IMAGE Encryption by Zigzag, Partitioning and Swapping with Steganography using LSB algorithm

Project Report Submitted

In partial fulfillment of the requirements

for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

By

D.Hanu Chandra – 1210313215

CH.Sai Smaran – 1210313213

G.Sandeep – 1210313219

M.Jaya Ram – 1210313238

Under the guidance of

Sri Ch.Sai Pratheek

Assistant Professor

Department of Computer Science and Engineering

GITAM Institute of technology

GITAM University

Visakhapatnam – 530 045

Andhra Pradesh, India.

Certificate

This is to certify that D.Hanu Chandra(1210313215), Ch.Sai Smaran(1210313213), G.Sandeep (1210313219), M.Jaya Ram (120313238),students of GITAM Institute of Technology has worked on the project titled as "IMAGE ENCRYPTION BY ZIGZAG , PARTITIONING AND SWAPPING WITH STEGANOGRAPHY USING LSB ALGORITHM" during the academic year 2016-2017 towards fulfilment of credit for the project phase of B.Tech and submitted good report, as compiled in the following pages under mysupervision.

Project Reviewers:	**Project Guide:**
1. Sri PNRL Chandra Sekhar	Sri.Ch.Sai Pratheek
Associate Professor	Assistant Professor
2. Sri SVG Reddy	
Associate Professor	

DECLARATION

We, D.Hanu Chandra (1210313215), Ch.Sai Smaran (1210313213), G.Sandeep (1210313219) and M.Jaya Ram (1210313238) hereby declare that the project report entitled **"IMAGE ENCRYPTION BY ZIGZAG, PARTITIONING AND SWAPPPING WITH STEGANOGRAPHY USING LSB ALGORTIHM"** is an original and authentic work done in the Department of CSE, GITAM Institute of Technology, GITAM University, Rushikonda, Visakhapatnam, submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering. The matter embodied in this project work has not been submitted earlier for any degree or diploma to the best of my knowledge.

Project batch members:

D.Hanu Chandra

Ch.Sai Smaran

G.Sandeep

M.Jaya Ram

ACKNOWLEDGEMENTS

It has been a humongous pleasure to work as a team and we are sincerely thankful to our project guide Sri.Ch.Sai Pratheek. His professionality in guidance and our diligence made this project successful. This whole course of project wouldn’t be possible without our reviewers Sri PNRL Chandra Sekhar and Sri SVG Reddy who had driven us in the path of progress.

I am extremely elated to extend my gratitude to **Dr. P.V. Nageswara Rao**, Professor and Head of Department of Computer Science and Engineering, for his encouragement all the way during this project.

I gratefully acknowledge **Prof. Lakshmi Prasad**, Principal, GITAM Institute of Technology, GITAM University, for giving us an opportunity in carrying out this project.

Abstract

As transferring images through internet and cellular phone become popular day by day, the importance of encrypting some of these also increases. To achieve that, many researchers proposed a lot of algorithms to encrypt image files. An algorithm for image encryption by using combination of Zigzag, Partitioning and Swapping is developed. In this algorithm encryption is done by rearranging the bits and also swapping of blocks of the image file. For decryption, key will guide to rearrange the partitions and then the procedure is just opposite of the encryption. As we are only rearranging the bit and block sequences the file size remains the same after encryption. Except this, the major advantage is that it needs a very short time to encrypt and decrypt but near to infinite time for cryptanalysis. In addition to this, the concept of steganography is also used before the image would undergo through the previously explained process. Before we apply the techniques of zigzag partitioning and swapping, we hide the information to be secured in the image by the technique of steganography using least significant bit algorithm. In the LSB approach, the basic idea is to replace the Least Significant Bits (LSB) of the cover image with the Bits of the messages to be hidden without destroying the property of the cover image significantly. Through this project the image itself is inaccessible along with the message which is hidden by using steganography. The use of Zigzag, partitioning and swapping along with steganography is much complex and the security thus achieved is of higher order.

Contents

1. Introduction

1.1 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 lines.

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.

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:

Antivirus software packages: These packages counter most virus threats if regularly updated and correctly maintained.

Secure network infrastructure: Switches and routers have hardware and software features that support secure connectivity, perimeter security, intrusion protection, identity services, and security management.

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 recipient.

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 an 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.

Modern cryptography is heavily based on mathematical theory and computer science practice; cryptographic algorithms are designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary. It is theoretically possible to break such a system, but it is infeasible to do so by any known practical means. These schemes are therefore termed computationally secure; theoretical advances, e.g., improvements in integer factorization algorithms, and faster computing technology require these solutions to be continually adapted. There exist information-theoretically secure schemes that provably cannot be broken even with unlimited computing power—an example is the one-time pad—but these schemes are more difficult to implement than the best theoretically breakable but computationally secure mechanisms.

The growth of cryptographic technology has raised a number of legal issues in the information age. Cryptography's potential for use as a tool for espionage and sedition has led many governments to classify it as a weapon and to limit or even prohibit its use and export. In some jurisdictions where the use of cryptography is legal, laws permit investigators to compel the disclosure of encryption keys for documents relevant to an investigation. Cryptography also plays a major role in digital rights management and copyright infringement of digital media.

1.3 Image Encryption:

As the internet speed is becoming higher and higher, information passing through video or IMAGE format becomes more popular then text. At the same time to ensure the IMAGE data security the necessity of effective encryption algorithm becoming higher and higher. Today, there are many powerful cryptographic algorithms in the marketplace for text data. These algorithms are not suitable for Image. Due to high size and the relationship between the pixels the value of any given pixel can be reasonably predicted from the value its neighbors. There are several algorithms to solve Image encryption problem. Most of them are complicated and take higher time and space for encryption and decryption. Some of them are not considered as secure.

So retrieving particular Image in a way that is both effective and efficient remains an open problem. It is necessary to design special encryption algorithms for multimedia data because of their special characteristics such as its coding structure, large amount of data, and real-time constraints. Among them the speed of Zigzag-Permutation Algorithm is very fast and is almost the same as the encoding/decoding time. But this algorithm compromised security as it does not withstand the known plaintext attack, therefore should not be considered as secure.

In our proposed algorithm rather than making it complex our intention is to change the bit sequences and also some of the blocks of the target file using several ways. And to remove the problem of zigzag permutation we use partitioning and swapping with zigzag. As the number of partitions and also number of swaps are not constant, it will not further vulnerable for known-plaintext attack. Moreover our proposed method will overcome the size problem as the encrypted file size will remain the same as input file.

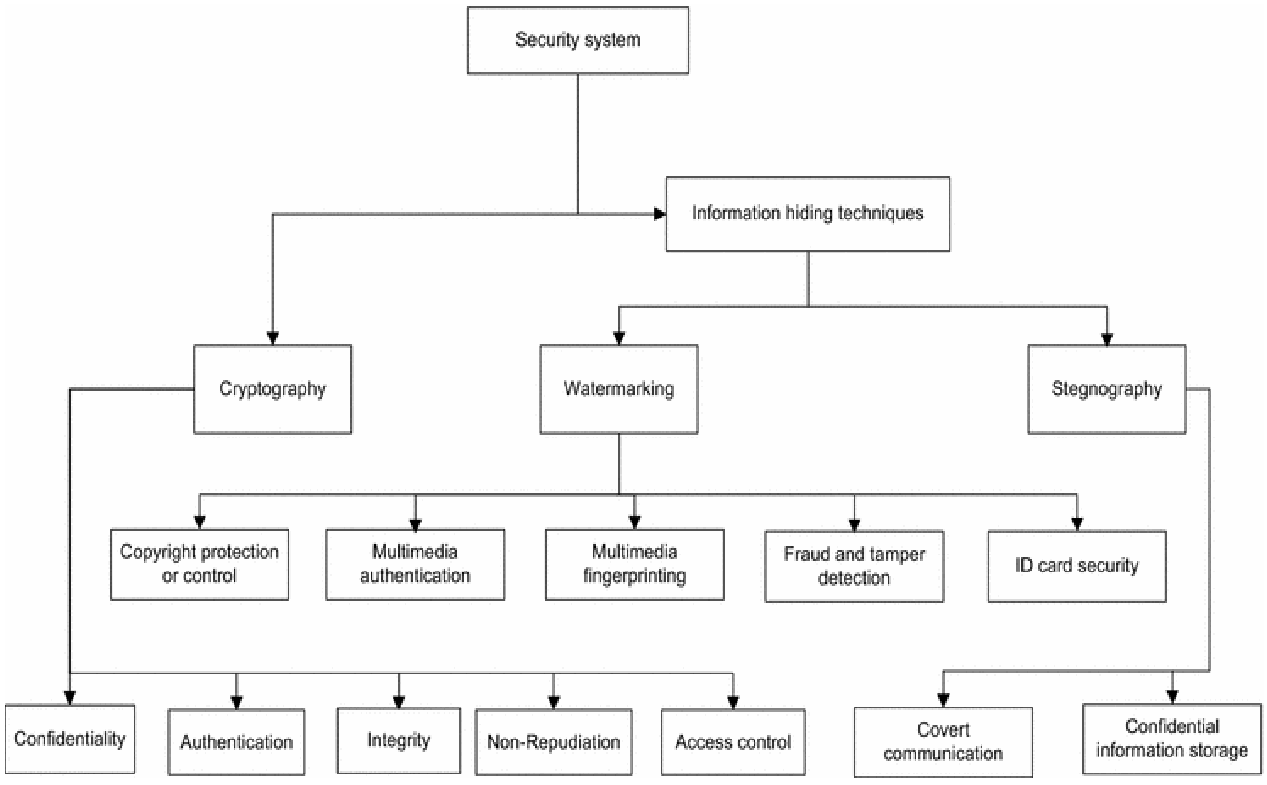
Advanced automated advances have made media data by and large available. Starting late, media procurements get essential in practice and along these lines security of sight and sound data has gotten standard concern. The central issues identifying with the issue of encryption has been inspected and moreover an outline on picture encryption strategies centered on chaotic techniques have been overseen in the present correspondence.

The chaotic image encryption could be made by using properties of chaos including deterministic elements, erratic conduct and non-straight change. This thought prompts routines that can in the meantime give security limits and a general visual check, which may be suitable in a couple of demands.

Automated pictures are for the most part used as a piece of diverse orders, that join military, genuine and helpful frameworks and these procurements need to control access to pictures and give the means to affirm uprightness of images.

The most aged and fundamental issue of cryptography is secure communication over a shaky channel. Party A needs to send to gathering B a mystery message over a correspondence line, which may be tapped by an enemy. The late developments in engineering, particularly in machine industry and correspondences, permitted possibly tremendous business for appropriating computerized interactive media content through the Internet.

Then again, the multiplication of advanced archives, picture handling apparatuses, and the overall accessibility of Internet access has made a perfect medium for copyright misrepresentation and wild appropriation of media, for example, picture, content, sound, and feature content.



Security system
Information hiding techniques
Cryptography
Watermarking
Stegnography
Copyright protection or control
Multimedia authentication
Multimedia fingerprinting
Fraud and tamper detection
ID card security
Confidentiality
Authentication
Integrity
Non-Repudiation
Access control
Covert communication
Confidential information storage

Fig-1

1.4 Steganography:

Steganography is a technique used to transmit a secret message from a sender to a receiver in a way such that a potential intruder does not suspect the existence of the message. Generally this can be done by embedding the secret message within another digital medium such as text, image, audio or video. The word steganography is of Greek origin and means “concealed writing” from the Greek words steganos meaning “covered or protected”, and graphie meaning “writing”. The first recorded use of the term was in 1499 by Johannes Trithemius in his Steganographia, a treatise on cryptography and steganography disguised as a book on magic. Generally, 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 a high security technique for long data transmission. There are various methods of steganography:

• Least significant bit (LSB) method

• Transform domain techniques

• Statistical methods

• Distortion techniques

Steganography under Various Media:

In the following sections we will try to show how steganography can and is being used through the media of text, images, and audio.

