

INVISIBLE WATERMARKING FOR AI-GENERATED IMAGES: A COMPARATIVE STUDY OF SPATIAL AND TRANSFORM DOMAIN TECHNIQUES

Dr. D.S. Zingade¹, Parth Dhengle², Atharva Deo³

¹Department of Artificial Intelligence and Data Science, AISSMS IOIT

²Department of Artificial Intelligence and Data Science, AISSMS IOIT

³Department of Artificial Intelligence and Data Science, AISSMS IOIT

Abstract - This paper explores the working principle of Invisible Watermarking in AI generated images. As diffusion models like Stable Diffusion XL generate photorealistic images indiscernible from human created content, the risk of undetectable deepfakes erodes trust in digital media, due to which ensuring the authenticity of digital content has become significantly important. Recent studies confirm that AI generated images now achieve near human level perceptual fidelity. To identify AI generated content, watermarks are used. It helps in preventing the spread of misinformation through deep fake, protecting property rights, accountability and more. This study examines methods such as Spatial Domain Approach and Transform Domain Approach, used for embedding data/information in a highly realistic image produced by AI generative models. The study highlights how an image serves as a carrier or a host for embedded data or secret information without making any noticeable changes in the original image. The work provides the first comparative analysis of spatial vs. transform domain watermarking, guiding practical implementation choices for AI integrity frameworks.

Key Words: Invisible watermarking, steganography, LSB, DCT, DWT, DFT, AI-generated images, digital watermarking, steganalysis, PSNR, SSIM, deepfake detection, C2PA, content provenance.

1. Objectives

1. Analyze the challenge of distinguishing AI-generated images from human-created content and its escalating impact on digital media integrity and the spread of misinformation.
2. Study invisible watermarking as a principled mechanism for embedding hidden provenance information in AI-generated images without perceptible quality degradation.
3. Understand the role of steganography in watermarking and differentiate it from cryptography in terms of security model, detectability, and practical deployment.
4. Explore spatial-domain watermarking (LSB technique) in depth-covering its working mechanism, capacity analysis, performance metrics, and fundamental limitations under common image-processing attacks.
5. Examine transform-domain watermarking techniques (DCT, DWT, DFT) as more robust alternatives, focusing on frequency-coefficient selection, embedding strategies, and robustness against compression, noise, and geometric attacks.
6. Implement and rigorously analyses DCT-based watermarking including spatial-to-frequency transformation, zig-zag coefficient ordering, mid-frequency embedding, parity-based bit encoding, and IDCT reconstruction with worked numerical examples.
7. Present a structured Results and Discussion section with representative PSNR, SSIM, MSE, BER, and NC values comparing LSB, DCT, and DWT methods across multiple attack scenarios.
8. Highlight contemporary challenges including diffusion-based removal attacks and steganalysis and discuss the role of emerging standards such as C2PA

2. Introduction

The emergence of powerful Generative Artificial Intelligence (AI) systems has fundamentally transformed digital content creation. Advanced models such as Generative Adversarial Networks (GANs), Stable Diffusion, DALL-E, and Midjourney are now capable of synthesizing highly realistic images at large scale. These systems can generate photorealistic outputs with impressive speed and quality, significantly reducing the effort required for visual content production [1,2]. Research has shown that AI-generated faces are often indistinguishable from real human faces and may even be perceived as more trustworthy by human observers [3]. As a result, the distinction between human-created and machine-generated media has become increasingly blurred.

As AI-generated images become more realistic and accessible, several societal and technological concerns have emerged. These include threats to digital media integrity, intellectual property ownership, academic dishonesty, and the rapid spread of misinformation through deepfakes and synthetic media [4]. In many situations, it is becoming difficult to verify whether an image was created by a human or generated by an AI system. This weakens trust in digital evidence and creates serious challenges for online platforms, institutions, and regulatory systems.

To address these concerns, watermarking has emerged as a proactive solution. AI watermarking refers to embedding hidden signals or digital signatures directly into generated media so that later algorithms can identify the source model, user, or creation context [5]. Unlike metadata-based methods, watermarking remains useful even when file information is removed or altered. Properly designed watermarking systems can survive common modifications such as resizing, compression, and redistribution. This makes watermarking valuable for content tracing, ownership verification, and responsible AI deployment.

