Architectural Design of Multi Level Steganography System for Data Transmission

Adedayo Adeolu Adeniji, Micheal Esiefarienrhe, and Naison Gasale

***Abstract*—**Recent advances in steganography systems have shown good results in bringing reliable data security over the internet. However' these systems have their unique vulnerabilities since they are faced with a variety of threats from attackers. Effective steganography systems must be able to conceal hidden data completely as if there were nothing hidden on them. That is, they must be undetectable to steganalysis procedures from unauthorized individuals. This work proposes a secure multi-level steganography system to safeguard data transmission and sharing over the Internet wherein cryptographic and steganographic techniques are combined. It also proposes compressing the cipher-text file using the Lempel-Ziv algorithm to reduce its size so that it uses less space in the cover image after data embedding. In this work, the use of Discrete Cosine Transform and Inverse Discrete Cosine Transform for Image processing is explained, since they will be used in the process of image compression and decompression respectively.

***Keywords*—**Cryptography, Data Transmission, Internet Security, Steganography

I. Introduction

IT is imperative to have robust security measurements to safeguard the privacy and security of data . Cryptography is the study of information hiding and verification which includes the protocols, algorithms and strategies to securely and consistently prevent or delay unauthorized access to sensitive information and enable verifiability of every component in a communication [1]. Also Steganography is the process of concealing information in ways that prevent the detection of hidden messages [1], in other words, is a process of hiding a secret message within an ordinary message and extracting it at its destination. Cryptography techniques alter the structure of the text itself whereas steganography techniques hide the text behind some other digitally representative media, thus transmitting it unsuspectingly.

Although two techniques achieve certain security effects, they make the secret messages unreadable and unnatural. But this protection can be broken with enough computational power [2].

Therefore this paper's focus is on designing a secure multi level steganography application system to safeguard data from been stolen or modifying by the hackers, This new field of study in Information Technology known as Steganography is more dependable compare to Cryptography system because hacker can easily perceive that the information being sent on the channel has been encrypted and is not the plaintext. This can naturally raise the curiosity level of a malicious hacker to conduct cryptanalysis attacks on the ciphertext (that is, analyze the cipher-text through the encryption algorithms and decrypt the ciphertext completely or partially).

To secure one to one communication, steganography is an important technique [3], the principle is that the third party must not have a hint of what is being transferred, Steganography enables information transfer in a covert manner such that it does not draw the attention of the unintended recipients [4]. The cover image is used to carry the secret message, because the human eyes are insensitive to very minute changes to the colours and hence, intelligently embeds the data and transmits it to the receiver [4-6]. When received, the data is securely retrieved by extraction.

Steganography and Cryptography are both intended to protect information from unwanted parties. Both Steganography and Cryptography are excellent means by which to accomplish this but neither technology alone is perfect and both can be broken. It is for this reason that most experts would suggest using both to add multiple layers of security [4].

In this paper we will take an in-depth look at this steganography technology and design the robust and undetectability with high capacity of hidden data steganography application system .

II. Related Work on Steganography Techniques

(a) P2P Steganography Application

This is a steganography application for hiding information in any image file. The scope of the project is implementation of steganography tools for hiding information includes any type of information file and image files and the path where the user wants to save Image and extruded file. The application was developed with C# and .Net framework 3.5 [4].

Adedayo Adeolu Adeniji, Masters Student of Computer Science Department, North West University, Mafikeng, South Africa. (corresponding author's phone:+2348034059165/+27847327418 ;e-mail: nikedayo2000@gmail.com).

Prof. Micheal Esiefarienrhe, is with North West University, Mafikeng, South Africa. He is now with the Department of Computer, Mafikeng North West University, (e-mail: esiefabukohwo@gmail.co).

Naison Gasela is with the Computer Science Department, North West University Mafikeng, South Africa. (e-mail: naison.gasela@nwu.ac.za.).

