CHAPTER 4

DESIGN OF IMAGE STEGANOGRAPHY ALGORITHMS WITH SECURE MESSAGE ROUTING FOR P2P NETWORKS

The present research proposes information hiding techniques by enhancing the operations of steganographic algorithms suitable for secure transmission over P2P networks. The study also proposes secure routing algorithm for safe delivery of the cover image. This chapter presents detailed description of the proposed secure message routing algorithm and steganographic algorithms. The algorithms and the enhancement procedures used are presented in this chapter.

4.1. PHASE I: DESIGN OF SECURE MESSAGE ROUTING ALGORITHM

Computers have become the integral part of day-to-day activities and computer networks are widely used to facilitate communication, sharing of data and resources. Computer network architecture refers to the layout of the network and consists of various components like hardware, software, connectivity, communication protocols and mode of transmission (wired or wireless). Two popular network architectures are client-server and Peer-to-Peer networks.

In the client-server model, two processes are involved, one on the client side and one on the server side. First, the client sends a message over the network to the server, the client then waits for a reply. When the server receives the request it performs the required work and sends back a reply.
In the P2P architecture, individuals form a loose group and can communicate with others in the group. It allows communication between two systems, in which each system is considered equal. Peer-to-peer networking is an alternative to the client-server model. Under the P2P model, each system is both a server and a client, commonly referred to as a servant. P2P networking has existed since the birth of computing networks. Recently, however, P2P networks have gained momentum with searchable peer-to-peer network file databases, increased network connectivity, and content popularity.

Peer-to-Peer networking is a serverless networking technology that allows several network devices to share resources and communicate directly with each other. Peer-to-Peer architecture is a commonly used computer networking architecture in which each workstation, or node, has the same capabilities and responsibilities. Peer-to-Peer architecture is often referred to as peer-to-peer network. Peer-to-Peer Infrastructure includes scalable and secure peer name resolution, efficient multipoint communication, distributed data management, secure peer identities and secure peer-to-peer groups.

Peer-to-Peer architecture distinguishes itself by its distribution of power and function. Rather than concentrating its power in the server, Peer-to-Peer models rely on the power and bandwidth of participants. They form *ad hoc* connections between nodes for sharing all kinds of information and files. As introduced in Chapter 1 (Introduction), a P2P network can be either structured or unstructured. This study proposes an enhanced message routing algorithm for structured P2P network.

Structured peer-to-peer networks generally holds the traits of using some type of algorithm for organization and optimization of the network. They use a global protocol to ensure that any node in the network can proficiently route a search to some peer that has the desired information. The most popular structured P2P network is the distributed hash table or DHT. DHT are a class
of decentralized distributed systems that provide a lookup service similar to that of a hash table. Pairs, key and value are stored in the distributed hash table and any connected node can retrieve the value associated with a given key. Maintaining the mapping from keys to values is the responsibility of each of the nodes present in the network. This is done in such a way that a change to the set of peer causes a negligible amount of disruption to the rest of the network. An example is shown in Figure 4.1.

![Distributed Hash Table](image)

**Figure 4.1: Distributed Hash Table**

One of the major advantages of P2P network is that each time a new node is connected to the network, in contrast to client/server network, the total capacity of the system increases. Another advantage is that there is no single point of failure in the network. Even if one node fails, the rest of the nodes are able to continue relaying information.

In spite of various advantages, one serious issue of current structured P2P networks is its security. For example, in the Sybil Attack (Douceur, 2002), a malicious node associates itself illegitimately with a large number of peer-IDs by impersonating other peers. It can behave as if it were a large number of peers, called Sybil ID peers, which can communicate with the legitimate peers. The sybil ID peers may drop, corrupt, misrouting messages or act as query’s destination. Castro (Castro *et al.*, 2002) and Wallach (2003)
summarize three categories of secure routing required in DHT-based P2P system, namely, Chord, Pastry and Tapestry for secure assignment node identifiers, secure routing maintenance and secure message routing.

Secure assignment node identifiers and secure routing maintenance can be achieved by minimizing the probability that nodes are controlled by attackers. However, an adversary can prevent correct message delivery throughout the overlay. Even a small fraction of malicious nodes can prevent correct message delivery throughout the overlay. Thus a need for secure routing that can identify malicious nodes is required. When one or more peers between initiator and target are malicious, a message might be dropped, polluted or forwarded to a wrong place. In this paper, we focus on the issue of secure message routing in structured P2P systems. The main idea behind the tracer message routing algorithm is to allow the sender of each query to identify malicious nodes in its routing path. On detecting an adversary, the sender asks the honest peer in the immediately previous hop to search an alternate routing path to bypass it. As a result, the routing latency consists of two parts: normal routing latency and extra routing latency incurred by bypassing malicious nodes.

P2P systems rely on other peers for message routing, thus each message should be properly forwarded to the next hop without any modification. Malicious nodes may attack the message routing in the following ways:

1. Alter the message in transit.
2. Disrupt the message routing, or take advantage of locality to control some routes.
3. Pretend to be the target.
In order to prevent attacks mentioned above, any message routing scheme should contain sufficient protection against attacks manipulating the overlay routing. First, the authenticity and integrity of messages can be handled by using signatures. A peer receiving a message can confirm that the message is not altered by verifying the signature. The signature of initiator allows the attack to be detected, the malicious peer to be identified. For this purpose a PIS (Peer-ID signature) scheme is used. Second, the routing path of a message can be controlled by the sender. On detecting that the message is forwarded to an undesirable node, the sender can search an alternate routing path used to get around malicious peer nodes. For this purpose a tracer routing algorithm that is combined with PIS is used.

4.1.1. Peer-Id Based Signature Scheme

As mentioned previously, in DHT-based P2P systems, the peer-ID is generated by hashing the IP address of the peer. Hashing a value as a mechanism to generate a peer-ID can be controlled by malicious nodes is inherently insecure. An adversary might obtain any desired peer-IDs by faking IP address. Thus P2P systems require a new mechanism to generate Peer-IDs. For this purpose, a centralized certificate authority (CA) is proposed, which is involved on the first time a peer wishes to join the overlay. A peer certificate asserts that the peer is authorized and has a unique Peer-ID. However, nodes require large amount of space to store the certificates and it is difficult to deploy such security protocol in P2P systems where churn is essential feature (Lua, 2007).

To address this issue, an ID Selector (IS) service and Private Key Generator (PKG) service is used. The Peer-ID is random number generated by IS and is not tied to IP address. By limiting the rate at which IDs are generated, the effectiveness of a Sybil attack can be reduced. Rate limiting mechanism can include a minimal charge or requiring a valid credit card
number. It makes obtaining a very large number required to corrupt a P2P system impossible. Each peer wants to generate a signature, obtains the private key from Private Key Generator (PKG) service. Its public key is a string that is derived from its peer identity or node geometric coordinates in the P2P system.

Suppose that peer \( x \) receives a signed message from peer \( y \), it verifies the signature using peer \( y \)'s identity as its public key. Here, peer \( x \) need not obtain peer \( y \)'s public key, since each forwarder can verify the received messages without contacting the initiator, PKG not only simplifies the key management process significantly but also enhances the efficiency of verifying. IS and PKG may be hosted at the bootstrap service layer of the P2P system. The private key is generated only once when the peer first joins the P2P system through the usual P2P operation of contacting the bootstrap nodes. The private key of a peer is valid as long as its public key is valid.