Among various watermarking approaches, invisible watermarking has gained major importance because it preserves the visual quality of the image. Invisible watermarking embeds secret information inside the image in a manner that is not noticeable to human eyes but can be detected by specialized algorithms [6]. This ensures that the image remains aesthetically unchanged while still carrying hidden ownership or authenticity information.

Invisible watermarking techniques are generally divided into two broad categories: spatial-domain methods and transform-domain methods. Spatial-domain methods operate directly on pixel values, with Least Significant Bit (LSB) substitution being the most common example. These methods are simple, fast, and provide high payload capacity, but they are highly fragile against attacks such as compression or noise addition [7]. Transform-domain methods, including Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), and Discrete Fourier Transform (DFT), embed watermark data into frequency coefficients rather than raw pixels. As a result, they offer better robustness against image processing operations and compression.

However, designing a practical watermarking system remains challenging. A successful method must balance three key requirements: invisibility, robustness, and payload capacity. Stronger watermarks may become visible, while highly invisible watermarks may be easier to remove. Recent studies have also shown that advanced AI-based regeneration attacks can erase many existing watermarks while preserving image quality [8]. Therefore, improving watermark resilience while maintaining image fidelity remains an active area of research.

This paper presents a comparative study of classical invisible watermarking techniques for AI-generated images, focusing on spatial-domain LSB methods and transform-domain approaches such as DCT, DWT, and DFT. The objective is to analyze their theoretical foundations, performance characteristics, strengths, limitations, and suitability for modern AI-generated media applications.

3. Literature Review

A substantial body of research underpins the field of invisible watermarking. We review foundational and recent work across spatial, transform, hybrid, and deep-learning-based paradigms.

1. **Spatial Domain Methods:** Panigrahi and Padhy (2025) proposed an LSB-based steganographic framework for secure data hiding, demonstrating PSNR values above 47 dB for standard images at moderate payload capacity [8]. Their study systematically analyzed the trade-off between the number of modified LSB planes and image quality, concluding that modifying beyond two LSB layers per channel significantly increases statistical detectability. Ghazanfari et al. (2011) introduced LSB++, an improved version of the LSB+ family that applies adaptive selection of embedding positions to reduce histogram anomalies exploitable by RS steganalysis [15].
2. **Transform Domain Methods:** Saikrishna and Resmipriya (2016) proposed invisible logo watermarking using Discrete Wavelet Transform (DWT) combined with Arnold transformation, embedding watermark data in frequency sub-bands to improve robustness against geometric and signal-processing attacks while maintaining high imperceptibility [9]. Wang et al. (2021) designed a Discrete Cosine Transform (DCT)-based invisible watermarking method for efficient hardware implementation, embedding data in mid-frequency coefficients and reporting PSNR values above 40 dB with strong robustness against JPEG compression [10]. Chandra (2012) combined DWT and DFT using pseudo-random sequences for copyright protection, showing that hybrid transform techniques improve resistance against signal-processing attacks [11]. Mandal et al. (2022) presented a comprehensive survey of digital image steganography, systematically cataloguing attack models, embedding strategies, and evaluation metrics [12]. Razafindradina and Karim demonstrated that DWT embedding in the HH sub-band with edge insertion provides robustness against GIF/JPEG compression, histogram spreading, noise, and small rotations, with an average PSNR of 26 dB [2].
3. **Hybrid and Multi-Domain Methods:** Multi-transform domain approaches combining DCT and DWT have consistently outperformed single-domain methods in robustness evaluations. Mohammed et al. employed a continuous DCT-DWT transformation with adaptive RGB component selection for watermark embedding, demonstrating resilience to geometric and filtering attacks while preserving invisibility [2]. Recent forensic DCT-DWT watermarking studies achieved PSNR values between 46 dB and 49 dB with BER = 0 under JPEG compression at quality factors above 60, while allowing perfect watermark recovery after rotations up to 90° [4]. Hybrid DWT-

DCT methods with CNN decoding have also reported PSNR values between 64.87 and 68.75 dB in medical imaging applications [14].