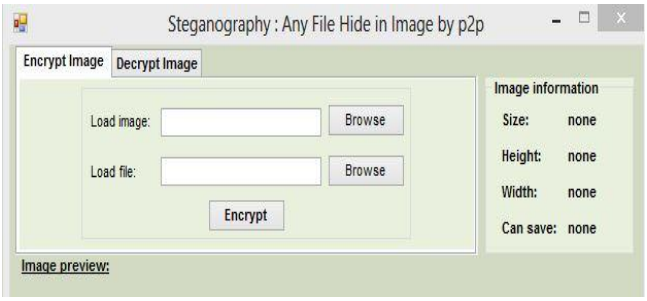

Fig. 1 Snapshot of P2P application

The algorithm used for Encryption and Decryption in the application was achieved using several layers lieu of using only LSB layer of image. Writing data starts from last layer (8st or LSB layer); because significant of this layer is least and every upper layer has doubled significant from its down layer. In order for a user to run the application, the user has two tab options – encrypt and decrypt. If user select encrypt, application give the screen to select image file, information file and option to save the image file. If user select decrypt, application gives the screen to select only image file and ask path where user want to save the secrete file [4].

(*b) QuickStego Steganography Application*

This is a steganography application that lets users hide text in pictures so that only other users of it can retrieve and read the hidden secret messages. Once text is hidden in an image the saved picture is still a 'picture', it will load just like any other image and appear as it did before. The image can be saved, emailed, uploaded to the web [7].

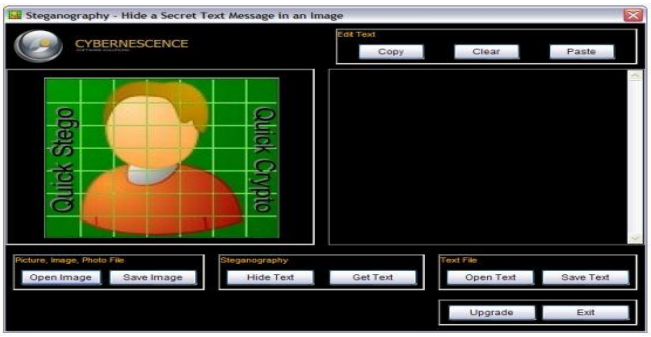

Fig. 2 Snapshot of QuickStego steganography application

The principle used in this application is hiding pure text under an image. The application enables a user to select secret text file or copy it content into textbox provided, after which the user will select the cover image. The text is hidden under the selected image and the stego image will be generated. The user can also save the resulting stego image [4].

III. Information Concealment In An Image

Coding secret messages in digital images is by far the most widely used of all methods in the digital world of steganography today. This is because it can take advantage of the limited power of the human visual system (HVS). Almost any plain text, cipher text, image and any other media that can be encoded into a bit stream can be hidden in a digital image.

Just before information concealment stage, the system requires a user to supply a message file which should be a text file, a secret key (this key is shared by both the sender and the receiver; the sender encrypt the message with this key and send it, whereas, the receiver is expected to supply the exact same key for him to extract the message from the image, decrypt and read the message), and the image file (which can be in JPEG format); the user may also specify if the message file is to be compressed in order to reduce its size and to hide more information in an image.

After all the inputs have been collected by the system, it goes to the message processing and encoding stage. In this stage, text message file is encrypted with the key supplied using the data encryption standard (DES) algorithm to generate the ciphertext, the ciphertext file may be compressed to shrink down a file so that it takes up less space in the cover image. In the message encoding stage, the message is encoded in order to improve the security of the system. Here, a constant is generated which is gotten from the system date (that is the Exclusive OR of day and month) as shown in eq. (3.35). The key and the message are converted to their bits equivalence respectively and arranged in group of eight bits (a byte), each group is encoded using the constant bits.

$$\text{Constant C} = (\text{Day bit}) \text{OR } (\text{Month Bit}) \quad (3.35)$$

A digital image is a rectangular grid of pixels or "picture elements", which are represented as arrays of values (bytes) in a computer's memory. These values represent the intensities of the three colors R (Red), G (Green) and B (Blue), where a value for each of three colors describes a pixel [8].