Often, although it is not necessary, the hidden messages will be encrypted. This principle states that the security of the system has to be based on the assumption that the enemy has full knowledge of the design and implementation details of the steganographic system. The only missing information for the enemy is a short, easily exchangeable random number sequence, the secret key. Without this secret key, the enemy should not have the chance to even suspect that on an observed communication channel, hidden communication is taking place. When embedding data, Bender reminds us that it is important to remember the following restrictions and features: -

•The cover data should not be significantly degraded by the embedded data, and the embedded data should be as imperceptible as possible. (This does not mean the embedded data needs to be invisible; it is possible for the data to be hidden while it remains in plain sight.)

• The embedded data should be directly encoded into the media, rather than into a header or wrapper, to maintain data consistency across formats.

• The embedded data should be as immune as possible to modifications from intelligent attacks or anticipated manipulations such as filtering and resembling.

• Some distortion or degradation of the embedded data can be expected when the cover data is modified. To minimize this, error-correcting codes should be used.

• The embedded data should be self-clocking or arbitrarily re-entrant. This ensures that the embedded data can still be extracted when only portions of the cover data are available. For example, if only a part of image is available, the embedded data should still be recoverable.

Steganography in Text

The example for hiding text in text is:

“Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suets and vegetable oils.”

Decoding this by taking the second letter in each word reveals the following message:

“Pershing sails from NY June 1.”

Methods of hiding text in text:

Line shift coding

In this method, text lines are vertically shifted to encode the document uniquely. Encoding and decoding can generally be applied either to the format file of a document, or the bitmap of a page image.

By moving every second line of document either 1/300 of an inch up or down, it is found that line-shift coding worked particularly well, and documents could still be completely decoded, even after the tenth photocopy.

Word shift coding

In word-shift coding, code words are coded into a document by shifting the horizontal locations of words within text lines, while maintaining a natural spacing appearance. This encoding can also be applied to either the format file or the page image bitmap. The method, of course, is only applicable to documents with variable spacing between adjacent words, such as in documents that have been text-justified. As a result of this variable spacing, it is necessary to have the original image, or to at least know the spacing between words in the un-encoded document.

Feature coding

A third method of coding data into text is known as feature coding. This is applied either to the bitmap image of a document, or to a format file. In feature coding, certain text features are altered, or not altered, depending on the code word. For example, one could encode bits into text by extending or shortening the upward, vertical end lines of letters such as b, d, h, etc. manually, but would be painstaking. Although this process can be automated, it can be made more challenging by varying the particular feature to be encoded. To even further complicate the issue, word shifting might be used in conjunction with feature coding, for example. Efforts such as this can place enough impediments in the attacker's way to make his job difficult and time-consuming.

Steganography in Audio

Because of the range of the human auditory system (HAS), data hiding in audio signals is especially challenging. The HAS perceives over a range of power greater than one billion to one and range of frequencies greater than one thousand to one. Also, the auditory system is very sensitive to additive random noise. Any disturbances in a sound file can be detected as low as one part in ten million (80dB below ambient level). However, while the HAS has a large dynamic range, it has a fairly small differential range - large sounds tend to drown quiet sounds.

When performing data hiding on audio, one must exploit the weaknesses of the HAS, while at the same time being aware of the extreme sensitivity of the human auditory system.

2. Literature Review:

- **An Improved Method for LSB Based Color Image steganography Combined with Cryptography Xinyi Zhou, Wei Gong, WenLong Fu, LianJing Jineach copyright 2016 IEEE ICIS 2016, June 26-29, 2016, Okayama, Japan**

- **Schneier, B., Applied Cryptography: Protocols, Algorithms. And Source Code in C, 2nd edition, John Wilen & Sons, Inc., 1996.**
- **L. Tang, "Methods for Encrypting and Decrypting MPEG Video Data Efficiently ," In Proceedings of The Fourth ACM International Multimedia Conference (ACM Multimedia'96), pages 219-230, Boston , MA, November 1996.**
- **J. But, G. Armitage, "An Evaluation of Current MPEG-1 Ciphers and their Applicability to Streaming Video,“ Australian Telecommunications Networks & Applications Conference 2004, (ATNAC2004), Sydney, Australia, December 8-10, 2004.**
- **X. Liu and A. M. Eskicioglu, "Selective Encryption of Multimedia Contents in Distribution Networks: Challenges and New Directions," IASTED International Conference on Communications, Internet, and Information Technology (CIIT 2003), Scottsdale, AZ, November 17-19,2003.**
- **J. But, G. Armitage, "An Evaluation of Current MPEG-1 Ciphers and their Applicability to Streaming Video,“ Australian Telecommunications Networks & Applications Conference 2004, (ATNAC2004), Sydney, Australia, December 8-10, 2004.**
- **M. A. Hashish, “The MPEG Encryption Algorithms, Survey and Analysis” http://www.mohamedhashish.com/index.php**

3. Project Scope

The scope of the system is that it provides security for IMAGE files and reducing the Encryption time and Decryption time. The sender using the symmetric key to encrypt and the receiver using same key to decrypt the same Image file. Here, we are implementing Image Encryption by using partitioning, zigzag and swapping and also we using this technique to secure multimedia files. With the use of steganography, the project as a whole offer maximum security. So, sender can encrypt the data to be hidden using both steganography and the image itself can be encrypted. The security hence offer is of higher level.

4. Project Objective

The system has a clear set of objectives to achieve. They are as follows:

- ➔ To hide the desired text in the image file using least significant bit (LSB) algorithm.
- ➔ The image is then encrypted using partitioning, zigzag and swapping.
- ➔ Offering security through steganography.
- ➔ Offering security by the three algorithms used.
- ➔ Main objective is to avoid the information retrieval by the unauthorized parties.

5. Project overview

The functional overview of the system is as follows:

1. The sender gives the Image file and two keys to the system.
2. The system allows us to choose the image file and the text to be hidden using steganography.
3. The system then generate the binary file for the corresponding Image file by using conversion technique.
4. Apply the partitioning technique and zigzag rule and swapping process in accordance with the key.

5. Convert the binary file to the Image file after applying the three processes and send it to the receiver.
6. The Receiver uses same key to Decrypt the Image files and he apply the partitioning technique and zigzag reverse rule and sapping process.
7. The receiver then extract the text hidden in the image file which is previously embedded using steganography.

6. Global Control Flow

Control flow is the sequencing of actions in a system. It defines the order of execution of operations. These decisions are based on external events generated by an actor or on the passage of time. These are two possible control flow mechanisms.

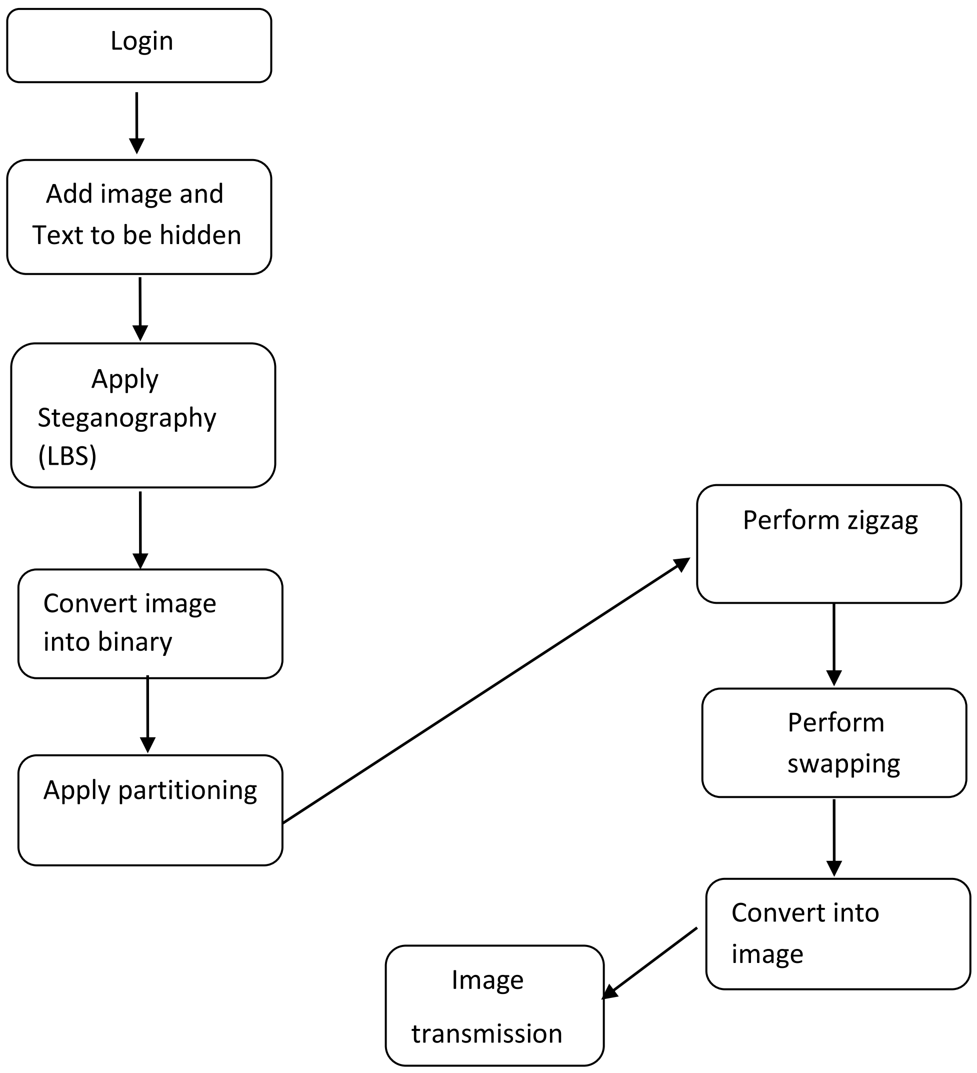


Fig-2

7. Algorithms or methods used:

7.1 Steganography Encoding and Decoding:

Encoding

In order to hide text in an image, the user must provide the text and a Target Image in which it is to be hidden. Optionally, users may enter or load a text key or generate a key from an image file. When a user key is not provided, a default key is used. To make choosing images easier, an image bar is provided for the user. The user may set the source directory, whose contents are displayed as thumbnails in the image bar located on the bottom panel of application. These thumbnails are automatically generated from the specified source directory. The user then selects a target image and loads it for use as the Key Image or the Target Image. The user may also open the Target Image by selecting Open Image under File menu. Additionally, the Key Image may be opened with the Open Image Key command on the Tools menu. The user may enter the text to be embedded in several ways. He or she may type it directly into the text window, open a text file by selecting Open Text under the File menu or use the Edit menu to paste from the clipboard. Loading a text key can be done from the Open Text Key selection on the Tools menu. When user has specified both the target image, and text body, this product is ready to hide the text body within the image. The user may then select Embed Text from the Tools menu. The key image will then be hashed into an appropriate Key Value. This value will be used for the encryption of the user's plaintext to produce cipher text. The cipher text and key value will then be input into the steganography and bit placement algorithms, and the Output Image will be generated. After the application has generated the image, the user may then inspect it by selecting the Image tab on the main panel. The user is then able to compare the encrypted target image with the original target image located on the right box of the main window. The user may wish to save the generated image by selecting Save Image or Save Image as…. under File menu.

Decoding

In order to retrieve the hidden text from the carrier, the user needs to provide a Source Image and any keys used when the source image was generated. Loading the source image file is done similarly to the process of opening a Target Image. If the sender has used the default key, the recipient need not load any keys to the application. The application is not able to identify the presence of the hidden message in images until the actual decoding process is attempted, as the image files are virtually indistinguishable from normal images.

Steganography using LSB insertion deals with secure communication between two parties-mainly hiding and revealing the text which is embedded with in a carrier image file.

This Application takes three steps to hide the data to make impossible for a third party to view without a key to view or edit the secret message.

1. Steganography algorithm: **S**teganography is the art of passing information in a manner that the very existence of the message is unknown. 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. Steganalysis is the art of discovering and rendering useless such covert messages. This can be achieved by the title LEAST SIGNIFICANT BIT INSERTION means hiding the binary converted data into the LSB bits of Digital image file chosen.