4. **Deep Learning-Based Methods:** Zhu et al. (2018) proposed Hidden, the first end-to-end trainable neural watermarking system incorporating a differentiable noise simulation layer between encoder and decoder networks. Tancik et al. (2020) introduced StegaStamp, which encodes 56-bit payloads with robustness to print-rescan scenarios. Bui et al. (2023) proposed TrustMark, a GAN-based universal watermarking framework balancing imperceptibility and robustness at arbitrary resolutions [2,4]. InvisMark (Microsoft, 2024) reported PSNR around 51 dB and SSIM of approximately 0.998 with over 97%-bit accuracy across image manipulations while embedding 256-bit watermarks [14]. However, diffusion-based regeneration attacks can reduce detection rates for StegaStamp and TrustMark to below 5%, compared with 30–50% under JPEG attacks [3].
5. **Standards and Industry Adoption:** C2PA version 2.1 (2024) incorporated Digimarc digital watermarking as an approved soft-binding mechanism within the Content Credentials framework [12]. A 2018 Imatag study found that 80% of images across 750 websites had metadata stripped, motivating invisible watermarks as provenance anchors that survive metadata loss [13]. The EU AI Act and emerging NIST guidelines increasingly reference watermarking as a technical measure for AI transparency and accountability.

4. Background

4.1 Steganography

Steganography [12,6] is the practice of hiding secret information inside another file or a medium in such a way that the existence of the information itself is concealed. Unlike cryptography, which secures communication by encrypting readable data, known as plaintext, into an unreadable format called ciphertext using mathematical algorithms and keys, ensuring that only authorized parties with the correct key can decrypt it back to its original form, steganography hides the data itself within images, audio, video, or text, so that the message remains invisible to unintended observers. In modern cybersecurity, it is used to smuggle data past security scanners by embedding hidden content in innocuous files, often utilizing methods like Least Significant Bit (LSB) [13] manipulation to alter pixel values without changing the image's visual appearance.

4.1.1 Steganography vs Cryptography

Table 1: Steganography vs Cryptography

Aspect	Steganography	Cryptography
Definition	Technique of hiding the existence of information within another medium	Technique of transforming information to make it unreadable to unauthorized users
Primary Goal	Conceal the presence of communication	Protect the content of communication
Visibility	Hidden data is invisible and undetectable to observers	Encrypted data is visible but appears meaningless
Data Appearance	Looks like a normal file (image, audio, etc.)	Looks like random or scrambled data
Security Approach	Relies on secrecy of embedding method	Relies on encryption algorithms and keys
Detection	Difficult to detect if done properly	Easy to detect (but hard to decode without key)
Robustness	Less robust; hidden data can be lost if the file is modified	More robust; encryption remains secure under transformations
Use Cases	Covert communication, digital watermarking, data hiding	Secure communication, data protection, authentication
Example	Secret message hidden inside an image	Message encrypted using AES or RSA
Combination	Often combined with cryptography for added security	Can be combined with steganography for concealment

4.2 Spatial Domain

The spatial domain [14] is the representation of an image in terms of its pixel values, where operations are performed directly on those pixels [15]. Spatial domain methods manipulate pixels either point wise (e.g., contrast stretching, thresholding) or neighborhood-wise (e.g., spatial filtering using a kernel or mask). The term spatial refers to space. In the context of digital

images, this space is a two-dimensional plane, commonly called the xy-plane. Therefore, the spatial domain refers to the image plane itself.

4.2.1 Mathematical Representation

Spatial domain methods involve directly modifying the values of pixels in an image. These processes can be mathematically represented as:

$$I_1(x, y) = T[I(x, y)]$$

Where:

- $I(x, y)$ represents the original image,
- $I_1(x, y)$ represents the processed (modified) image,
- T is an operation applied to the pixel at coordinates (x, y) , typically based on its neighbourhood.