During the process of information concealment, the capacity of the JPG image file is calculated as shown in eq. (3.36) which determines if the image is large enough to hide the information.

$$Capacity = DCT - (DCT_1 + DCT_0) \quad (3.36)$$

where DCT is the number of DCT coefficients, DCT_0 is the of zero DCT coefficients and DCT_1 is the number of DCT coefficient with value 1

The process of message insertion starts with inserting the secret key bit in the image after which the message is inserted. The proposed approach scan through each pixels (each pixel has R, G, and B components) that make up the image, and the blue object of each pixel component selected is replaced with the message bit. The blue object of the first pixel component is used to hide the secret key bits and the blue object of the next pixel component is used to hide the constant bits which serve as the terminator. The essence of this approach is that, at the receiver end, secret key is first collected and compared with one stored in the image. If the keys match, the process proceeds to message retrieval and if it fails, it doesn't. After the terminator the message is hidden by following the same approach.

Each pixel that made up the cover image has red (R), green (G) and blue (B) components and each of these components

has 8-bits (since 24 bits per pixel image is being used). For instance, let's say the message to be hidden is grouped in 8 bits $10100101, 11001101, 10110101$ and so on. If the first three pixels of the cover image in RGB components are $(11001100\ 10110100\ 01100111)$, $(10110101\ 10001100\ 01010010)$, and (10010011 11001001 01001001). The proposed model replaces the blue object of each components with each group of the message bits and yields $(11001100\ 10110100\ 10100101)$, $(10110101\ 10001100\ 11001101)$, and $(10010011\ 11001001\ 10110101)$. Each of these replaced blue objects is encoded with the green object of the pixel components as illustrated in eq. (3.37) which make up the stego bits.

$$\text{stego bit} = (\text{Blue bit})\text{OR}\ (\text{Green Bit}) \quad (3.37)$$

Then the resulting stego pixels components will be $(11001100\ 10110100\ 00010001)$, $(10110101\ 10001100\ 01000001)$, and (10010011 11001001 01111100).

If the cover image is $image$ in the algorithm, $MsgFile$ represent the message in group of 8 bits and $Byte\_Len$ is the number of groups in the message. Algorithm 1 summarizes the procedure used in information concealment and algorithm 2 shows the procedure used in retrieving the information.

Start

```
Read Message, Key, Image;
Msg_Enc = Encrypt (Message, Key);
const = XOR(Day, Month);
c = GetByte(const);
Msg_Cmp = Compress (Msg_Enc); is Compressed = true;
IF (isCompressed == true)
   Msg = Msg_Cmp;
else
   Msg = Msg_Enc;
End If
Img_Capacity = Size(Image);
byte[] msgFile; i=0;
msgFile[i++] = c; // Terminator
while ((byteRead = Key.ReadByte() != -1)
{
   msgFile[i++] = byteRead XOR c;
}
msgFile[i++] = c;
while ((byteRead = Message.ReadByte() != -1)
{
   msgFile[i++] = byteRead;
}
msgFile[i++] = c;
Msg_Size = Size(msgFile);
If (Img_Capacity > Msg_Size)
   Img_Pxl[row, column] = ExtractPixel(Image);
   Byte_Len = MsgFile.Length;
   len = 0;
   for (i=0; i<Height; && (len < Byte_Len); i++)
   {
      for (j=0; j<Width; j++)
      {
         pixel = Image.GetPixel[j, i];
         r = pixel.Red;
         g = pixel.Green;
         b = pixel.Blue;
         msgBit = (byte) ((MsgFile[len] & (1 << j)) >> j);
         b = (byte)(msgBit == 1 ? b | (1 << 0) : b & ~(1 << 0));
         len++;
      }
   }
End If
```