2. Security through obscurity: This is useful for adding another level of security to the Steganography by positioning the bits in random order. For example, if we had 4 bits to hide and the space available was more than 4 bits we can position the 4 data bits in a random order not like as 0,1,2,3.

3. Encryption Algorithm: The last level, which is used to highly strengthen the Steganography is encryption level. This means converting the meaningful information into un-understandable form which is nothing but Cipher. The generally used encryption algorithm is one among the advanced encryption standards.

Through the use of above three specified algorithms, this product will provide secure communication between parties. Finally this product can be a secure tool for communication as the three algorithms in conjunction would be very hard if not impossible for a keyless third party to break.

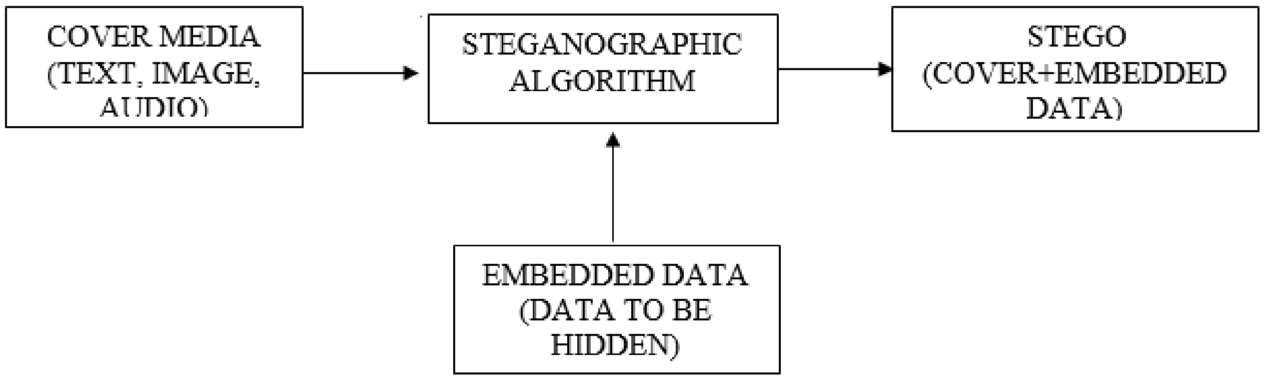

COVER MEDIA (TEXT, IMAGE, AUDIO)
STEGANOGRAPHIC ALGORITHM
STEGO (COVER+EMBEDDED DATA)
EMBEDDED DATA (DATA TO BE HIDDEN)

Fig-3 STEGANOGRAPHIC PROCESS

7.2 Least significant bit (LSB) insertion method:

One of the most common techniques used in steganography today is called least significant bit (LSB) insertion. Also called LSB (Least Significant Bit) substitution and it is the process of adjusting the least significant bit pixels of the carrier image. It is a simple approach for embedding message into the image. In this method some information from the pixel of the carrier image is replaced with the message information so that it can't be observed by the human visual system, therefore it exploits some limitations of the human visual system. The Least Significant Bit insertion varies according to number of bits in an image. For an 8-bit image, the least significant bit i.e. the 8th bit of each byte of the image will be changed by the 1-bit of secret message. For 24 bit image, the colors of each component like RGB (red, green and blue) will be changed. LSB steganography involves the operation on least significant bits of cover image, audio or video. The least significant bit is the lowest bit in a series of binary number. In LSB substitution the least significant bits of the pixels are displaced by the bits of the secret message which gives rise to an image with a secret message embedded in it. The method of embedding differs according to the number of bits in an image (different in 8 bit and 24 bit images).

When converting an analog image to digital format, we usually choose between three different ways of representing colors:

- 24-bit color: every pixel can have one in 2^24 colors, and these are represented as different quantities of three basic colors: red (R), green (G), blue (B), given by 8 bits (256 values) each.
- 8-bit color: every pixel can have one in 256 (2^8) colors, chosen from a palette, or a table of colors.
- 8-bit gray-scale: every pixel can have one in 256 (2^8) shades of gray.

For example, let us assume that we want to hide the letter “A” in the image using LSB insertion. The letter 'A' has an ASCII code of 65(decimal), which is 1000001 in binary. Hence, we need to hide **“1000001”** in the least significant bits of the image bits.

Let's say that the pixels before the insertion are:

10000000. 10100100. 10110101. 10110101. 11110011. 10110111. 11100111.

Then their values after the insertion of an 'A' will be:

1000000**1**. 1010010**0**. 1011010**0**. 1011010**0**. 1111001**0**. 1011011**0**. 1110011**1**. (The values in **bold** are the ones that were modified by the transformation)

For 24-bit images the modification can be extended sometimes to the second or even the third LSBs without being visible. 8-bit images instead have a much more limited space where to choose colors, so it's usually possible to change only the LSBs without the modification being detectable.

Result:

Fig 4 shows how actually the project interface looks when it is in the embedding phase. The figure shows the java swing windows designed in NetBeans IDE. It has a text area in which the message to be embedded or hidden in the image using LSB algorithm is to be inserted. Load button is used to load the image file in the local computer in which the previously entered text is to be hidden or embedded. The key must be entered in the secret key field which is further used in the extraction of text from the image (kind of authentication).

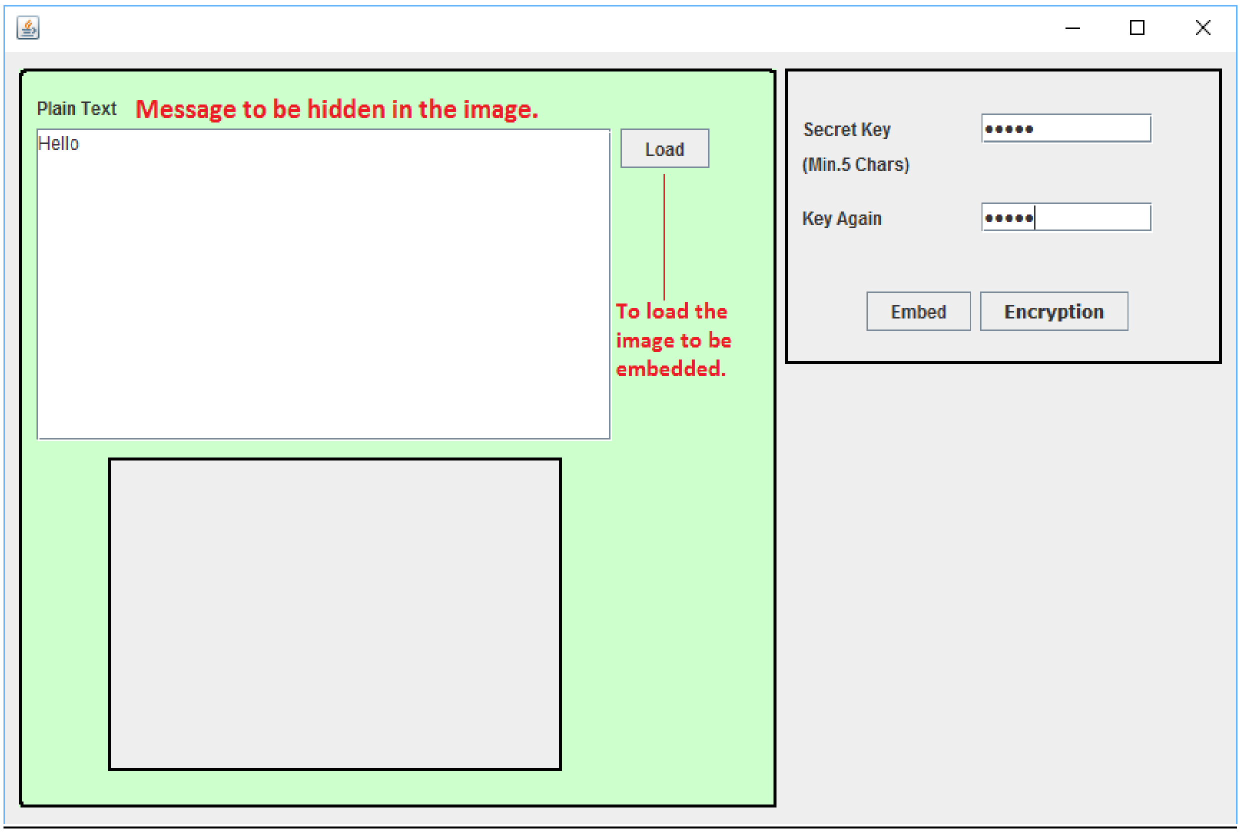


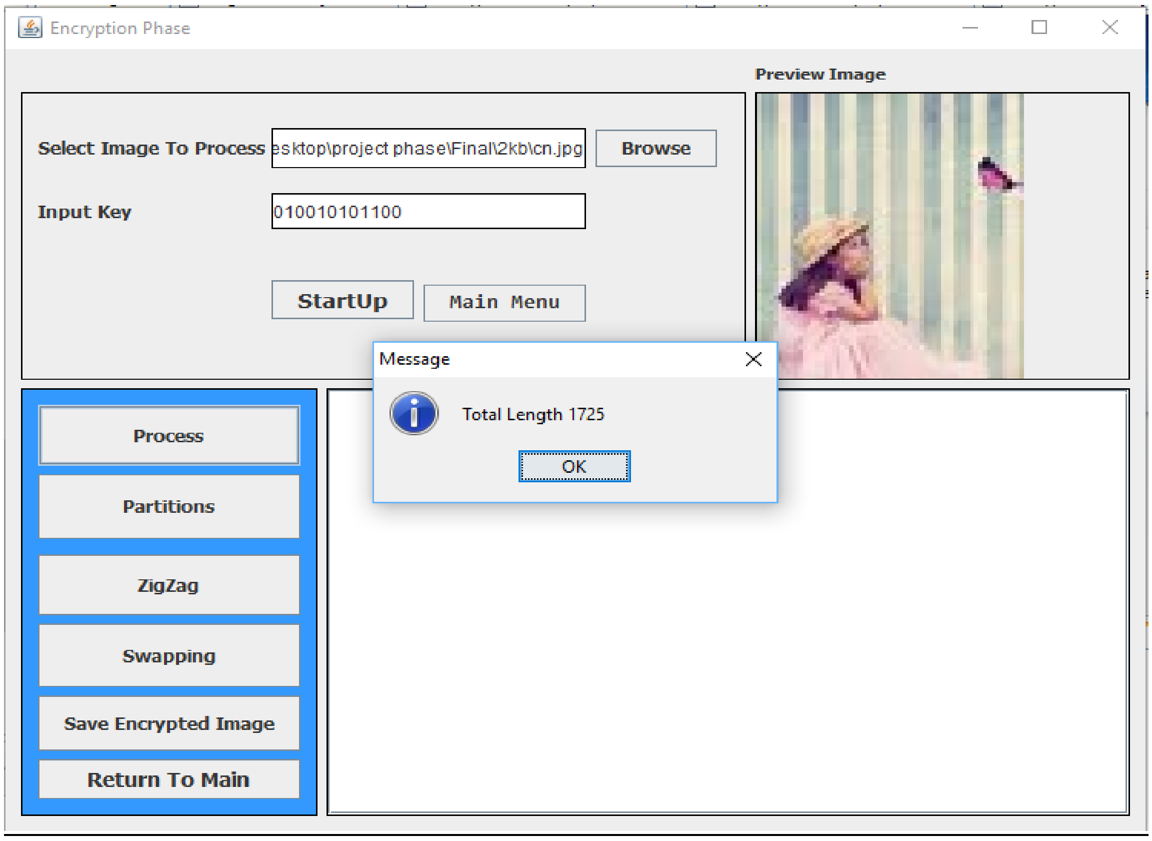
Fig-4

7.3 Convert into Binary:

The algorithms like partitioning, zigzag and swapping are to be applied in the bits of the image. For these algorithms to be implemented, the image should be converted into binary data in the first place. Hence, the process of encryption starts off with the conversion of chosen image into binary format. For this each pixel is converted into 8 bits or 1 byte.