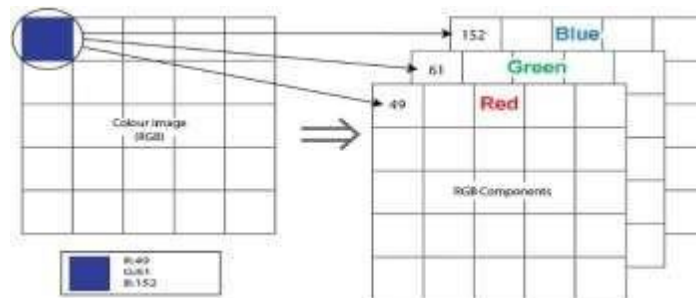


Figure 1: Spatial domain for colour image (RGB)

4.3 Frequency Domain

The frequency domain in image processing is a representation where an image is analyzed based on its frequency components rather than its pixel intensities at specific spatial coordinates [7]. In this domain, image processing operations such as filtering are often more efficient because convolution in the spatial domain becomes simple multiplication in the frequency domain. This approach is particularly effective for addressing global image properties such as removing periodic noise, smoothing images, or enhancing specific frequency bands, tasks that can be computationally expensive or difficult to quantify directly in the spatial domain.

4.3.1 Mathematical Representation

The transformation from spatial domain to frequency domain is expressed as:

$$F(u, v) = \mathcal{F}[I(x, y)]$$

Where:

- $I(x, y) \rightarrow$ original image (spatial domain)
- $F(u, v) \rightarrow$ transformed image (frequency domain)
- $\mathcal{F} \rightarrow$ Fourier Transform
- $(u, v) \rightarrow$ frequency coordinates

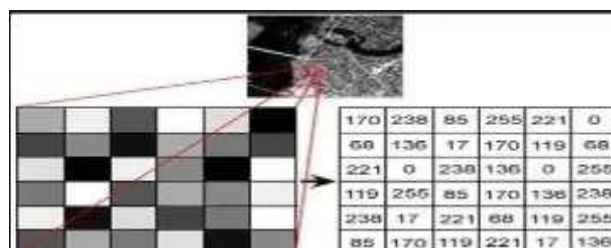


Figure 2: Frequency domain representation: 2D Fourier Transform converting spatial image to frequency spectrum

4.4 Performance Metrics

The quality, imperceptibility, and robustness of a watermarking scheme are commonly evaluated using several standard performance metrics. These metrics help determine how effectively a watermark is embedded while preserving image quality and surviving different attacks or distortions.

- **Peak Signal-to-Noise Ratio (PSNR):** Measures the visual similarity between the original and watermarked image and is widely used to evaluate imperceptibility. It is mathematically expressed as:

$$PSNR = 10\log_{10} \left(\frac{MAX^2}{MSE} \right)$$

where MAX is the maximum possible pixel intensity value. PSNR values above 40 dB are generally considered imperceptible to the human eye, while values above 50 dB indicate near-perfect image quality.

- **Structural Similarity Index (SSIM):** Measures perceptual similarity by comparing luminance, contrast, and structural information between the original and watermarked images. SSIM values range from 0 to 1, where values above 0.99 indicate nearly identical images. SSIM is often preferred over PSNR because it models human visual perception more accurately [9].
- **Mean Square Error (MSE):** Represents the average squared pixel difference between the original and watermarked images. Lower MSE values indicate better image quality and less distortion.
- **Bit Error Rate (BER):** Measures the fraction of watermark bits incorrectly extracted after attacks or image modifications. It is given by:

$$BER = \frac{\text{Number of Incorrect Bits}}{\text{Total Number of Bits}}$$

A BER value of 0 indicates perfect watermark recovery, while values below 0.01 are generally considered acceptable.

- **Normalised Cross-Correlation (NC):** Measures similarity between the original watermark and the extracted watermark pattern or logo. A value of $NC = 1.0$ indicates perfect recovery and strong robustness.

5. Spatial Domain Watermarking: LSB Technique

5.1 Definition and Motivation

The Least Significant Bit (LSB) is the rightmost bit in a binary representation of a pixel value and carries the smallest weight ($2^0 = 1$) [13]. In an 8-bit grayscale channel, changing the LSB modifies the pixel value by at most 1 out of 255, which is generally imperceptible to the human eye. Due to the human visual system's low sensitivity to such minor intensity variations, LSB has become the foundational spatial-domain steganographic primitive.

In a standard RGB image, each pixel consists of three 8-bit colour channels: Red, Green, and Blue. This provides three possible embedding positions per pixel. The LSB of each channel can be replaced by one bit of a secret binary stream. Because the visual distortion is minimal and the algorithm is extremely simple to implement, LSB remains widely used in lightweight watermarking and data-hiding applications despite its known fragility [8].