End

Algorithm 1: Embedding algorithm

Start

```
Read stegoImage, Key;
kByte = ReadByte(Key);
cPixel = stegoImage.GetPixel[0, 0]; // Retrieve Terminator i.e constant bits
cb = cPicel.Blue;
bit = (byte) (( cb & (1 << 0)) >> 0);
b = (byte)(bit == 1 ? b | (1 << j) : b & ~(1 << j));
cbit = b;
int k = 0;
byte[] keyRetrieved, msgRetrieved;
for (i=0; i<Height; i++)
```

```
{
   for (j=0; j<Width; j++)
   {
      pixel = Image.GetPixel[j, i];
      r = pixel.Red;
      g = pixel.Green;
      b = pixel.Blue;
      bit = (byte) (( cb & (1 << 0)) >> 0);
      b = (byte)(bit == 1 ? b | (1 << j) : b & ~(1 << j));
      If (b == cBit)
         count += 1;
         isTerm = true;
      Else
         isTerm = false;
      End If
      If (count == 1 && isTerm == false)
         keyRetrieved[k++] = b;
      End If
      If (count == 2 && isTerm == true)
         k = 0;
         If (kByte != keyRetrieved)
            STOP; Display Error;
         End If
      Else If (count == 2 && isTerm == false)
         msgRetrieved[k++] = c XOR cBit;
   msgRetrieved[k++] = c XOR cBit;
      Else
         Break Loop;
      End If
   }
}
   msg = OutputMessage(msgRetrieved);
   If (msg is Compressed)
      Decompress;
   End If
   msg_Decr = Decrypt (msg, keyRetrieved);
End
```

Algorithm 2: Information retrieval algorithm

IV. Steganography Application

The Steganography application will allow the user to load an image into the system for steganography encoding, supply the message file, encrypt the file content using a specified key and compress the file if required. Once the user has supplied all this parameters, the stego-Encoder takes over during the process of sending it via the communication channel (internet).

Stego-Encoder and Stego-Decoder.

This is the section that encodes steganography process at the sender end and decodes the process at the destination end. Stego-encode as in figure 3.5 collects user information such as the message file, image file and the secret key to be used for the message encryption and perform the necessary action depending on the user requirement. The encoder encrypt the message file content using the key provided, then may compress the file before encoding the message under the original image using the key supplied to form stego-image and then send it via the communication medium to the destination.

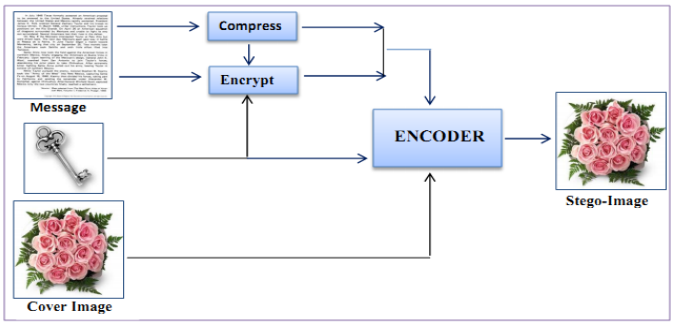

Fig. 3.5 The Steganography Encoder Architecture

The Stego-decoder as in figure 3.6 on the other hand, decodes the stego-image file using the same key as specified by the sender to get the message file, decompress the message file if need be and then decrypt it with the suitable key to get the message content.

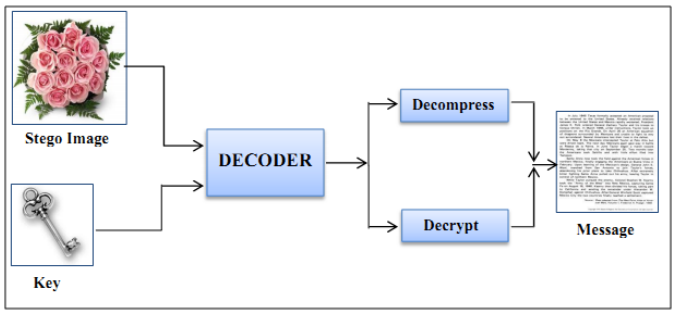

Fig. 3.6 The steganography decoder architecture

V. Data Modeling

The purpose of data design is to show how the individual parts of the system will interact with each other. After analyzing the various models in this section, this section details the use case diagrams and class diagrams required for the system.

The Use Case Diagram

The use case diagram is used to identify the primary elements and processes that form the system. The primary elements are termed as "actors" and the processes are called "use cases." The Use case diagram shows which actors interact with each use case [9]. A use case diagram captures the functional aspects of a system.