Result:

The image which is chosen for encryption is of size 1.68 kb which is equivalent to 1725 bytes. **Fig-5** shows the result of the image being converted into binary. As soon as the "Process" button is clicked, the image is being loaded into the system and converted into binary format. Also, the dialogue box shows the total length of the image in bytes (*8 gives number of bits).



Select Image To Process: esktop\project phase\Final\2kb\cn.jpg

Input Key: 010010101100

Total Length 1725

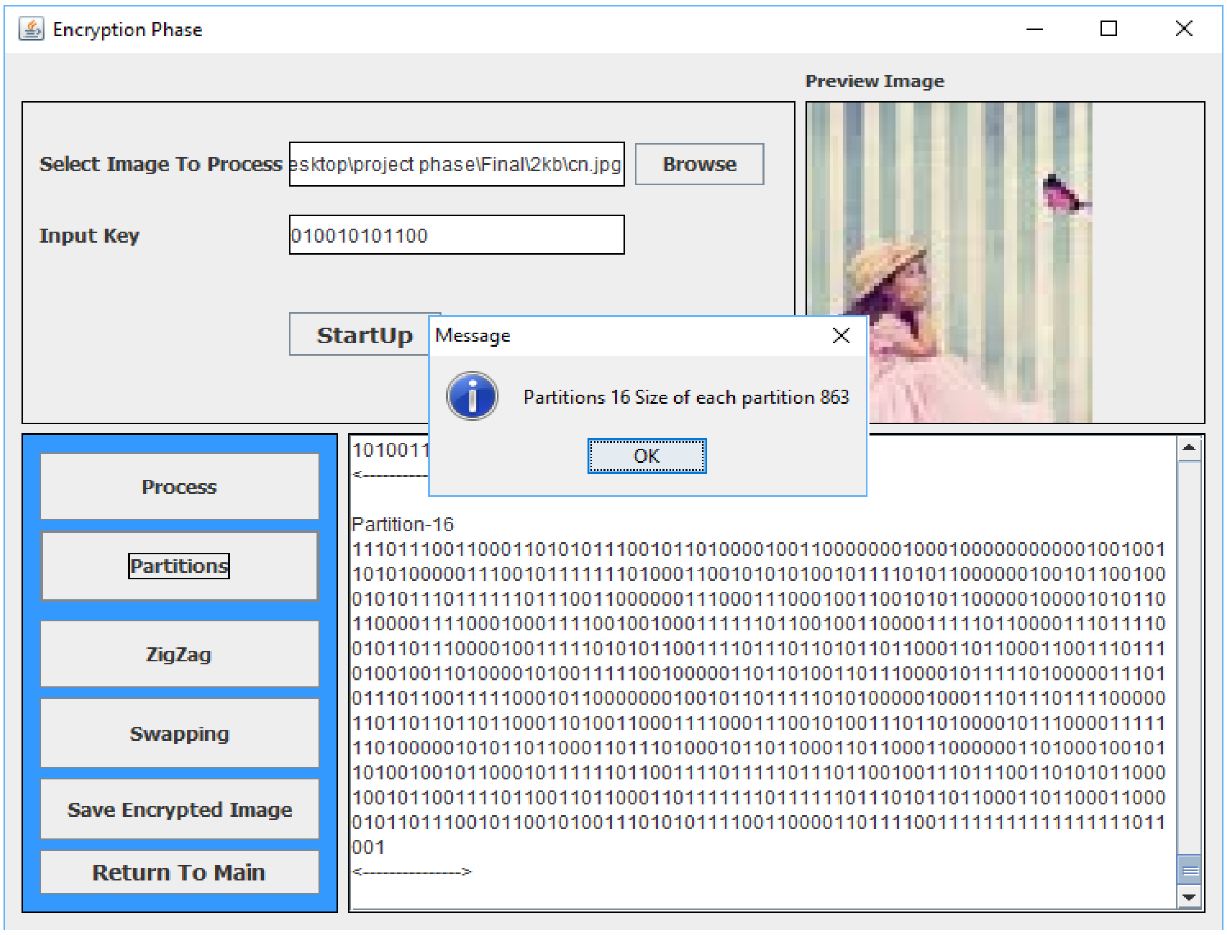
Fig-5

7.4 Partitioning Method:

Partitioning is possible in various ways. For example, let N = 4 bits. Using 4 bits we can partition bit sequences into 2^4 or 16 parts. Each part will get equal number of bits. If there is m-number of bits in the data file, each partition will get the ceiling of [M/2N] number of bits. So, except the last one each partition will get same number of bits. Let, the data bit is 1000 bits long. As N= 4 each partition will get (1000/2^4 = 62.5 or) 63 bits. And thus in the last or 2N th partition there will (1000 - 63x15) 55bits.

Result:

Fig-6 Shows the project interface in encryption phase. The user chooses the image which is previously embedded with hidden text using LSB algorithm and enter the key. The second process is to do partitions according to the input key. When the button “partitions is clicked, the swing shows the number of partitions and the bits in each partition as below. Also, the bits in each partition is displayed. In the scenario portrayed in the figure below, the first four digits in the input key is “**0100**” which is nothing but **4** when converted into decimal. Hence, the number of partitions is 2^4 = 16. Each partition gets 863 bits and displayed.



Select Image To Process: esktop\project phase\Final\2kb\cn.jpg

Input Key: 010010101100

Partitions 16 Size of each partition 863

```
1010011
<----------

Partition-16
111011100110001101010111001011010000100110000000100010000000000001001001
10101000001110010111111101000110010101010010111101011000000100101100100
01010111011111101110011000000111000111000100110010101100000100001010110
11000011110001000111100100100011111101100100110000111110110000111011110
01011011100001001111101010110011110111011010110110001101100011001110111
01001001101000010100111110010000011011010011011100001011111010000011101
01110110011111000101100000000100101101111101010000010001110111011100000
11011011011011000110100110001111000111001010011101101000010111000011111
11010000010101101100011011101000101101100011011000110000001101000100101
10100100101100010111111011001110111110111011001001110111001101010111000
10010110011110110011011000110111111101111110111010110110001101100011000
01011011100101100101001110101011110011000011011110011111111111111111011
001
<--------------->
```

Fig-6

7.5 Zigzag rule:

A line or course that proceeds by sharp turns in alternating directions is known as zigzag rule. By this rule first bit of selected bits will be placed in right position. Again second bit will place in left and third bit will be placed in right position and so on. Here is an example of zigzag:

Let’s select two bit BIT = “1010010100101011”

BIT_{Zigzag}= [Empty]

First bit of BIT = 1

It will be placed in the right position of BIT_{Zigzag}

So, BIT_{Zigzag}= “1”

Second bit of BIT = 0

It will be placed in the left position of BIT_{Zigzag}.

So, BIT_{Zigzag} = “01”

Third bit of BIT=1

It will be placed in the right position of BIT_{Zigzag}.

So, BIT_{Zigzag} = “011”

Forth bit of BIT= 0

It will be placed in the left position of BIT_{Zigzag}.

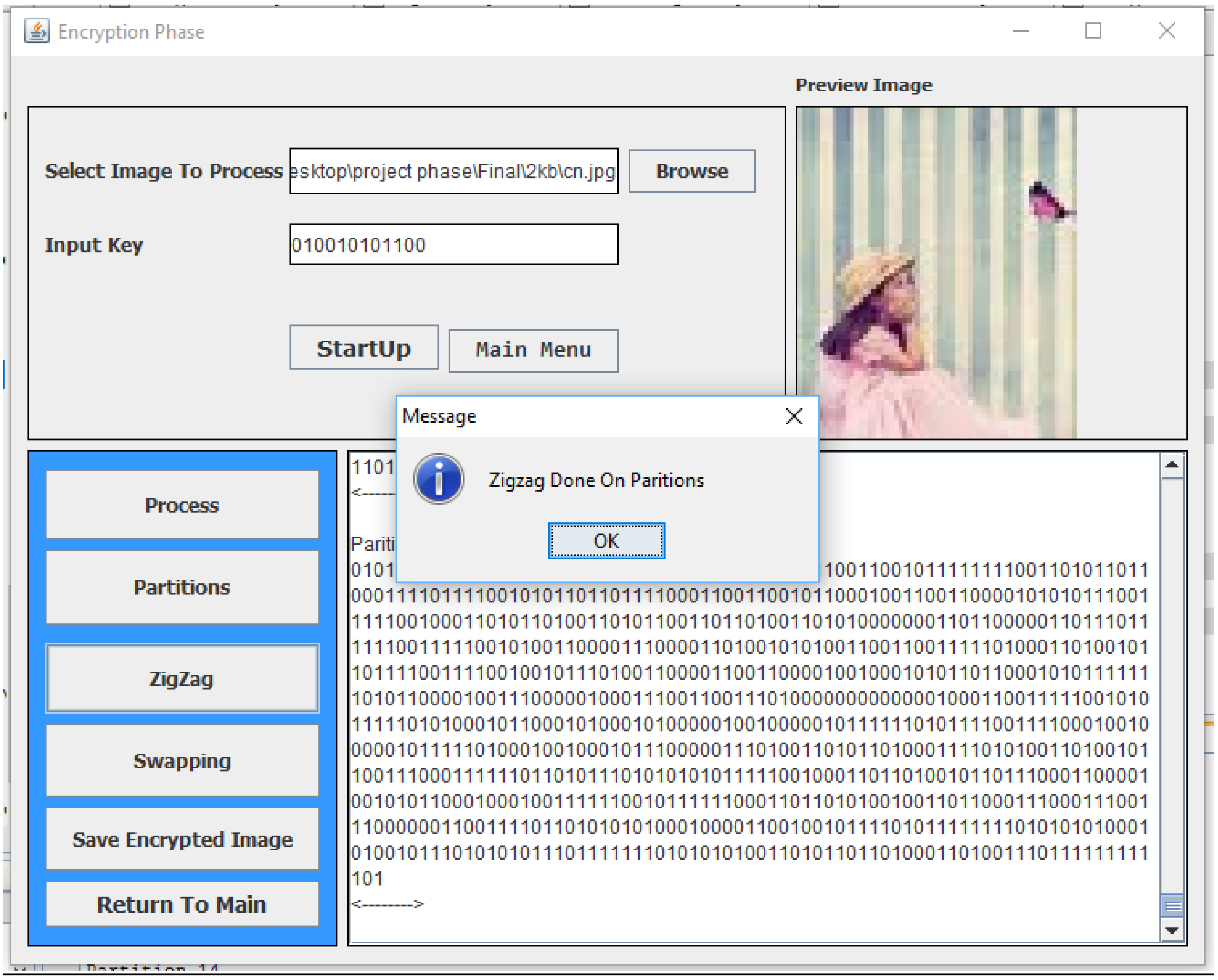
So, BIT_{Zigzag} = “0011”

After placing all bits BIT will become BIT_{Zigzag} = “1000110011000111”

By this procedure all bits of selected partition will be converted. Then position of the partitions will also change by swapping.

<u>Result:</u>

As said earlier the image has been converted into binary and the binary bits are divided into 16 partitions each having 863 bits. Next process in the encryption phase is to zigzag rule. Zigzag is process is discussed above taking 8 bits as an example. This technique is applied on all the 16 partitions each having 863 bits. **Fig-7** shows the output screen when zigzag is applied on the partitions. It also displays all the 16 partitions after zigzag is applied.