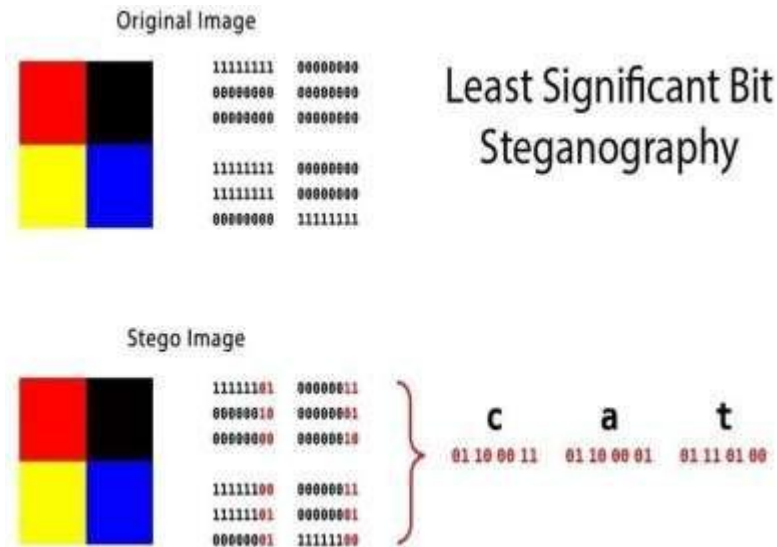


Figure 3: Least Significant Bit (LSB) steganography: embedding the word “cat” into pixel values by modifying the least significant bits

5.2 Encoding Procedure

LSB watermark embedding generally follows these steps:

1. Convert the secret message (text, binary stream, or watermark logo) into binary form using ASCII, UTF-8, or another suitable encoding.
2. Traverse the carrier image pixel by pixel in raster scan order or pseudo-random order using a secret key.
3. Replace the LSB of each pixel channel with the next bit of the binary message.
4. Continue until the complete message is embedded, then store or transmit the stego-image.

Mathematically, for a pixel channel value p and message bit $b \in \{0,1\}$:

$$p' = (p \text{ AND } 11111110_2) \text{ OR } b$$

This operation clears the original LSB and inserts the new secret bit, ensuring that the pixel value changes by at most ± 1 .

5.3 Payload Capacity

Modifying one LSB per channel in an RGB image provides 3 bits per pixel. Thus, a 1024×768 image can theoretically store approximately 2.36 million bits, or nearly 295 KB of hidden data. Using two LSBs per channel increases the capacity to 6 bits per pixel, though this also increases statistical detectability. Practical high-quality LSB systems have reported payloads around 2.7 bits per pixel while maintaining PSNR values above 47 dB [13].

5.4 Decoding Procedure

Watermark extraction is the reverse of embedding. The receiver traverses the stego-image in the same order used during encoding, reads the LSB from each pixel channel, reconstructs the binary stream, and converts it back into the original message or watermark format.

5.5 Advantages of LSB

- **Simplicity:** Straightforward implementation without requiring complex mathematical transforms.
- **High Imperceptibility:** PSNR values typically range between 42 dB and 60 dB for one or two LSB modifications, preserving visual quality [6,7].
- **Large Capacity:** Provides one of the highest payload capacities among spatial-domain techniques.
- **Universal Compatibility:** Works effectively with common lossless image formats such as BMP and PNG.

5.6 Limitations and Challenges

- **Fragility to Compression:** JPEG compression re-quantises pixel values and often destroys embedded LSB data entirely.

- **Sensitivity to Noise and Filtering:** Gaussian noise, salt-and-pepper noise, and filtering operations can corrupt hidden bits.
- **Statistical Detectability:** LSB embedding alters least significant bit-plane statistics, making detection possible through RS analysis, chi-square tests, and CNN-based steganalysis [8].
- **Limited Robustness:** Offers weak resistance against deliberate watermark removal attacks.
- **Geometric Fragility:** Cropping, rotation, and scaling disturb pixel alignment, causing synchronisation loss during extraction.

6. Transform Domain Techniques

6.1 Introduction to Transform Domain

6.1.1 Definition

Transform domain steganography refers to a class of data hiding techniques in which the cover image is first converted from the spatial domain into a frequency domain representation [12]. Instead of directly modifying pixel intensities, secret information is embedded within the transformed coefficients obtained through mathematical transformations such as Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), or Discrete Fourier Transform (DFT).