Additionally, this scheme offers a merit that all peers have the ability of determining the ID of the remoter peer. By verifying signature of the target, the initiator can confirm that the query result is delivered from the target, rather than a compromised peer along the routing path. In the proposed tracer algorithm, the peer-ID is selected by IS, while the ID of key content object is produced by hashing the key. Thus the proposed scheme can easily be implemented in DHT-based P2P systems.

### 4.1.2. Tracer Routing Algorithm

The routing scheme used by a P2P system is very important to both its performance and resistance to attacks. Inspite of several mechanisms being used, in each case, the initial message is sent from the sender to the peer in the routing table most likely to route correctly, as defined by the DHT algorithm
in use. Subsequently, the peer may provide further routing using one of the
two mechanisms, iterative and recursive (Shen et al., 2010).

In iterative routing, the querying node contacts each intermediate
node directly. If the contacted peer is not responsible for the target ID, then the
contacted peer issues a 302 redirect response pointing the search peer toward
the best match the contacted peer has for the target ID. The searching peer
than contacts the peer to which it has been redirected and the process iterates
until the destination peer is located.

In recursive routing, a lookup request is forwarded from node to
node until it reaches its destination. If the contacted peer is not responsible for
the target ID, it will forward the query to the nearest peer to the target that it
knows and the process repeats until the target is reached. The response
unwinds and follows the same path on the message return. Because distributed
Session Initiation Protocol (dSIP) uses SIP messages for transport, SIP's proxy
behaviour is used to enable recursive routing.

An example for recursive routing is presented in Figure 4.2, where
the initiator issues a query to the nearest peer to the target according to its
routing table. If the intermediate peer, say $x$, is not the target, $x$ will forward
the query to the peer in the next hop, without making any acknowledgement to
the initiator. The process repeats until query reaches the target. The target
sends the query result back to the initiator directly. Recursive routing enable
routing queries as quickly as possible, but the initiator has no control over the
routing process.
Figure 4.2: Example of Recursive Routing

In iterative routing (Figure 4.3), the initiator can know the whole routing process. In each step, the initiator asks intermediate peer $x$ to return the IP address of the next hop, instead of letting $x$ forward the query directly. With the returned IP address, the initiator sends the query to each peer of the routing path on its own, repeating until the query reaches the target.

The recursive routing is efficient, but is not secure, even combined with the technique of verifying the ID of remote peer such as Peer-ID based signature scheme. The fact that intermediate peers may assist in obfuscating the intermediate peer disrupts the message forwarding. For example, an adversary may do not forward queries that it receives, or forwards them to a wrong place. The initiator has no knowledge about the real cause. Iterative routing is not efficient, but it gains some benefits due to its manageable behavior. The intermediate peers reply with the IP address of the next hops to the initiator, the initiator can send the message to the peer in the next hop directly. Three kinds of attacks are considered.

Figure 4.3: Example of Iterative Routing
Scenario 1: The intermediate peer x pollutes or forges the content of the query. The next hop will still receive the original query since initiator sends the query by itself.

Scenario 2: Peer x drops the query. If no relay from the next hop is received in a given timeout, initiator will determine that x drops it.

Scenario 3: Peer x returns initiator an incorrect next hop. If the incorrect next hop colludes, the fact that initiator cannot verify the identity of the next hop makes determining which peer sends the incorrect reply impossible. Thus techniques that verify the ID of the remote peer are necessary. Combined with Peer-ID based signature scheme, initiator can identify the malicious node x.

To make the routing strategy perform best, a routing strategy called tracer routing is used (Xiang and Jin, 2009). An example is shown in Figure 4.4.

![Figure 4.4: Example of Tracer Routing](image)

The basic principle of the routing strategy is as follows. In each step, the intermediate peer x not only forwards the query to the next hop, but also returns the IP address of the next hop to initiator. With the additional information, the initiator has the knowledge about the whole routing process.
Each intermediate peer directly forwards the query to the next hop, thus the query can be routed quickly. Combined with the peer-ID based signature scheme, tracer routing offers a good tradeoff between routing efficiency and security.

In the proposed secure message routing scheme, the initiator appends a signature to a query. When an intermediate peer $x$ receives the message (including query and its signature), $x$ verifies the message and discards the polluted or forged one using the initiator’s public key. Recall that the public key is the Peer-ID of initiator. Then $x$ forwards the message it received to the next hop. At the same time, $x$ sends an acknowledgement (including the Peer-ID of the next hop, query and the signature generated using the private key of $x$) to initiator. The process is repeated until the query reaches the target.

4.2. VISUAL CRYPTOGRAPHY

The study uses Visual Cryptography (VC) in both Phase II and III steganography algorithms. The concepts of VC are presented in this section. VC is a Visual Secret Sharing Scheme (VSSS) introduced by Naor and Shamir (1995) for encrypting materials like written text, printed text, pictures, in a secure fashion. It uses the Human Visual System (HVS) to decrypt a secret message without expensive and complicated decoding process. A VSSS partitions an image encoded with a secret digital signal ‘S’ into a collection containing ‘n’ black and ‘m’ white pixels. Each collection of $m \times n$ pixels is referred to as a share, which resembles a noisy and scrambled image when viewed separately. During decoding phase, these shares or subset of shares are stacked together to allow the visual recovery of the secret message.

VC has been applied to many applications, including E-voting system (Paul et al., 2003), financial documents (Hawkes et al., 2000),
information hiding (Bonnis and Santis, 2004), general access structures (Ateniese et al., 1996), visual authentication and identification (Naor and Pinkas, 1997). More detailed information about visual cryptography can be found in (Yang, 2002). VC technique has many advantages as listed below.

- Simple to implement and low cost algorithm
- Encryption does not require any heavy computation
- No separate decryption algorithm is required (uses only Human Visual System) and thus, no prior knowledge on cryptography is needed during decrypt.
- Any share by itself or using a subset of shares does not provide enough information, but together, overlaid in correct order reveals S.
- Highly secure method, even with the help of infinite computation power, it is very difficult to predict S.

This research work uses the basic (2, 2)-threshold VSSS. The procedure (Figure 4.5) is explained below.

![Visual Cryptography Procedure](image)

**Figure 4.5: Visual Cryptography Procedure**
In this algorithm, each pixel of the binary secret image is expanded into 2 * 2 pixels (Figure 4.6). To share a white pixel of the secret image, one row from the first 6 rows is chosen randomly. Similarly, the two shares of a black pixel are determined by a random selection from the 6 last rows of Figure 4.6. As a result, an M*N pixels secret image is expanded into two 2M*2N pixels share-images.

Considering security of the method, presence of only one share image does not reveal the corresponding secret image, i.e., each 2*2 pixels block of one share-image may correspond to either a white pixel or a black pixel of the secret image. Stacking the shares of a black secret pixel results in 4 black subpixels, whereas only 2 black subpixels is gained by stacking shares of a white secret pixel (Houmansadr and Ghaemmaghami, 2005). So, secret image is revealed to human eyes by stacking the shares without performing any cryptographical computations.