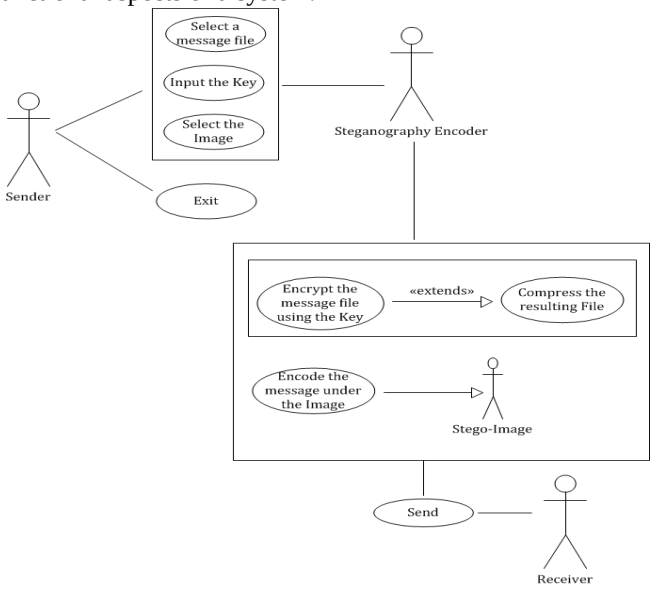

Fig. 3.7 Use case diagram representing the sender end of System

Fig 3.7 represents all actions taken by a user (sender) when he tries to send a hidden message to another user (receiver) using the steganography system. Within the steganography application, the user must specify a message file and enter an encryption key and an image and submit it to the steganography encoder for processing. The steganography encoder then encodes the message and embeds it in the image before being forwarded to the receiver.

The receiver as shown in figure 3.8 after receiving the stego image applies the key (which must be equivalent to the one used by the sender) to decode it in order to get the original message.

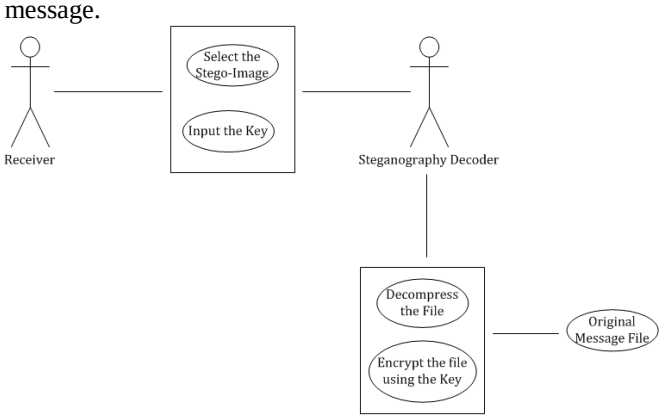

Fig. 3.8 Use case diagram representing the receiver end of the system

Class Diagram

A class diagram is a diagram showing a collection of classes and interfaces, along with the collaborations and relationships among classes and interfaces. [9]. A class diagram is a pictorial representation of the detailed system design. Figure 3.9 and Figure 3.10 illustrate the class diagram for the overall system.