Select Image To Process esktop\project phase\Final\2kb\cn.jpg

Input Key 010010101100

Zigzag Done On Paritions

```
1101
<-----

Pariti
0101                                             1001100101111111100110101011011
000111101111100101011011011110001100110010110001001100110000101010111001
1111001000110101101001101011001101101001101010000000110110000011011101 1
11110011111001010011000011100001101001010100110011001111101000110100101
10111100111100100101110100110000110011000010010001010110110001010111111
10101100000100111000000100011100110011101000000000000001000110011111001010
111110101000101100010100010100000100100000010111111010111100111100010010
00001011111010001001000101110000011101001101011010001111010100110100101
10011100011111101101011101010101011111001000110110100101101110001100001
00101011000100010011111100101111110001101101010010011011000111000111001
11000000110011110110101010100010000110010010111101011111110101010100 01
01001011101010101110111111110101010100110101101101000110100111011111111
101
<--------->
```

Fig-7

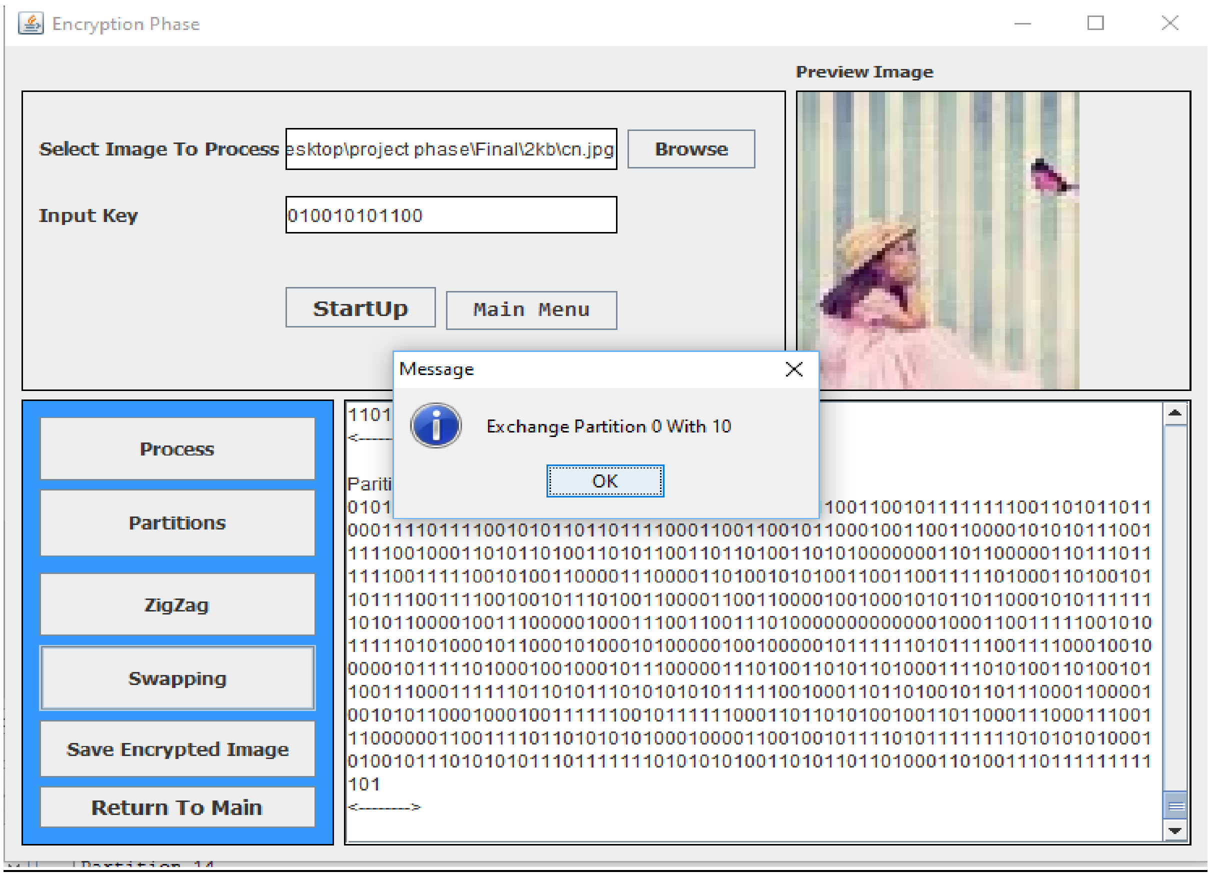
7.6 Swapping method:

In our key first i-bit represent the value of N and remaining j-bits used for swapping. Each N number of bits from j will select the partitions for swapping. Number of swap (M) = j/N;

Let, KEY = 0100 1101 0011. Here i = 4 so j = key-4 = 8. So, M= 8/4 = 2 swaps. As, N = 0100 b = 4d there will 2^4 = 16 partitions. Then j1=1101b = 13d, so 13th partition will swap with the 1st partition.

Result:

Swapping is a simple technique but when applied along with partitioning and zigzag, offers strength to the encryption phase. Swapping is done according to the input key, 5th bit to 12th or last bit. . As shown in the **Fig-8**, we have taken the input key as “**010010101100**”. Where, 5th to 8th bits give “**1010**” which is 10 when converted into decimal and 9th to 12th bits give “**1100**” which is 12 when converted into decimal. Hence according to the algorithm, we will swap 0th and 1st partitions with 10th and 12th partitions respectively. **Fig-9** shows the output screen when 1st partition is swapped with 12th partition.



Select Image To Process: esktop\project phase\Final\2kb\cn.jpg

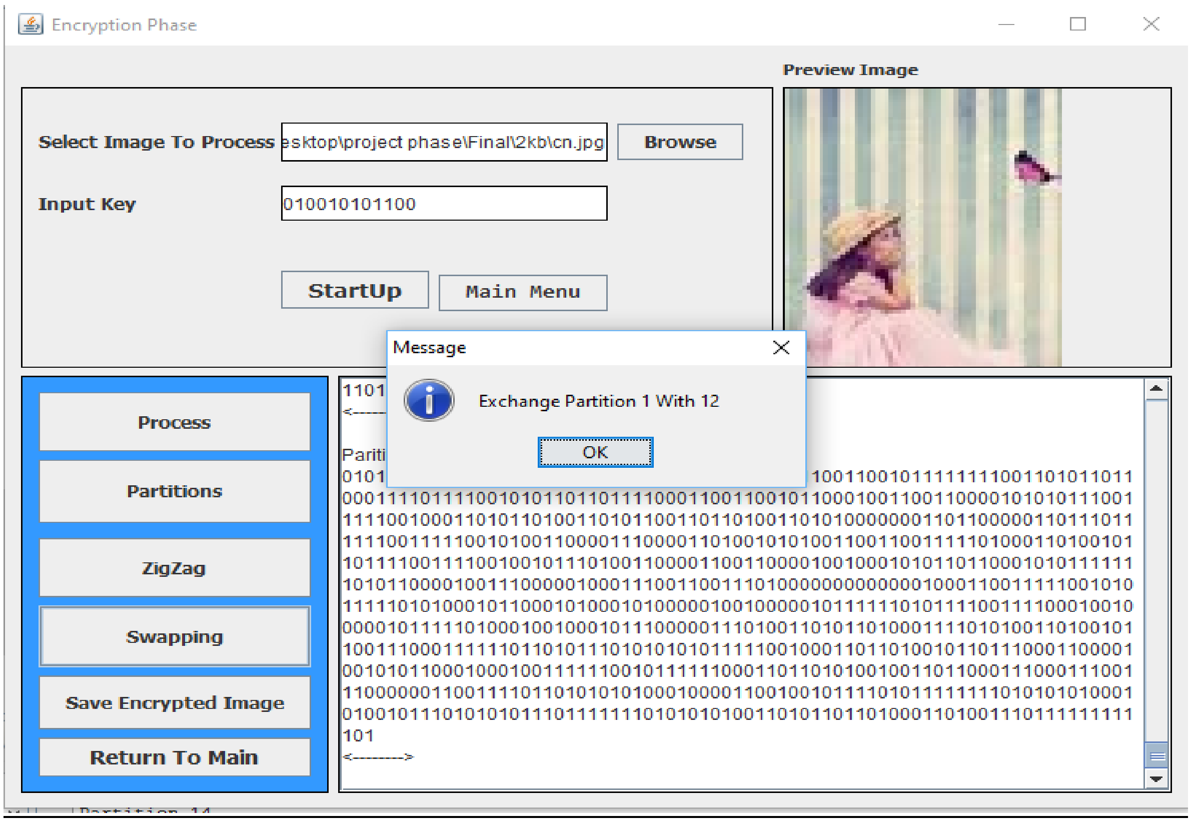
Input Key: 010010101100

Message: Exchange Partition 0 With 10

```
1101
<-------

Pariti
0101                                              10011001011111111001101011011
00011110111100101011011011110001100110010110001001100110000101010111001
11110010001101011010011010110011011010011010100000001101100000110111011
1111001111100101001100001110000110100101010011001100111101000110100101
10111100111100100101110100110000110011000010010001010110110001010111111
10101100001001110000010001110011001110100000000000000100011001111001010
1111101010001011000101000101000001001000001011111101011100111100010010
00001011111010001001000101110000011101001101011010001111010100110100101
10011100011111101101011101010101011111001000110110100101101110001100001
00101011000100010011111100101111110001101101010010011011000111000111001
11000000110011110110101010100010000110010010111101011111111010101010001
01001011101010101110111111101010101001101011011010001101001110111111111
101
<------->
```

Fig-8



esktop\project phase\Final\2kb\cn.jpg

010010101100

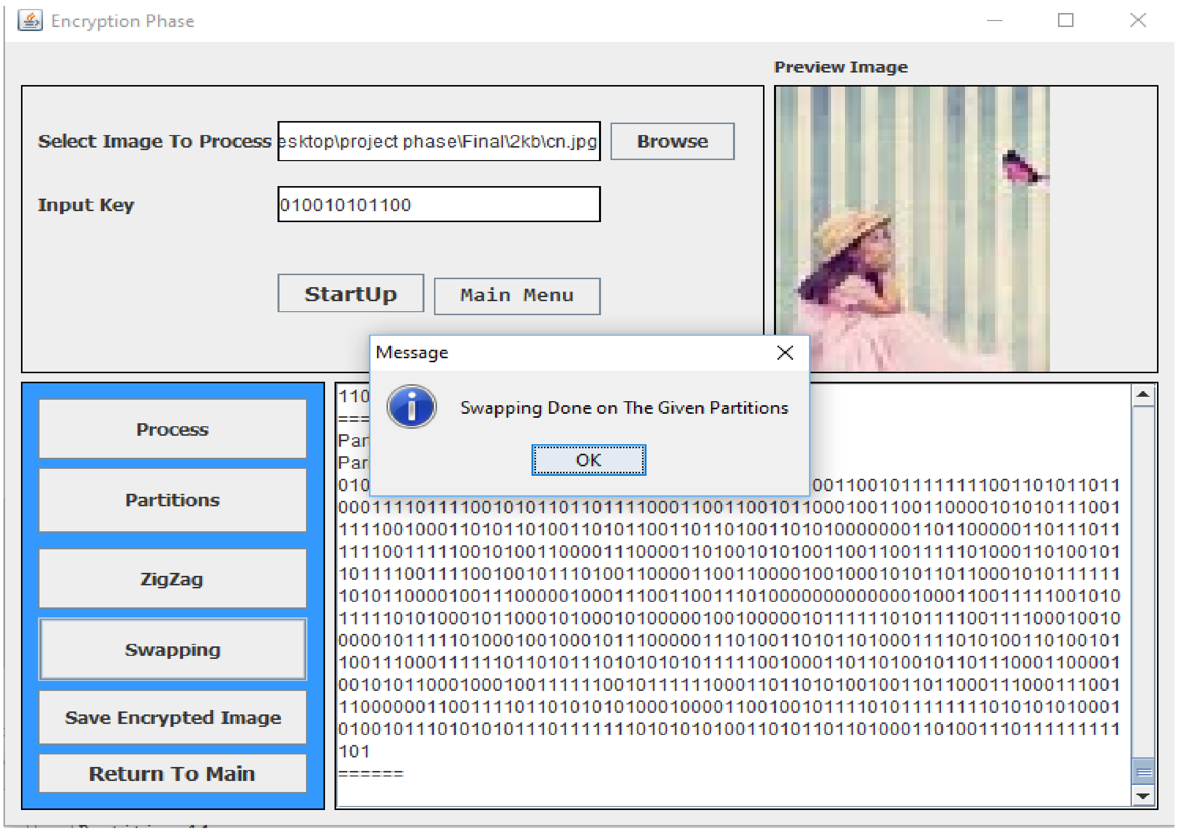
Exchange Partition 1 With 12

```
1101
<-----

Pariti
0101                                   10011001011111111001101011011
00011110111100101011011011110001100110010110001001100110000101010111001
11110010001101011010011010110011011010011010100000001101100000110111011
11110011111001010011000011100001101001010100110011001111101000110100101
10111100111100100101110100110000110011000010010001010110110001010111111
10101100001001110000001000111001100111010000000000000001000110011111001010
11111010100010110001010001010000010010000010111111010111100111100010010
00001011111010001001000101110000011101001101011010001111010100110100101
10011100011111101101011101010101011111001000110110100101101110001100001
00101011000100010011111100101111110001101101010010011011000111000111001
11000000110011110110101010100010000110010010111101011111111101010101010001
01001011101010101110111111110101010100110101101101000110100111011111111
101
<-------->
```

Fig-9

Fig-10 marks the end of the encryption phase. The dialogue box shows that the swapping is done on the given partitions. As shown in the figure, when the button “**Save Encrypted Image**” is clicked, the image on which steganography, partitioning and swapping is done is saved to the location specified in the code. The image’s metadata is preserved in the first 4 pixels or first 32 bits in the binary format. Hence, during swapping, this metadata, which is used by the operation system to recognize that it is an image is disturbed. As the metadata is disturbed, the OS cannot recognize that the file is an image and fail to load the necessary program to open it. As a result the encrypted image cannot be opened by the operating system and it shows that “**Windows cannot open this file**”.