One drawback of using (2, 2) VCS is that the complete recovery of white pixels is difficult and thus a loss in contrast in the reconstructed image appears. This is due to the OR operation used during reconstruction. Therefore, a XOR based VCS scheme is generally used, where the images are superimposed using XOR operation. This results in perfect reconstruction of both black and white pixels as shown in Figure 4.7.
<table>
<thead>
<tr>
<th>Secret Pixel</th>
<th>Share 1</th>
<th>Share 2</th>
<th>Stacked</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image1" alt="Share 1" /></td>
<td><img src="image2" alt="Share 2" /></td>
<td><img src="image3" alt="Stacked" /></td>
</tr>
<tr>
<td>White</td>
<td><img src="image4" alt="White Share 1" /></td>
<td><img src="image5" alt="White Share 2" /></td>
<td><img src="image6" alt="White Stacked" /></td>
</tr>
<tr>
<td>Black</td>
<td><img src="image7" alt="Black Share 1" /></td>
<td><img src="image8" alt="Black Share 2" /></td>
<td><img src="image9" alt="Black Stacked" /></td>
</tr>
</tbody>
</table>

Figure 4.6: (2, 2) VC Scheme with 2x2 Subpixels
Phase II of the study, as mentioned in the previous chapter (Chapter 3, Methodology), enhances the traditional LSB algorithm and combines with four different variants of DCT. The variants are:

(i) 1- DCT
(ii) 2- DCT
(iii) 3-DCT
(iv) 4-DCT

Thus, five steganographic algorithms are proposed in Phase II. They are,

(i) Improved LSB (L2LSB)
(ii) Coupled L2LSB + 1DCT
(iii) Coupled L2LSB + 2DCT
(iv) Coupled L2LSB + 3DCT
(v) Coupled L2LSB + 4DCT

All the algorithms use a scrambling procedure which first converts the original secret message into a form that is unrecognizable to a hacker. The
main goal of the scrambling algorithm is to deliver a message that is not similar to the original message. Detailed description of each of these models is presented in the following paragraphs.

4.3.1. The L2LSB Algorithm

The design of proposed steganography algorithms consist of three major steps as listed below.

(i) Prepare the secret digital information
(ii) Prepare the cover image
(iii) Embed / Extract the secret data

(i) Preparation of Secret Digital Information

While creating stego images, certain preprocess steps are undertaken to ensure security. The secret digital data undergoes a transformation process which makes it different from the original signal. This process is discussed in this section.

First, the secret message/copyright data is first encrypted using a simple encryption method. The encrypted data is then binarized. In the next step, using Visual Cryptography, two shares $S_1$ and $S_2$, are created in such a way that all odd positioned bits are in share 1 and even positioned bits are in Share 2. Share 1 and Share 2 are then consolidated using a simple compression algorithm like Run Length Encoding. The process is pictorially given in Figure 4.8 and the encryption algorithm is shown in Figure 4.9.

The result of this step is an image which does not resemble the original secret message. An example of Lena image after applying the above scrambling algorithm is shown in Figure 4.10. To show that the similarity between original and scrambled images a wide histogram of the two images were constructed and the result is shown in Figure 4.11.
Procedure BitEncDec(SecMsg, ArrInt)
1. Repeat till Step – for each character/pixel in SecMsg
2. Get Ascii value (intAscVal) of the character/pixel
3. Perform encryption
   (intAscVal = intAscii Or (2 ^ bit))
   or decryption
   (intAscVal = intAscii Xor (2 ^ bit))
4. Set Result as Result + intAscVal

Figure 4.8: Secret Message Image Preparation

Figure 4.9: Scrambling Algorithm

Figure 4.10: Original and Scrambled Image
Figure 4.11: Histogram of the Original and Encrypted Lena image

To ensure that the descrambling algorithm (reverse of scrambling produces the original images) successfully brings back the secret image, the NC value between the original and descrambled image was calculated. While considering Lena image, the NC was 1, which shows that the original image was obtained without any loss. Similar result was obtained with other images also.

The descrambling algorithm for secret message file was analyzed using error rate in bits per pixel (bpp). Results for three sample text files are presented in Table 4.1.

The lower error rates indicate that the scrambling algorithm is successful in reproducing the original secret message file.

TABLE 4.1

SECRET MESSAGE EMBEDDING RESULT

<table>
<thead>
<tr>
<th>Secret File Size</th>
<th>ER (bpp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4985</td>
<td>0.062</td>
</tr>
<tr>
<td>4805</td>
<td>0.056</td>
</tr>
<tr>
<td>4999</td>
<td>0.041</td>
</tr>
</tbody>
</table>
(ii) **Prepare the cover image**

Initially, the cover image is divided into two shares, Share 1 and Share 2, share 2 alone is used to embed the secret data. During embedding procedure, Share 2 image is partitioned into two bit strings, namely, Most Significant Bits (MSB) and Least Significant Bits (LSB). In traditional LSB algorithm, one LSB of each color is chosen to embed the secret data. This has the advantage of introducing least visual distortion to the image. But statistical analysis reveals a major color change.

For example, consider a cover color pixel A8A8A8 with value 10101000-10101000-10101000 and a secret message 111. After embedding secret message into cover pixel, the result would be 10101001-10101001-10101001. After secret data insertion, the color pixel value changes to A9A9A9. In theory, since only 3 LSBs are changed negligible distortion takes place. However, considering the color representation of the original pixel stream (decimal value 11053224) with embedding bit stream (11119017), it could be seen that there is a vast variation in the scale of colors (65793).

This problem can be solved if only one color channel is used. For example, if 3 LSBs of the green color channel is used, then the embedded bit stream would be 10101000-10101111-10101000. The equivalent pixel value is changed to A8AFA8 (decimal value 11055916). Now the color variation is only 1792. If Blue channel is used this difference would be only 7. But the problem now is to decide which color channel will best suit a situation. This problem is solved by using a simple Restrict Pixel Procedure (RPP). The steps are detailed in Figure 4.12.

The result of the above procedure will be a pixel pair from a color channel. This pair will exhibit high distortion and therefore the result of the withholding of information will be less detectable.
(iii) **Embed / Extract the Secret Data**

This section describes the enhancement operation made to LSB first followed by the embedding and extraction procedure.