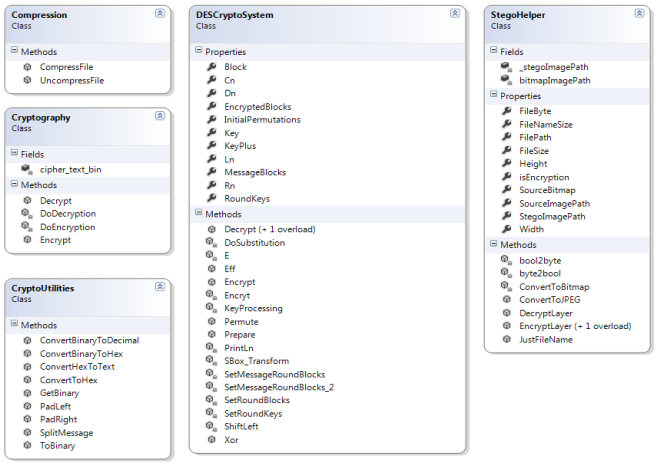

Fig. 3.9 Class diagram of the steganography system

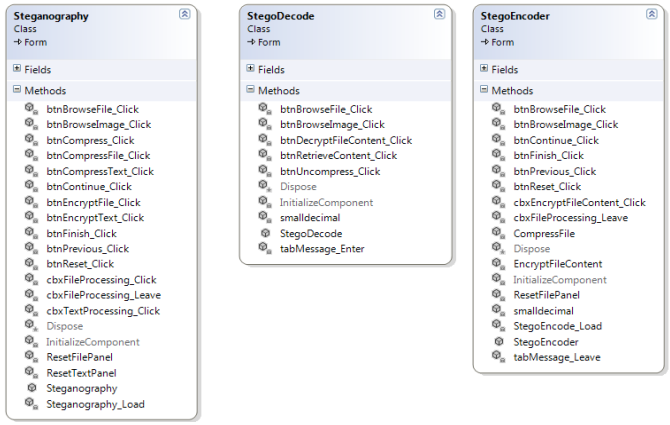

Fig. 3.10 Class diagram of the steganography system

VI. Quality of Reconstructed Image

Peak Signal-to-Noise Ratio (PSNR) is a term which measures the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. PSNR is usually expressed in terms of the logarithmic decibel scale.

PSNR is used to measure the quality of reconstruction of lossy compression codecs (for example, for image compression). The signal in this case is the original data, and the noise is the error introduced by compression. When comparing compression codec's, $PSNR$ is an approximation to human perception of reconstruction quality. Although a higher $PSNR$ generally indicates that the reconstruction is of higher quality, in some cases it may not. One has to be extremely careful with the range of validity of this metric; it is only conclusively valid when it is used to compare results from the same codec (or codec type) and same content.

PSNR is most easily defined via the mean squared error (MSE). Given a noise-free $m \times n$ image I and its noisy approximation K, MSE is defined as:

$$MSE = \frac{1}{m.n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \quad (3.38)$$

PNSR is defined as:

$$PNSR = 10.\log_{10} \frac{MAX_i^2}{MSE} \quad (3.39)$$

$$PNSR = 20.\log_{10} \frac{MAX_I}{\sqrt{MSE}} \quad (3.40)$$

MAX_I is the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, this is 255. For colour images with three RGB values per pixel, the definition of PSNR is the same except the MSE is the sum over all squared value differences divided by image size and by three. Alternately, for colour images the image is converted to a different colour space and PSNR is reported against each channel of that colour space.

VII. Conclusion

The quality of the reconstructed (stego) image is better compared to that of P2P and Quickstego applications, as emphasis is placed on the area of data security, privacy protection, it is believed that this research work will do exactly that by improving on the security and the hiding capacity of the existing technologies.

References

[1] T. Morkel, "An overview of Image Steganography," Information and Computer Security Architecture (ICSA) Research Group, 2005.
[2] M. Natarajan and N. Lopamudra, "Steganalysis Algorithms for Detecting the Hidden Information in Image, Audio and Video Cover Media," International Journal of Network Security & Its Application (IJNSA), , vol. 2, 2010.
[3] N. Meghanathan and L. Nayak, "Steganalysis algorithms for detecting the hidden information in image, audio and video cover media," international journal of Network Security & Its application (IJNSA), vol. 2, pp. 43-55, 2010.
[4] S. Changder, N. C. Debnath, and D. Ghosh, "A New approach to hindi text steganography by shifting matra," in Advances in Recent Technologies in Communication and Computing, 2009. ARTCom'09. International Conference on, 2009, pp. 199-202.
[5] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," IBM systems journal, vol. 35, pp. 313-336, 1996 http://dx.doi.org/10.1147/sj.353.0313.
[6] S. Singh and G. Agarwal, "Use of image to secure text message with the help of LSB replacement," International journal of applied engineering research, vol. 1, 2010.
[7] Cybernescence, "Hide text in images fast and easy," http://www.quickcrypto.com/free-steganography-software.html, 2010.
[8] K. Amanpreet, D. Renu, and S. Geeta, "A new Image Steganography Based on First Component Alteration Technique," International of Computer Science and Information Security, vol. 6, 2009.
[9] M. Chitnis, T. Pravin, and A. Lakshmi, "Creating use case diagram," http://www.developer.com/design/article.php/10925_2109801_4/Creating-Use-Case-Diagrams.htm, 2006.