Encryption Phase

Select Image To Process: esktop\project phase\Final\2kb\cn.jpg

Input Key: 010010101100

Message: Swapping Done on The Given Partitions

```
110
===
Par
Par
010                                     0011001011111111001101011011
00011110111100101011011011110001100110010110001001100110000101010111001
111100100011010110100110101100110110100110101000000011011000000110111011
1111001111100101001100001110000110100101010011001100111101000110100101
10111100111100100101110100110000110011000010010001010110110001010111111
1010110000100111000000100011100110011101000000000000001000110011111001010
1111101010001011000101000101000000100100000010111111010111100111100010010
00001011111010001001000101110000011101001101011010001111010100110100101
10011100011111101101011101010101011111001000110110100101101110001100001
00101011000100010011111100101111110001101101010010011011000111000111001
11000000110011110110101010100010000110010010111101011111111010101010001
010010111010101011101111111010101010011010110110100011010011101111111111
101
======
```

Fig-10

8. Decryption

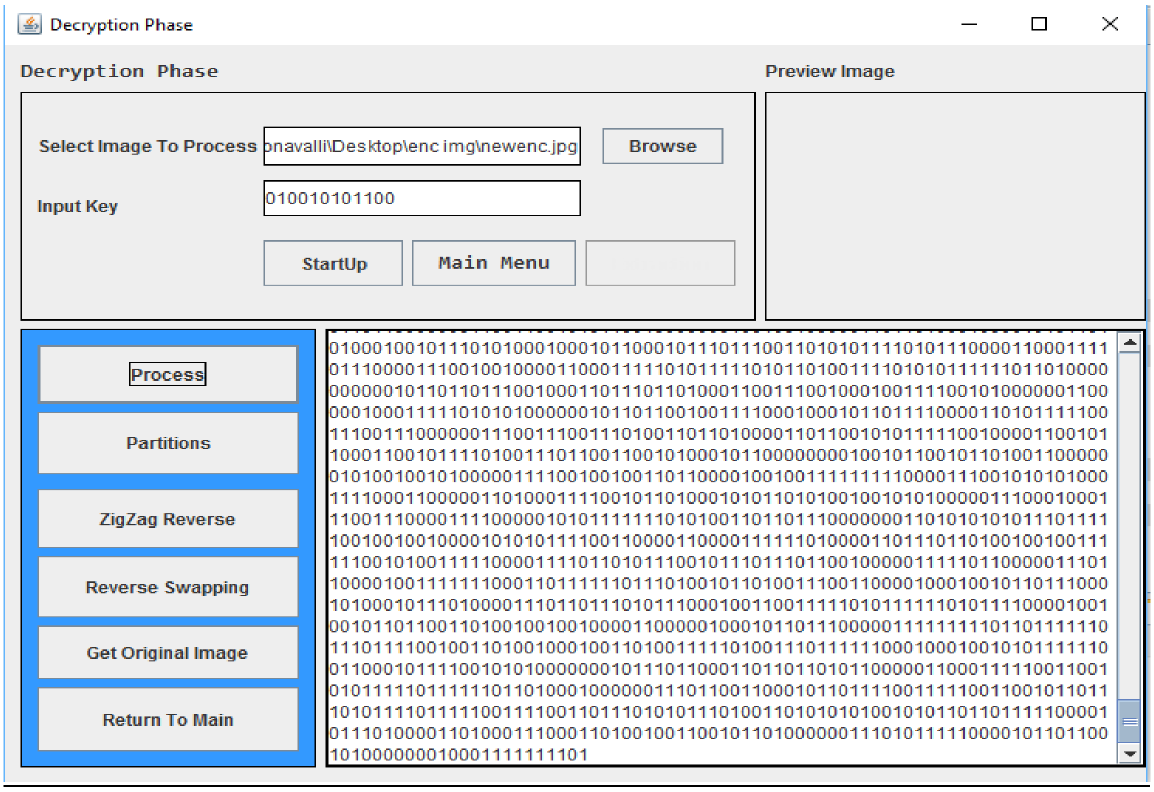
Decryption process generally depends on the key. As KEY = i+j, so from first i-bits the value of N can be determined. Then each N-bit from j will select the partitions to swap. For example, when KEY = 0100 1101 0011 and i = 4: 0100 4 = N so there are 2^N or 16 partition. Remaining 8 bits (j) divided by 4 will give M = 2. Now each N or 4-bit will converted into decimal and declare which partitions to swap.As we have got the value of N, following the same procedure used in encryption method number of bits in each partition can be determinable. Then reverse of zigzag rule will apply on each partition. After that, by rearranging the partitions in reverse order) according to the key will return the original data bits.

8.1 Convert into binary:

As decryption is nothing but the reverse of the encryption process, all the processes in the encryption phase has to be reverted in the decryption phase in a particular sequence. To implement the decryption techniques, which deals with the bits of the image, the image must be converted into binary in the first place just as done in the encryption.

<u>Result:</u>

Fig-11 shows the project output screen at the starting of the decryption phase. Just as done in the encryption, the image is converted into binary format, where each pixel in converted into 1 byte or 8 bits. In the decryption phase, the image which is saved at the end of encryption phase is loaded with the use of browse button. And the receiver has to enter the input key which is same as the input key used during encryption. Later on, when the "**Process**" button is clicked, the image in converted into binary and shows all the bits of the image on the screen.



Select Image To Process: onavalli\Desktop\enc img\newenc.jpg

Input Key: 010010101100

```
010001001011101010001000101100010111011100110101011101011100001100011111
011100001110010010000110001111101011111010110100111101010111111011010000
000000101101101110010001101110110100011001110010001001111001010000001100
000100011111010101000000101101100100111100010001011011110000110101111100
111001110000001110011100111010011011010000110110010101111100100001100101
100011001011110100111011001100101000101100000000100101100101101001100000
010100100101000001111001001001101100001001001111111110000111001010101000
111100011000001101000111100101101000101011010100100101010000011100010001
110011100001111000001010111111101010011011011100000001101010101011101111
100100100100001010101110011000011000011111010000110111011010010010011 1
111001010011111000011110110101110010111011101100100000111110110000011101
100001001111111000110111111101110100101101001110011000010001001011011000
101000101110100001110110111010111000100110011111010111111010111000010 01
001011011001101001001001000011000001000101101110000011111111101101111110
111011110010011010010001001101001111010011101111110001000100101011111 10
011000101111001010100000001011101100011011011010110000011000111110011001
010111110111111011010001000000111011001100010110111001111100110010110 11
101011110111110011110011011101010111010011010101010010101101101111100001
011101000011010001110001101001001100101101000000111010111110000101101100
1010000000100011111111101
```

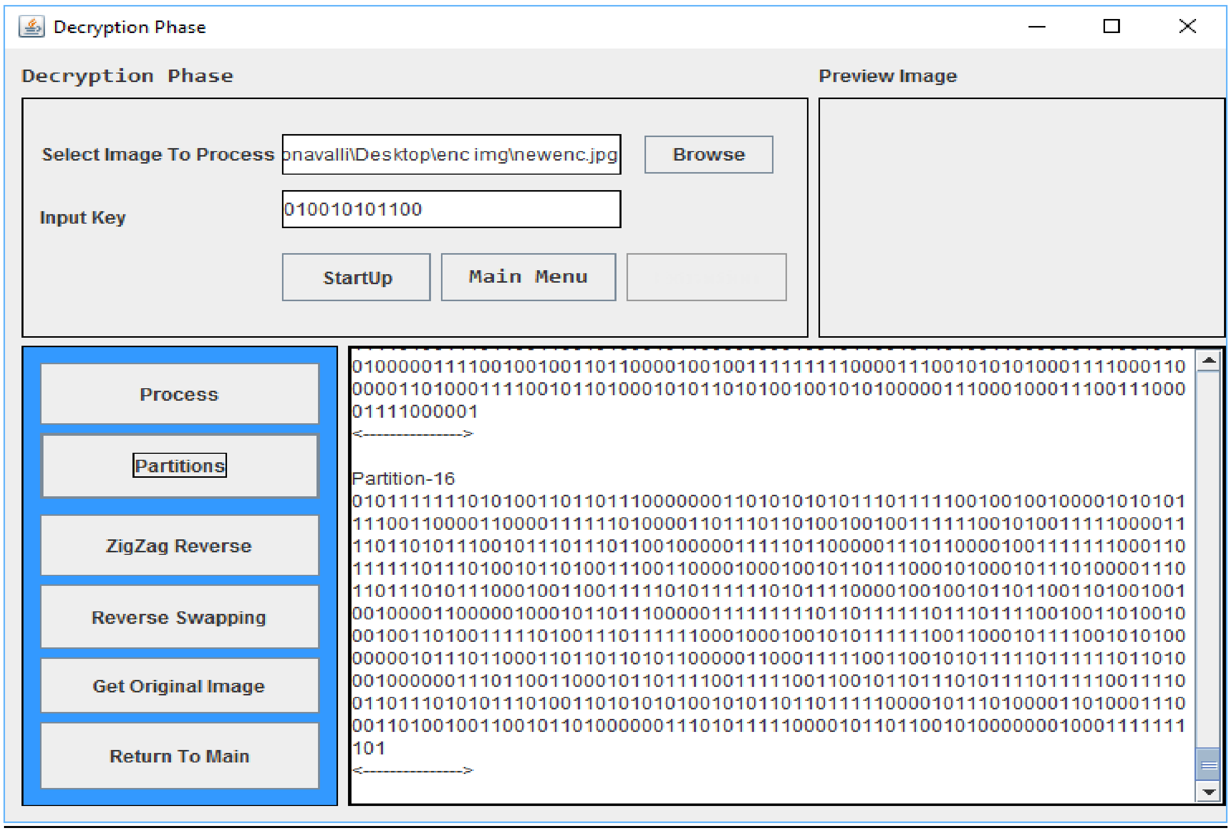
<u>Fig-11</u>

8.2 Partitioning:

The process of partitioning is also done in the same way as done in the encryption phase. Partitioning is done according to the key entered by the receiver which is the one used during encryption. If same key isn't used, the original image cannot be retrieved as the whole process of decryption is done in accordance with the input key. So, the key entered is "**010010101100**". The first 4 bits i.e "**0100**" which is 4 when converted into decimal. Hence, the number of partitions will be 2^4=16 partitions.

Fig-12 show the java swing window resulting when the "Partitions" button is clicked. It displays all the 16 partitions on the screen.