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Consider each color channel (R, G, B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 2</td>
<td>Repeat steps 2a-2c for each pair of pixels</td>
</tr>
<tr>
<td>Step 2a: Check for closeness property.</td>
<td>A pair of pixel is said to be close,</td>
</tr>
<tr>
<td>If</td>
<td></td>
</tr>
<tr>
<td>(</td>
<td>(R_1 \text{ or } G_1 \text{ or } B_1) - (R_2 \text{ or } G_2 \text{ or } B_2)</td>
</tr>
<tr>
<td>(</td>
<td>(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2</td>
</tr>
<tr>
<td>C = C + 1</td>
<td></td>
</tr>
<tr>
<td>Step 2b: Check for uniqueness property.</td>
<td>A pair of pixel is said to be unique</td>
</tr>
<tr>
<td>If</td>
<td></td>
</tr>
<tr>
<td>(</td>
<td>R_1 - R_2</td>
</tr>
<tr>
<td>U = U + 1</td>
<td></td>
</tr>
<tr>
<td>Step 2c: Calculate Ratio R = C/U</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>Select pixels which has high ratio, because it represents more diversity, leading to less noticeable changes and return them</td>
</tr>
</tbody>
</table>

**Figure 4.12: Restrict Pixel Procedure**

- **Enhanced Least Significant Bit Algorithm (L2LSB)**

One of the earliest digital image steganography schemes is a simple technique that embeds information in Least Significant Bit (LSB). Let G be an image with \( m \times n \) pixels, where each pixel is represented as an 8-bit sequence, the technique embeds the secret data in the last bit (i.e., least significant bit) of the selected pixels of the cover image. To the human eye, the resulting image will look identical to the cover object (Johnson and Jajodia, 1998b).
For example, consider an image X with 9 bytes in which letter ‘A’ (with binary value 10000001) has to be hidden. Let the original raster data consist of the following bit pattern.

\[(\text{00100111 11101001 11001000})\]
\[(\text{00100111 11001000 11101001})\]
\[(\text{11001000 00100111 11101001})\]

Inserting the binary value for A into the above bit pattern results with the following modified bit stream.

\[(\text{00100111 11101000 11001000})\]
\[(\text{00100110 11001000 11101000})\]
\[(\text{11001000 00100111 11101001})\]

Careful scrutiny reveals that the bit values are changed only in three positions (highlighted bits). On average, LSB requires only half the bits in an image be changed. The above example uses only one of the LSB position, but can be modified to use the last two bits also. These changes will still be undetected to human eyes and results with more embedded bits. An example is shown in Figure 4.13.

![Figure 4.13: Cover image X and Stegano Image X' with hidden Letter ‘A’](image.png)

The general methodology used by LSB scheme to store secret message is shown in this flowchart (Figure 4.14).
The traditional LSB technique is a simple method that is easy to implement and does not generate serious distortion to the image. However, it also has some serious flaws. It is not very robust against attacks. For instance, an attacker could simply randomize all LSBs, which effectively destroys the hidden information.

- The technique can embed only fixed length secret messages
- During embedding, the technique uses all three color channels to embed bits (one bit from R, G and B).
- Even though HVS does not show variation, statistical analysis show difference and hence are less secure.

To solve the above problems, the traditional LSB technique is enhanced in the present study. The traditional LSB-based technique is enhanced in five manners.

1. Scrambling algorithm (Section 4.3.1) is used to modify the original message, which is compressed before embedding to avoid the problem of fixed length and to increase the security of the secret message file
2. Inclusion of visual cryptography to increase security during attacks
3. A novel Restrict Pixel Procedure (RPP) is added to determine which color channel is used to embed the secret message.

4. Finally a pixel enhancement procedure is used as a post processing step to increase quality of the stegano image.

5. Further, the LSB technique is enhanced to use three pixels instead of one (Roque and Minguet, 2009).

**Embedding and Extraction Tasks**

The secret message is prepared according to the procedure presented in Section 4.1. The embedding procedure uses 3 LSBs of the selected color band and replaces it with 3 bits of secret message. An example is shown in Figure 4.15. As a post processing step, a LSB match adaptation algorithm is used to further reduce the difference between the original pixel and the embedded pixel. This method calculates the distance between the original color and the embedded color. If this distance is greater than 3, then the color is decremented to get a final color closest to the original. This further reduces distortion in color caused by the hidden information.

**Figure 4.15: Example of 3-bit Bit Replacement LSB Algorithm**

As a simple example, consider cover bit stream 11001000 and secret 3-bit stream 111. The enhanced LSB will result the embedded byte as 11001111, having a difference of seven when compared with the original
signal. Since the difference is greater than 3, its immediate neighbor (fourth LSB) is decremented by 1, giving the final byte as 1100111. This results with a distance 1 from original byte and has the same hidden information. The procedure is consolidated pictorially in Figure 4.16.

A reverse of the embedding process is used to extract the original secret data. Several experiments were conducted to understand the efficiency gain obtained by the proposed enhanced LSB technique and are presented in Chapter 5, Results and Discussion.

**Figure 4.16: Enhanced LSB-Based Technique**

### 4.3.2. Coupled L2LSB-EDCT Algorithms

The coupled L2LSB-EDCT algorithm embeds secret data into cover image using a series of steps. First the cover and secret images are
prepared using procedure mentioned in Section 4.1. Huffman Encoding is performed over the secret image/message before embedding and each bit of Huffman code of secret image/message is embedded inside the cover image by altering the least significant bit (LSB) of each of the pixel's intensities of cover image. The size of the Huffman encoded bit stream and Huffman Table are also embedded inside the cover image, so that the Stego-Image becomes standalone information to the receiver. In the next step, DCT coefficients of the cover image are obtained. The study analyzes the use of four types of DCT, namely, 1D DCT, 2D DCT, 3D DCT and 4D DCT. All the four types of DCT divide the image into 8 x 8 blocks, from which the DCT coefficients are estimated. Finally inverse DCT is performed to obtain the stegano image. The steps of the proposed L2LSB models are shown in Figure 4.17.

![Figure 4.17: The Coupled L2LSB-DCT Algorithm](image-url)
To avoid visual distortion, embedding of secret information is avoided for DCT coefficient value 0. If DCT coefficient value is below threshold (T) then replace LSB(s) with pixels in Secret Image. That is, when T < 0, replace LSB(s) with pixels in Secret Image. Here threshold value is taken as zero and hence the pixels with DCT coefficient value below zero are used for data hiding.

(i) **Preparation of Cover Image**

The heart of the coupled L2LSB-DCT algorithms is the Discrete Cosine Transformation (DCT) translates the image information from spatial domain to frequency domain to be represented in a more compact form. Its stochastic properties are similar to Fourier transform and consider the input image to be a time invariant or stationary signal. The discrete cosine transform (DCT) is a special case of Discrete Fourier transformation (DFT) in which the sine components have been eliminated leaving only the cosine terms.

A Discrete Cosine Transform (DCT) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering, from lossy compression of audio (e.g. MP3) and images (e.g. JPEG) (where small high-frequency components can be discarded), to spectral methods for the numerical solution of partial differential equations. The use of cosine rather than sine functions is critical in these applications.

In particular, a DCT is a Fourier-related transform similar to the Discrete Fourier Transform (DFT), but using only real numbers. DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), where in some variants the input and/or output data are shifted by
half a sample. There are eight standard DCT variants, of which four are common.

The most common variant of discrete cosine transform is the type-II DCT, which is often called simply “the DCT”, its inverse, the type-III DCT, is correspondingly often called simply "the inverse DCT" or "the IDCT". Two related transforms are the Discrete Sine Transform (DST), which is equivalent to a DFT of real and odd functions, and the modified discrete cosine transform (MDCT), which is based on a DCT of overlapping data.

