.

[8] C. Kessler. (2001). Steganography: Hiding Data within Data. An edited version of this paper with the title "Hiding Data in Data". Windows & .NET Magazine.

[9] N. F. Johnson and S. Jajodia, "Steganalysis: The Investigation of Hidden Information", IEEE Information Technology Conference, September 1998.

9.2 Book References

1. Software Engineering Principles by Roger.S. Pressman
2. Object oriented s/w engineering By Tata MCGRAW HILL
3. Cryptography and Network Security By William Stallings.
4. Steganography in Digital Media: Principles, Algorithms, and Applications By Jessica Fridrich
5. Image Steganography and Steganolysis By Philip Bateman
6. Core Java an integrated approach of Black Book By R.Nageswara Rao
7. JAVA Complete Reference, By Herbert Schildt

10. APPENDIX

10.1 List of Tables

10.2 List of Figures