<u>Result:</u>



Select Image To Process: pnavalli\Desktop\enc img\newenc.jpg

Input Key: 010010101100

```
0100000111100100100110110000100100111111111000011100101010100011110001100
0000110100011110010110100010101101010010010101000001110001000111001110000
01111000001
<--------------->


Partition-16
0101111111010100110110111000000011010101010111011111001001001000010101010
1110011000011000011111101000011011101101001001001111110010100111110000111
1101101011100101110111011001000001111101100000111011000010011111110001100
1111110111010010110100111001100001000100101101110001010001011101000011100
1101110101110001001100111110101111110101110000100100101101100110100100100
0010000110000010001011011100000111111111011011111101101101110010011010010
0010011010011110100111011111100010001001010111111001100010111100101010000
0000010111011000110110110101100000110001111100110010101111101111111010100
0010000000111011001100010110111100111110011001011011101011110111110011100
0110111010101110100110101010100101011011011111000010111010000110100011100
0011010010011001011010000001110101111100001011011001010000000100011111101
101
<--------------->
```

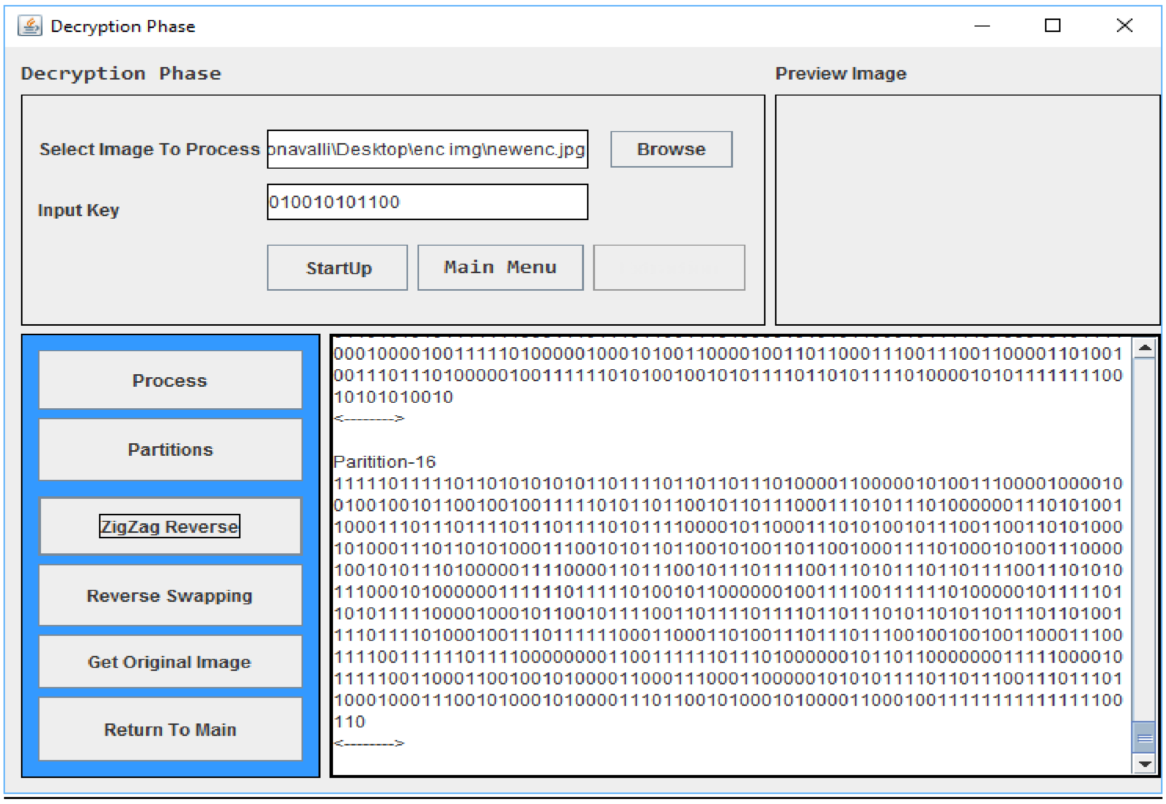
<u>Fig-12</u>

8.3 Zigzag Reverse:

Previously, in the encryption phase, zigzag is applied on all the partitions. So, to decrypt, which is nothing but reverting the process of encryption, zigzag performed on the partitions has to be undone. To undo the process of zigzag, we perform the zigzag reverse algorithm on all the partitions. So, after performing zigzag reverse, the partitions come to same state as earlier i.e the orientation of the original image.

<u>Result:</u>

Fig-13 Shows the result screen of the partitions after implementing zigzag reverse. When the button “**Zigzag Reverse**” is clicked, zigzag reverse algorithm is applied on each partitions undoing the zigzag done on these partitions in the encryption phase.



Select Image To Process onavalli\Desktop\enc img\newenc.jpg

Input Key 010010101100

```
000100001001111101000001000101001100001001101100011100111001100001101001
001110111010000010011111101010010010101111011010111101000010101111111100
10101010010
<-------->

Paritition-16
111110111110110101010101101111011011011101000011000001010011100001000010
010010010110010010011111010110110010110111000111010111010000001110101001
100011101110111101110111101011110000101100011101010010111001100110101000
101000111011010100011100101011011001010011011001000111101000101001110000
100101011101000000111100001101110010111011110011101011101101111001110101 0
111000101000000111111011111010010110000000100111100111111010000001011111 01
101011111000010001011001011110011011110111101101110101101011011101101001
111011110100010011101111110001100011010011101110111001001001001100011100
111100111111011110000000011001111110111010000000101101100000001111000010
111110011000110010010100001100011100011000001010101111011011100111011101
100010001110010100010100001110110010100010100001100010011111111111111100
110
<-------->
```

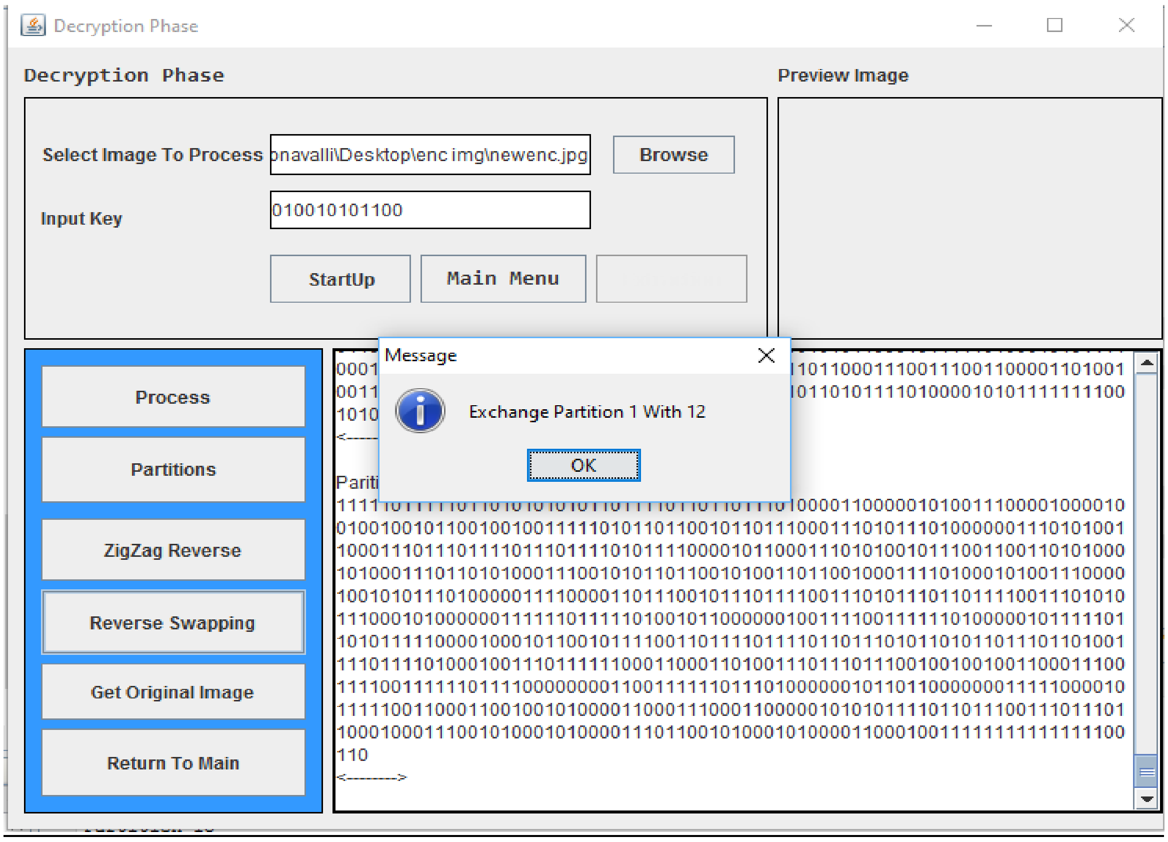
<u>**Fig-13**</u>

8.4 Reverse Swapping:

Earlier in encryption phase, swapping was done according to the 5th to 12th bits in the key. In the same way, to decrypt, the swapping done in the encryption phase has to be undone. This undoing is achieved by reverse swapping.

Result:

Fig-14 shows the output when the button "**Reverse Swapping**" is clicked. During encryption, 0th partition is swapped with 10th and 1st partition is swapped with 12th partition. To reverse this process the same has to be done again. Thus, as a result the partitions align same as in the original image. Finally the bits now have same orientation as of the original image.



```
onavalli\Desktop\enc img\newenc.jpg
010010101100
```

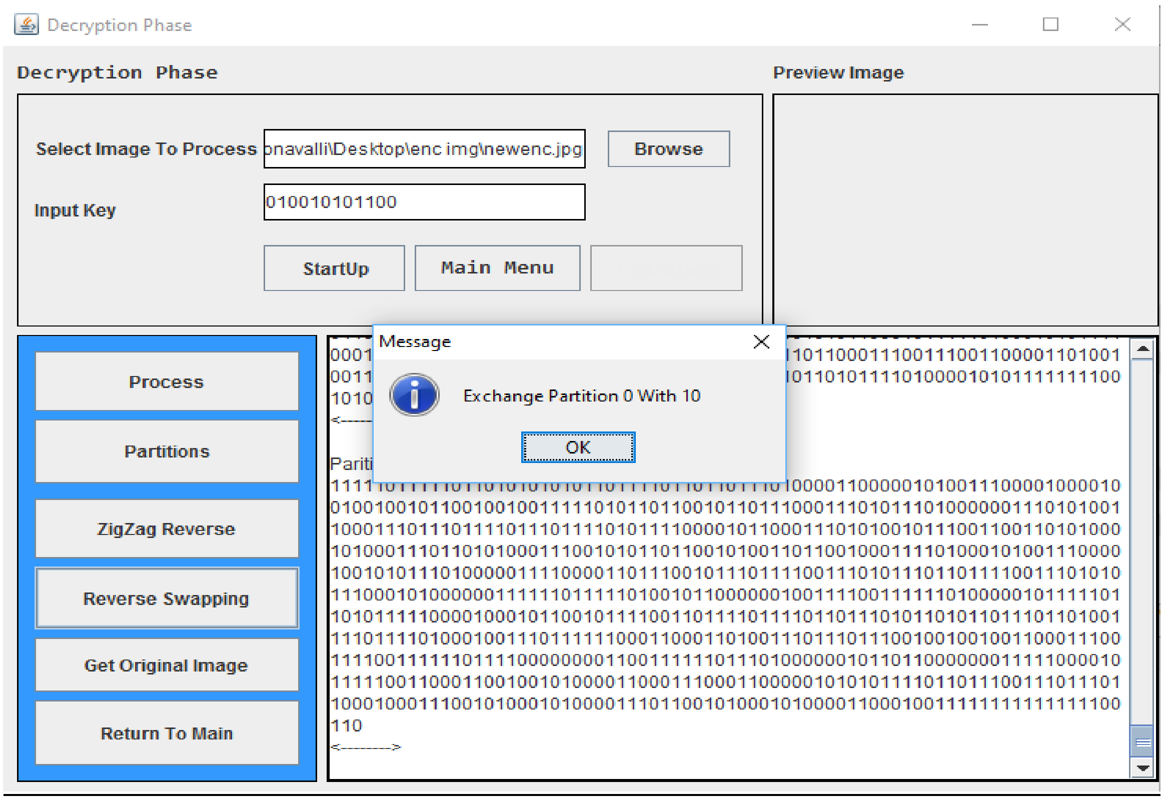
Exchange Partition 1 With 12

```
0001                    1011000111001110011000011010 01
0011                    0110101111010000101011111111100
1010
<------

Pariti
1111101111110110101010101101111011011011101000011000001010011100001000010
0100100101100100100111110101101100101101110001110101110100000011101010 01
1000111011101111011101111010111100001011000111010100101110011001101010 00
1010001110110101000111001010110110010100110110010001111010001010011100 00
1001010111010000011110000110111001011101111001110101110110111100111010 10
1110001010000001111110111110100101100000010011110011111010000010111110 1
1010111110000100010110010111100110111101111011011101011010110111011010 01
1110111101000100111011111100011000110100111011101110010010010011000111 00
1111001111110111100000000110011111101110100000010110110000000111110000 10
1111100110001100100101000011000111000110000010101011110110111001110111 01
1000100011100101000101000011101100101000101000011000100111111111111111 00
110
<--------->
```