The discrete cosine transform (DCT) is an important transform in 2D signal processing. It is known to be close to optimal in terms of its energy compaction capabilities and can be computed via a fast algorithm. DCT coefficients are used for JPEG compression. The cover image is split into 8*8 blocks and each block is used to encode one message bit. It separates the image into parts of differing importance. It transforms a signal or image from the spatial domain to the frequency domain. It can separate the image into high, middle and low frequency components. Details regarding the four DCT types are presented below.

The general equation for the 1-DCT (N data items) is defined by the following equation:

$$C(u) = \alpha(u) \sum_{t=0}^{N-1} f(x) \cos \left( \frac{(2x + 1)u\pi}{2N} \right)$$

(4.1)

for u=0, 1, 2... N-1 and N=7. The general equation for a 2-DCT (N by M image) is defined by the following equation:

$$C(u) = \alpha(u)\alpha(v) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(x,y) \cos \left( \frac{(2x + 1)u\pi}{2N} \right) \cos \left( \frac{(2y + 1)v\pi}{2N} \right)$$

(4.2)
The third variant of the DCT is a composition of three DCTs. The general equation for a 3-DCT is defined by the following equation:

\[
C(u, v, w) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} C(i, j, k) X_{ui} X_{mj} X_{nk}
\]

(4.3)

where

\[
X_{ui} X_{mj} X_{nk} = \cos \left( \frac{\pi}{N} \left( i + \frac{1}{2} \right) l \right) \cos \left( \frac{\pi}{N} \left( j + \frac{1}{2} \right) m \right) \cos \left( \frac{\pi}{N} \left( k + \frac{1}{2} \right) n \right)
\]

The general equation for a 4-DCT is defined by the following equation:

\[
C(u, v, w, r) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} C(i, j, k, n) X_{ui} X_{mj} X_{nk} X_{op}
\]

(4.4)

where

\[
X_{ui} X_{mj} X_{nk} X_{op} = \cos \left( \frac{\pi}{N} \left( i + \frac{1}{2} \right) l \right) \cos \left( \frac{\pi}{N} \left( j + \frac{1}{2} \right) m \right) \cos \left( \frac{\pi}{N} \left( k + \frac{1}{2} \right) n \right) \cos \left( \frac{\pi}{N} \left( o + \frac{1}{2} \right) p \right)
\]

(ii) **Preparation of Secret Data**

Before embedding the secret image into cover image, it is first encoded using Huffman coding (Gonzalez and Woods, 2006; Jayaraman et al., 2009). Huffman codes are optimal codes that map one symbol to one code word. For an image, Huffman coding assigns a binary code to each intensity value of the image and a 2-D M × N image is converted to a 1-D bits stream
with length $L_{H} < M \times N$. Huffman table (HT) contains binary codes to each
intensity value. Huffman table must be same in both the encoder and the
decoder. Thus the Huffman table must be sent to the decoder along with the
compressed image data. Huffman code $H$ is decomposed into 8-bits blocks $B$.
Let the length of Huffman encoded bits stream be $L_{H}$. Thus, if $L_{H}$ is not
divisible by 8, then last block contains $r = LH \% 8$ number of bits (where \% is
the modulo operator).

(iii) Embedding and Extraction Procedures

The embedding of the secret message into the cover image consists
of the following steps. In the first step, the cover image is divided into non-
overlapping blocks of size 8 x 8. Any one of the select DCT (1DCT or 2DCT
or 3DCT or 4DCT) variants is applied on each of the blocks to obtain
 corresponding DCT coefficients. The next step performs Huffman encoding
on the secret data and the result is converted into 1D bit stream $H$. The
Huffman code $H$ is then decomposed into 8 bit blocks $B$. The least significant
bit of all DCT coefficients inside 8 x 8 DCT blocks is replaced with a bit from
the 8 bit block $B$ in a left to right fashion. The method used for this purpose is
given below.

\[
\text{for } k=1 \text{ to } 7 \\
\quad \text{if } CD < 0 \text{ then} \\
\quad \quad \text{LSB}(CD) \leftarrow B(k) \\
\quad \text{end} \\
\text{end}
\]

where $B(k)$ is the $k^{th}$ bit from left to right of a block $B$ and $(CD)$ is the DCT
coefficient in binary form. Finally, an inverse DCT is performed on all blocks
to a new image $f1$ which contains secret image. The steps are summarized in
Figure 4.18.
Input: An M1×N1 carrier image and a secret message/image.

Output: A stego-image.

1. Obtain Huffman table of secret message/image.
2. Find the Huffman encoded binary bit stream of secret-image by applying Huffman encoding technique using Huffman table obtained in step 1.
3. Calculate size of encoded bit stream in bits.
4. Divide the carrier image into non overlapping blocks of size 8×8 and apply DCT on each of the blocks of the cover image.
5. Repeat for each bit obtained in step 3
   (a) Insert the bits into LSB position of each DCT coefficient of 1st 8×8 block found in step 4.
6. Decompose the encoded bit stream of secret message/image obtained in step 2 into 1-D blocks of size 8 bits.
7. Repeat for each 8-bit blocks obtained in step 6
   (a) Change the LSB of each DCT coefficient of each 8×8 block(excluding the first) found in step 4 to a bit taken from left(LSB) to right(MSB) from each 8 bit block B.
8. Repeat for each bit of the Huffman table
   (a) Insert the bits into LSB position of each DCT coefficient
9. Apply inverse DCT using identical block size.
10. End.

Figure 4.18: Embedding Procedure of L2LSB-EDCT Algorithm

During extraction, the stego-image is received in spatial domain and outputs the secret data. DCT is applied on the stego-image using the same block of size 8 × 8 to transform the stego-image from spatial domain to frequency domain. The size of the encoded bit stream is extracted from 1st 8 × 8 DCT block by collecting the least significant bits of all of the DCT coefficients inside the 1st 8×8 block. The least significant bits of all of the DCT coefficients inside 8×8 block (excluding the first) are collected and added to a 1-D array. The next step constructs the Huffman table of the secret
message/image by extracting the LSB of all of the DCT coefficients inside 8×8 blocks excluding first block. Perform decoding of the 1-D array using the Huffman table. The steps are summarized in Figure 4.19.

Figure 4.19: Extraction Procedure of L2LSB-EDCT Algorithm
4.4. PHASE III: DESIGN OF FEATURE BASED AND TRANSFORMATION BASED STEGANOGRAPHIC ALGORITHMS

Phase III of the study proposes two algorithms, namely, VCBPM and VCWS that enhances KP-Gilles based algorithm and Wavelet based algorithm.

4.4.1. Enhanced KP-Gilles Feature-Based Method (VCBPM)

Salient points or interest points of an image include corners, edges of an object, etc. Salient or interest point detection is a task that has been tackled by many researchers and existing techniques include corner detection algorithms like Hessian-Corner, Susan-Corner and LoG. These techniques are sensitive to the noise present in the image and thus during embedding often fails to extract features efficiently. If the detection of some feature points used during embedding fails, it results in small triangles and thus degrades image quality.

Moreover corner detectors tend to be influenced by the change of position of pixels. So, new ways for defining salient points in image hiding is gaining more interest. In this research a method that combines two feature detecting algorithms, namely, Hessian-Affine algorithm and a variant of the traditional Gilley’s Key detecting algorithm, namely, KP Gilles, is proposed. This section begins with a brief explanation of the Hessian-Affine and KP Gilley’s key detecting algorithms followed by the proposed algorithm.

- **Hessian-Affine Feature Detector**

  A scale and affine invariant interest point detector, called Hessian-Affine detector, is proposed by Mikolajczyk and Schmid (2004). The detector is adopted to obtain the feature points and the characteristic regions of an
image in our proposed algorithm. The procedure of the Hessian-Affine detector is as follows and is diagrammatically presented in Figure 4.20.

1. Detect initial points with Hessian detector and select the characteristic scale.
2. Estimate the shape with the second moment matrix.
3. Normalize the ellipse region to circular one.
4. Refine the point location and scale.
5. Go to step 2 if the second moment matrix of new point is not isotropic.

In first step, the Hessian detector is based on the Hessian matrix \( H \).

\[
H(x, \sigma_D) = \begin{bmatrix}
L_{xx}(x, \sigma_D) & L_{xy}(x, \sigma_D) \\
L_{xy}(x, \sigma_D) & L_{yy}(x, \sigma_D)
\end{bmatrix}
\]

where \( x \) is the coordinate of a point \((x, y)\), \( \sigma_D \) is the differentiation scale of the Gaussian kernel, and \( L_{ij} \) is the second derivative of a point with respect to \( i \) and \( j \) variables.

![Hessian-Affine Detector Process Diagram](image)

**Figure 4.20: Hessian-Affine Detector Process**

A point is regarded as a feature point if the second derivative test discriminant of the point is a local maximum. Moreover, a scale selection function, called Laplacian of Gaussian (LoG), in Equation 4.6 is applied in order to deal with scale changes.
\[ |\text{LoG}(x, \sigma_n)| = \sigma_n^2 (L_{xx}(x, \sigma_n) + L_{yy}(x, \sigma_n)) \]  

(4.6)

where \( \sigma_n \) indicates the Gaussian scale factor at scale \( n \). The operator responses are computed for a set of scales \( \sigma_n \). The response attains an extreme when the size of the LoG kernel matches with the size of a blob-like structure. In step 2, the elliptical region of the feature point is obtained by using the eigenvalues of the second moment matrix, \( \mu \), of the point.

\[
\mu(x, \sigma_1, \sigma_D) = \sigma_D^2 G(x, \sigma_1) \times \begin{bmatrix}
L_x^2(x, \sigma_D) & L_x L_y(x, \sigma_D) \\
L_x L_y(x, \sigma_D) & L_y^2(x, \sigma_D)
\end{bmatrix}
\]  

(4.7)

where \( \sigma_1 \) is the integration scale.

In step 3, the obtained elliptical region of the feature point is transformed into circular one according to the square root of the second moment matrix of the point. In step 4, a new feature point is obtained when the obtained circular region is also applied to the Hessian detector with a new differential scale. Finally, the procedure would be repeated if the eigen values of the second moment matrix of the new feature point is not equal to that of the original one.

- **Gilles’ Key Detector**

The saliency-based algorithm developed by Kadir and Brady (2001) based on Gilles (1998) is described in this section. Gilles redefined saliency in terms of local signal complexity and proposed the use of Shannon Entropy of local attributes to estimate the saliency. Kadir and Brady modified Gilles’s original algorithm in two aspects. First, the modified algorithm can automatically detect salient regions at multiple scales by looking for self-similarity. Second, the modified algorithm can achieve more stable and salient features by weighting the local entropy values at the peaks. This algorithm is
referred to as KP-Gilles algorithm in this dissertation and is summarized as follows: For each pixel location \((x, y)\) in the image \(I\):

1) For each scale \(s\) between \(s_{\text{min}}\) and \(s_{\text{max}}\):

   - Measure the local descriptor (e.g., using histograms) values within a circular region of radius (scale) \(s\).
   - Compute the PDF \(p_D(s, x)\), which is the probability density function of scale \(s\) from the local descriptor.
   - Calculate the local entropy \(H_D(s, x)\) using Equation (4.8)
     \[
     H_D(s, x) = \int_{i \in D} p_D(s, x) \log_2 p_D(s, x) \, \text{d}i
     \]  
     (4.8)

2) Select scales where the entropy is peaked as candidate scales.

3) The entropy values are weighted by \(W_D(s, x)\), which is the sum of absolute difference of the PDF of the local descriptor around candidate scales and is calculated using Equation (4.9).

   \[
   W_D(s, x) = s \int_{i \in D} \left| \frac{\partial}{\partial s} p_D(s, x) \right| \, \text{d}i
   \]  
   (4.9)

4) Calculate the saliency metric \(y_D(s, x)\) using Equation (4.10).

   \[
   y_D(s, x) = H_D(s, x) \times W_D(s, x)
   \]  
   (4.10)

The values of entropy \(H_D(s, x)\) are weighted by the sum of absolute difference of the histograms at the peak. The addition of an inter-scale saliency measure can correctly capture the most salient features and scales, i.e., a feature derived from the center point and scale would produce fewer false matches than one derived from the edge point. Finally, the \(N\) regions with
highest saliency over the image provide the features for learning and recognition. Each feature is defined by its center and radius (the scale).

- **Proposed Enhanced KP Gilles Algorithm**

  The Gilles and KP-Gilles methods pick single salient points in entropy space to represent salient features. Point-wise operations are slow and moreover, it is unlikely that features exist entirely within the local region and hence several neighboring positions are likely to be equally salient. So, the proposed method instead of operating on single point, works with a region or block of points. The proposed algorithm differs from the KP-Gilles in manners.

  - Choose salient regions rather than points.
  - Original algorithm uses high salient features to select high entropy regions.

    The proposed algorithm inverses the process and selects low entropy regions.

    To identify salient regions, the common solution would be to use some clustering algorithm like K-means. But such algorithms are highly parametric and require a priori definition of K (number of clusters). Moreover, these algorithms are sensitive to outliers (noise) in an image. Ideally, an algorithm for this situation should analyze the whole saliency space such that each salient feature is represented. The best approach is to use a thresholding method. Using a global threshold approach would result in highly salient features in one part of the image dominating the rest, while a local threshold approach would require the setting of another scale parameter.

    To solve the above requirements, a simple procedure is used. The algorithm selects high salient points by considering its local support. That is, it selects all neighbors with similar saliency and scale while making sure that the
grouped neighbors are sufficiently distant from other groups. This will make sure that each group has separate quality. For robustness, a representation that includes all of the points in a selected region is used. The proposed feature detection algorithm consists of the steps in Figure 4.21.

1) Apply a global threshold (T1).
2) Choose the lowest salient point in saliency-space (Y).
3) Find the K nearest neighbors (K = 8 used in experiments).
4) Test the support of these neighbors using variance of the centre points.
5) Find distance, D, from salient regions already clustered.
6) Accept, if D > scale_{mean} of the region and if sufficiently clustered (variance is less than T2).
7) Store as the mean scale and spatial location of K points.
8) Repeat from step 2 with next lowest salient point.