Fig-14

Fig-15 Marks end of the decryption algorithms. The reverse swapping process is completed by swapping 0th and 1st partitions with 10th and 12th partitions respectively. All the decryption processes are implemented to the encrypted image to obtain the original image. Next step is to convert these bits into image and original image is retrieved.



onavalli\Desktop\enc img\newenc.jpg

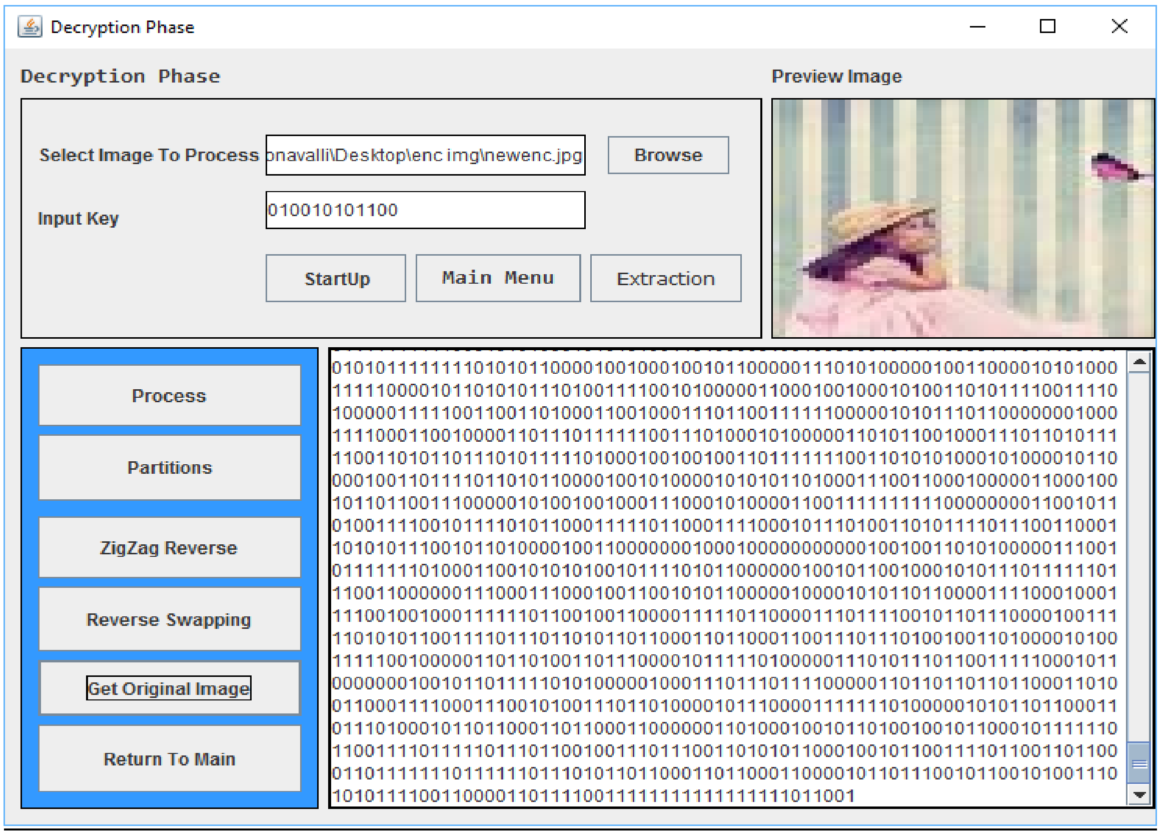
010010101100

Exchange Partition 0 With 10

```
0001                    1011000111001110011000011010 01
0011                    1011010111101000010101111111100
1010
<-----

Pariti
111110111111011010101010110111101101101110100001100000101001110000100001 0
010010010110010010011111010110110010110111000111010111010000001110101001
100011101110111110111011110101111000010110001110101001011100110011010100 0
101000111011010100011100101011011001010011011001000111101000101001110000 0
100101011101000001111000011011100101110111100111010111011011110011101010
111000101000000111111011111101001011000000100111100111111010000010111110 1
101011111000010001011001011110010111101111011011101011010110111011010 01
111011110100010011101111111000110001101001110111011100100100100110001110 0
111100111111011110000000011001111110111010000000101101100000001111100001 0
111110011000110010010100001100011100011000001010101111011011100111011101
100010001110010100010100001110110010100010100001100010011111111111111100
110
<-------->
```

Fig-15



onavalli\Desktop\enc img\newenc.jpg

010010101100

```
010101111111101010110000100100010010110000011101010000010011000010101000
111110000101101010111010011110010100000110001001000101001101011110011110
100000111110011001101000110010001110110011111100000101011101100000001000
111100011001000011011101111110011101000101000001101011001000111011010111
110011010110111010111110100010010010011011111110011010101000101000010110
000100110111101101011000010010100001010101101000111001100010000011000100
101101100111000001010010010001110001010000110011111111110000000011001011
010011110010111101011000111110110001111000101110100110101111011100110001
101010111001011010000100110000000100010000000000010010011010100000111001
011111110100011001010101001011110101100000010010110010001010111011111101
110011000000111000111000100110010101100000100001010110110000111100010001
111001001000111111011001001100001111101100001110111100101101110000100111
110101011001111011101101010110110001101100011001110111010010011010000101 00
111110010000011011010011011100001011111010000011101011101100111110001011
000000010010110111110101000001000111011101111000001101101101101100011010
011000111100011100101001110110100001011100001111111010000010101101100011
011101000101101100011011000110000001101000100101101001001011000101111110
110011110111111011101100100111011100110101011000100101100111101100110110 0
011011111110111111011101010110001101100011000010110111001011001010011 10
10101111001100001101111001111111111111111111011001
```

Fig-16

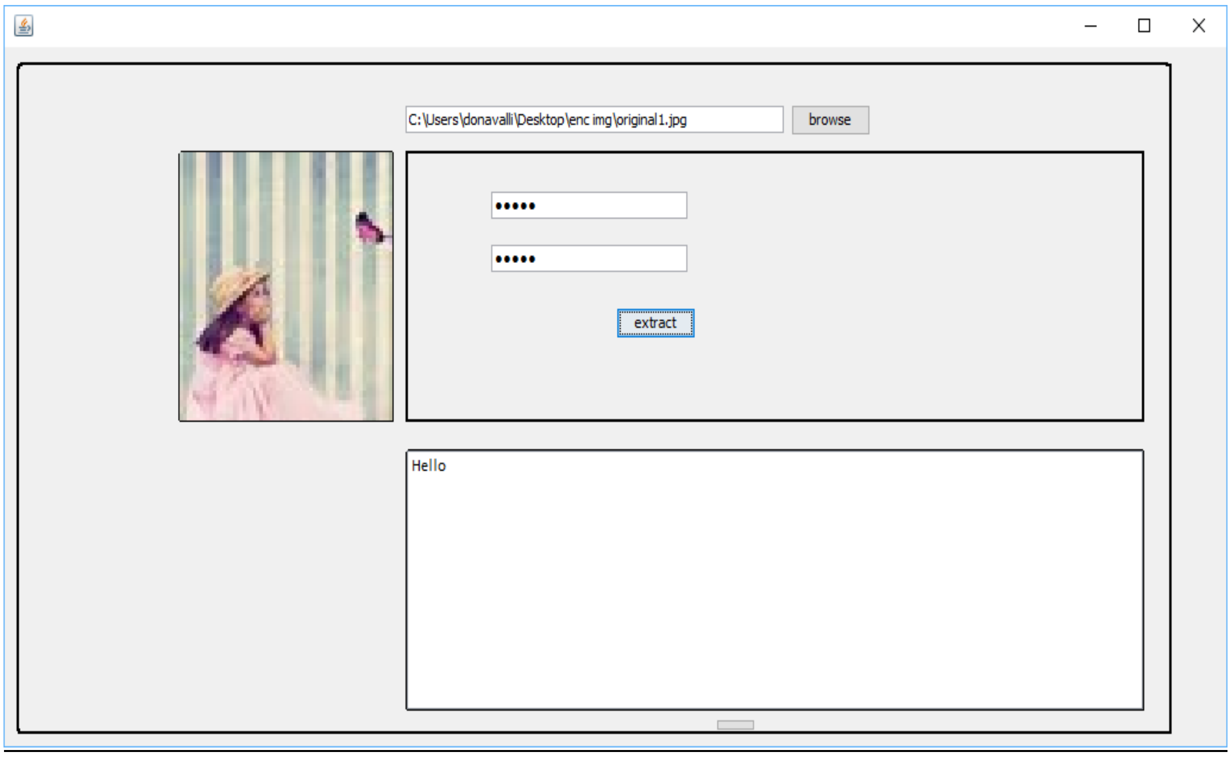
In **Fig-16,** when the button “**Get Original Image**” is clicked, the swing asks whether the user wants the preview of the image. When the user clicks “**OK**”, the preview of the original image is displayed at the top right corner of the swing window. Also, the original image after decryption is saved at a location specified in the code. This saved image is then extracted to retrieve the text message embedded by LSB algorithm.

8.5 Extraction:

Through the processes partitioning, zigzag reverse and reverse swapping, we have obtained the original image from the encrypted image. Initially through the concept of steganography and using technique of least significant Bit (LSB) algorithm, we have embedded some text which is hidden in the image. By retrieving this embedded or hidden text from the decrypted image, we complete every process as planned in the project execution. To achieve this, the receiver must know the key (user defined) which was used while embedding the text into the image for authentication. The reverse process of LSB is applied on the image i.e, least significant bits of the image bits are retrieved and combined to obtain the embedded text from the image.

Result:

Fig-17 shows the output screen of extraction process i.e, retrieval of embedded text from the image. As shown in the image the browse button is used to load the image which is was decrypted. Then the user enters the user defined key which was used while embedding the text in the image. Now, when the “Extract” button is clicked, the text which is hidden in the image is displayed below.


C:\Users\donavalli\Desktop\enc img\original1.jpg

•••••

•••••

Hello

Fig-17

10. Conclusion:

Image encryption is one of the oldest and efficient techniques which took data confidentiality to a whole new level. In the project we employ algorithms like steganography, partitioning, zigzag and swapping. Here, both the image and the text hidden using LSB algorithm are inaccessible to unauthorized parties. The algorithms used in the project are very simple to analyze but when these are implemented together in a sequence, the process of encryption is very solid and cannot be accessed by the unauthorized parties or intruders. Even when the encrypted image and the key is accessed by the intruders, They cannot access the information as they do not know the sequence of steps undergone during the process of encryption which makes them clueless regarding the decryption process. Though the encryption techniques used in the project aren't the most complex techniques of image encryption, they are used altogether in a sequence which results in flawless security.