**Figure 4.21: Steps in Enhanced KP-Gilles**

The purpose of step 1 is to reduce computation time by removing the high salient features. This should be set quite high so as to keep all the features of interest and experimentation showed that a 80% threshold (of the least salient feature value) works well. Selection of T2 affects the robustness of the algorithm and for correct region separation the variation between the pixels in the region should always be less than this threshold. This threshold was set to 5 and was set after several experiments.

- **Embedding and Extraction Process**

    After the feature detection process, the secret data is embedded into the cover image using the selected features. The secret data is prepared according to the techniques presented in Section 3.3.1. The general procedure is shown in Figure 4.22.
At first, the feature points of an image are obtained by using the detector algorithms explained in the previous section. Many feature points and characteristic regions are obtained after feature detection. However, some of the feature points are useless and redundant because of the weakness of a region and the overlap between the regions. Therefore, a region selection algorithm is used to remove large or small regions. This process is performed because a big or small characteristic region will be vulnerable if the local geometric transform is applied. All the regions whose characteristic scale is below 2 or above 12 are removed. Moreover, the region with smaller second derivative test discriminant is also removed when there are regions overlapped with each other.

The secret data is embedded in the selected regions. The embedding algorithm is presented in Figure 4.23. The extraction is the reverse process of embedding and the procedure used in given in Figure 4.24.

Experimental results related to this section are tabulated and discussed in Chapter 4, Section 4.1.2.
1. Transform each selected region from an elliptic shape to circular shape using square root of the second moment matrix of a feature point.

2. Calculate the direction of the gradient of the pixels within circular region.

3. Maintain the perceptual quality of the image after insertion of secret data using Noise Visible Function (NVF) (Voloshynovskiy et al., 1999) for pixel x

\[
NVF(x) = \frac{1}{1 + \theta s(x)} \quad \text{where} \quad \theta = \frac{50}{s_{\text{max}}} \tag{4.11}
\]

where \( s(x) \) is the local variance of the pixel, and \( s_{\text{max}} \) is the maximum local variance of the image, and D is an experimentally determined constant. The D is set for 50 or 100 experiments.

4. The secret data bits are inserted into the cover image using the function

\[
I_w(x) = I(x) + (1 - NVF(x)) w(j) + NVF(x) w(j) \tag{4.12}
\]

where \( I(x) \) is the original value of the pixel \( x \), \( I_w(x) \) is the value of the pixel embedded with the secret data \( w(j) \).

5. After embedding the copyright secret data into all selected characteristic regions, each region is translated back to an elliptical one.

**Figure 4.23: Embedding Procedure of Enhanced KP-Gilles**

1. Transform each selected region from an elliptic shape to circular shape using square root of the second moment matrix of a feature point.

2. Calculate the direction of the gradient of the pixels within circular region.

3. A Wiener filter is used to extract the hidden watermark from a difference image which is calculated between the watermarked image and its Wiener-filtered image.

4. A bit-error is the difference between the extracted watermark and the original one. If the bit-error is lower than a predefined threshold, then it is considered as existence of the watermark.

**Figure 4.24: Extraction Procedure of Enhanced KP-Gilles**
4.4.2. Improved DWT Combined With SVD (VCWS)

Under transform-based steganographic schemes, most of the algorithms use Discrete Cosine Transformation and Discrete Wavelet Transformations. Recently, SVD has gained popularity in many applications including steganography. This section first describes the concepts of DWT and SVD followed by the proposed algorithm.

- **Overview of Wavelets**

  The heart of the proposed steganography algorithms is the wavelet transformation technique. Wavelets are mathematical functions that can be used to divide a given signal into different frequency components. The components can then be studied individually with a resolution that matches its scale. A wavelet transform is the representation of a function by wavelets. The major advantages of wavelets are given below:

  - Division of input coding into non-overlapping 2-D blocks is not required
  - Allows good localization both in time and spatial frequency domain
  - Transformation of the whole image that introduces inherent scaling
  - Better identification of data that is relevant to human perception thus achieving high quality of extracted secret data.

  The goal of any steganography algorithm is to embed data in a way that it maintains visual quality of the cover image. The transformation technique works by applying the transform to one row at a time, then continues by transforming the columns. Let the sequence \( f_0, f_1, \ldots, f_n \) describes the values of grey scale components in a row of pixels. The aim is to separate rapid changes in the sequence from slower changes. To this end, a sequence of wavelet coefficients is created as below:
Wavelets have the added advantage that even coefficients record the average of two successive values. This is called as the low pass band since information about high frequency changes is lost. While the odd coefficients record the difference in two successive values and is called as the high pass band as high frequency information is passed on. The number of low pass coefficients is half the number of values in the original sequence (as is the number of high pass coefficients). The original $f$ values can be recovered from the wavelets for reconstructing the image as given below:

\[
\begin{align*}
    a_0 &= \frac{f_0 + f_1}{2} \\
    a_1 &= \frac{f_0 - f_1}{2} \\
    a_2 &= \frac{f_2 + f_3}{2} \\
    a_3 &= \frac{f_2 - f_3}{2} \\
    \vdots &= \vdots
\end{align*}
\]  

\[ (4.11) \]

The wavelet coefficients are reordered by listing the low pass coefficients first followed by the high pass coefficients. The same operation is applied to transform the wavelet coefficients vertically. This result in a 2-dimensional grid of wavelet coefficients divided into four blocks by the low and high pass bands (Figure 4.25)

![Figure 4.25: Wavelet Coefficient divided into Four Blocks](image)

The LL region is obtained by averaging the values in a 2 by 2 block and so represents a lower resolution version of the image. In practice, the image is broken into tiles, usually of size 64 by 64 (Figure 4.26). If the
coefficients in the LL region are transmitted first, the image at a lower resolution can be reconstructed before all the coefficients had arrived. The same operation is performed on the LL region thereby obtaining images of lower resolution (Figure 4.27).

![Image Broken into Tiles](image1.png)

**Figure 4.26: Image Broken into Tiles**

![Decomposed Blocks](image2.png)

**Fig. 4.27: Decomposed Blocks**

The wavelet coefficients are computed through a lifting process as below:

\[
\begin{align*}
    a_0 &= \frac{(f_0 + f_1)}{2} \\
    a_1 &= a_0 - f_1
\end{align*}
\]  \hspace{1cm} (4.13)

The advantage is that the coefficients may be computed without using additional computer memory, that is, \(a_0\) first replaces \(f_0\) and then \(a_1\) replaces \(f_1\). Also, the lifting process enables faster computation of the
coefficients. Wavelet transforms have proven to be very powerful tool in image steganography (Taubman and Marcellin, 2002). In this research work, Haar wavelets are employed during the embedding and extraction of secret data into the cover image.

- **Overview of SVD**

  Let $A$ be an image matrix of size $N \times N$ and with rank $r$, such that $r \leq N$. In SVD, the matrix $A$ can be decomposed into a product of 3 matrices using Equations (4.14-4.16).

  \[
  A = U \times \lambda \times V^T \quad \text{(4.14)}
  \]

  \[
  = [u_1, u_2, \ldots, u_N] \times \begin{bmatrix}
  \lambda_1 \\
  \lambda_2 \\
  \ddots \\
  \lambda_n
  \end{bmatrix} \times [V_1, V_2, \ldots, V_N]^T
  \]

  \[
  = \sum_{i=1}^{r} \lambda_i u_i v_i^T \quad \text{(4.15)}
  \]

  where $U$ and $V$ are called left and right singular vectors and are orthogonal matrices such that $U^T U = I$, $V^T V = I$. $S$ is a $N \times N$ diagonal matrix whose elements ($\lambda$'s) are singular values of $A$.

  Thus the SVD of $A$ can be represented as Equation (4.17).

  \[
  A = \lambda_1 U_1 V_1^T + \lambda_2 U_2 V_2^T + \ldots + \lambda_r U_r V_r^T \quad \text{(4.17)}
  \]

  where $r$ is the rank of matrix $A$. Each singular value specifies the luminance of an image layer while the corresponding pair of singular vectors specifies the geometry of the image layer.
In SVD-based steganography, generally, SVD is applied to the whole cover image and all the singular values are modified to embed the secret data. An important property of SVD-based steganography is that the largest of the modified singular values change very little for most types of attacks. A theoretical analysis of the effects of various attacks on the singular values of an image is provided by Zhou and Chen (2004) and showed that they provide maximum security and can be used in image steganography.

In this section, a method that combines DWT and SVD in a novel fashion is presented. The combination of DWT-SVD is already examined by some researchers (Rezazadeh and Rafiei, 2007; Ganic and Eskicioglu, 2004a, 2004b). However, the existing algorithms still can be improved in terms of security, capacity, robustness and transparency. To achieve this, this work presents a new fashion of combining DWT-SVD algorithm proposed by adding a simple encryption algorithm, combining a content-based key, using visual cryptography and a novel shuffling method to embed the secret data in the wavelet coefficients.
- Embedding Process

**Step 1: Prepare Cover Image**

COVER IMAGE → DWT → SVD

**Step 2: Preparing Secret Image**

SECRET DATA → Visual Cryptography RPP Procedure

SECRET DATA → Share 1 → Encryption → DWT → SVD

SVD

**Step 3: Embedding Process**

SVD_{LL1} → SVD_{LH1} → SVD_{HL1} → SVD_{HH1}

SVD_{LL2} → SVD_{LH2} → SVD_{HL2} → SVD_{HH2}

Shuffle Embedding

IDWT

Stegano Image

Figure 4.28: Embedding Process of DWT-SVD Algorithm
The proposed steganography algorithm that combines DWT and SVD consists of three steps (Figure 4.28).

In the first step, DWT is applied on cover image to obtain four quadrants (LL, LH, HL and HH) containing wavelet coefficients. The SVD for each quadrant is calculated. Let this be termed as SVD$_{LL}$, SVD$_{LH}$, SVD$_{HL}$ and SVD$_{HH}$. In the second step, the secret data is first divided into two shares using VC algorithm. Share 1 is kept separately, while Share 2 is used to create a content-based key for added security. The process is given below.

1. Perform three level decomposition on Share 2 and select the LL3 subband.

2. Divide the LL3 bands into fixed size non-overlapping blocks (8 x 8 block used in this study)

3. Perform quantization for each block and store it as secret key. The quantization $q_i$ for $i^{th}$ block is computed using Equation 4.18.

\[
q_i = q_m + (q_M - q_m) \times \frac{S_i - S_{\text{min}}}{S_{\text{max}} - S_{\text{min}}}
\]

(4.18)

where $S_i = m_i \times (\sigma_i)^{1.25}$, $m$ and $\sigma$ are the mean value and standard deviation of $i^{th}$ block in the wavelet domain. This is done to take into account the structural and background statistics of the block. Parameters $q_m$ and $q_M$ represent the minimum and maximum quantization step values respectively. The parameter $q_i$ denotes the quantization step of block $i$ and is proportional to the numerical value $S_i$. These quantization parameters are saved as secret keys for secret data recovery.

4. Apply SVD on each block and then compute Euclidean norm of singular values of the blocks. Let $\lambda_i = [\lambda_1^i, \lambda_2^i, ..., \lambda_r^i]$ be the vector of
singular values of block i, the norm of this vector is calculated using the formula

\[ N_i = \text{Norm}(\lambda_i) = \sqrt{\sum_{p=1}^{r} (\lambda_{ip})^2} \]  \hspace{1cm} (4.19)

5. Quantize the norm value to 0 or 1 using Equation (4.20) (Sun \textit{et al.}, 2002; Chen and Wornell, 2001).

\[ Z_i = N_i \mod q_i \] \hspace{1cm} (4.20)

If secret data bit is zero, then \( N_i^w \) is calculated using Equation (4.21).

Else if secret data bit is one then \( N_i^w \) is calculated using Equation (4.22).

\[ N_i^2 = \begin{cases} N_i + \frac{q_i}{4} - Z_i & \text{if } Z_i < \frac{3q_i}{4} \\ N_i + \frac{5q_i}{4} - Z_i & \text{Otherwise} \end{cases} \] \hspace{1cm} (4.21)

\[ N_i^2 = \begin{cases} N_i + \frac{q_i}{4} - Z_i & \text{if } Z_i < \frac{q_i}{4} \\ N_i + \frac{3q_i}{4} - Z_i & \text{Otherwise} \end{cases} \] \hspace{1cm} (4.22)

6. The modified singular value vector for each block is calculated using Equation 4.23.

\[ \lambda_i^w = \lambda_i \times \frac{N_i^w}{N_i} \] \hspace{1cm} (4.23)

7. Perform inverse SVD and DWT to obtain secured copy of Share 2.

8. Now compress and combine Share 1 and Share 2 as explained in Section 3.3 and calculate SVD of the new modified image.

The third step is the actual embedding part. The embedding procedure uses a novel shuffling manner to embed the secret methods SVD.
values into the cover image. The shuffling algorithm modifies the singular values of the decomposition image in each subband with the singular values of the secret data (Figure 4.28). After embedding an inverse SVD and DWT is performed on the cover image to obtain the stegno image.

The embedding algorithm has the advantage of providing multiple securities through the use of visual cryptography, encryption and content based embedding of the secret data. Moreover, the usage of SVD makes the algorithm more robust and perceptible.
• Extraction process

![Diagram of Extraction Process of DWT-SVD Algorithm]

**Figure 4.29: Extraction Process of DWT-SVD Algorithm**
The extraction process is just the reverse of embedding process. The algorithm first performs DWT and then SVD to extract the secret data, from which share 2 is obtained. The secret key stored inside share 2 is retrieved and verified. After successful verification Share 1 and 2 are stacked together using procedure given in Section 3.1 to retrieve the secret data. The procedure used is given in Figure 4.29. Experimental results related to this section are presented in Chapter 4, Section 4.1.3.

4.5. CHAPTER SUMMARY

This chapter presented the enhanced secure message routing algorithm and the various steganographic algorithms to hide secret data into cover image. All the proposed algorithms have been tested to analyze the advantages obtained. The results obtained are tabulated and discussed in the next chapter, Results and Discussion.