6.1.2 Fundamental Principle

The fundamental concept behind transform domain techniques is to manipulate the frequency components of an image rather than its raw pixel values. The general workflow can be summarized as follows:

1. The input image (spatial domain) is transformed into the frequency domain using an appropriate transform.
2. The resulting frequency coefficients represent different components such as low, mid, and high frequencies.
3. Secret data is embedded into selected coefficients based on a defined embedding strategy.
4. An inverse transform is applied to reconstruct the stego image in the spatial domain.

6.1.3 Motivation for Transform Domain Techniques

Spatial domain methods, such as Least Significant Bit (LSB) substitution, suffer from several limitations, including vulnerability to statistical attacks and poor robustness against common image processing operations like compression and filtering [8]. Transform domain approaches address these limitations by:

- Improving resistance to steganalysis attacks
- Providing higher robustness against lossy compression (e.g., JPEG)
- Preserving perceptual quality of the image

6.1.4 Working Mechanism

In transform domain steganography, the image is represented in terms of its frequency characteristics rather than pixel intensities. These characteristics include:

- Frequency components (low, mid, high)
- Energy distribution across the image
- Signal representation in transformed space

Embedding is typically performed in mid-frequency coefficients, as they offer a balance between imperceptibility and robustness [10].



Figure 4: Flow chart of the working mechanism used in transform domain steganography

6.1.5 Security Advantages

Transform domain techniques offer enhanced security due to the indirect nature of data embedding. Since the hidden information is distributed within frequency coefficients [6]:

- It is not directly observable in pixel values
- It is less susceptible to simple visual or statistical detection
- It can withstand moderate image processing operations, including compression and noise addition

6.2 Types of Transform Methods

Transform domain steganography primarily employs three widely adopted mathematical transforms: Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), and Discrete Fourier Transform (DFT). These methods differ in terms of frequency representation, computational complexity, and robustness characteristics.

Table 2: Comparison of Transform Domain Methods

Method	Domain Type	Key Strength	Common Application
DCT	Frequency Domain	Compression compatibility	JPEG compression
DWT	Multi-resolution	High perceptual quality	Image processing
DFT	Full frequency	Accurate frequency analysis	Signal processing

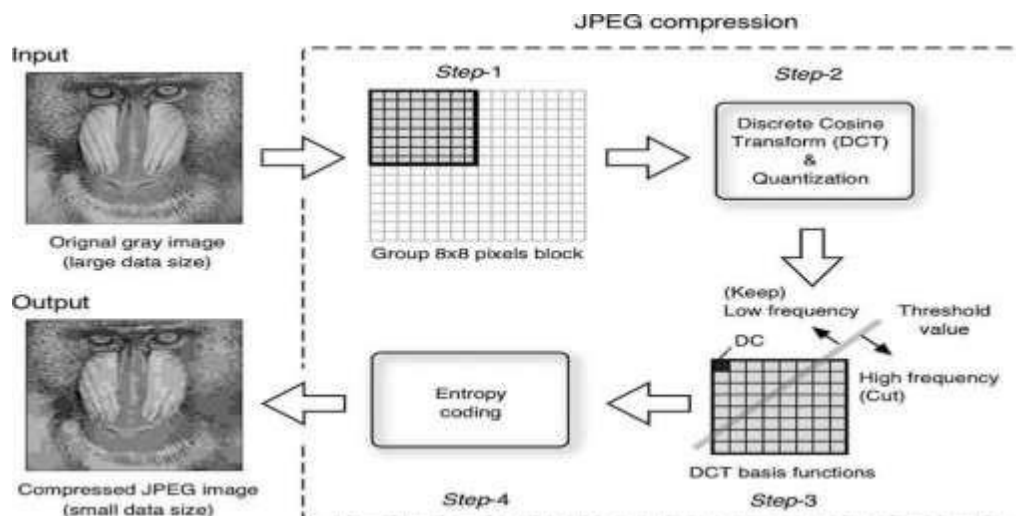
6.2.1 Discrete Cosine Transform (DCT)

Definition: The Discrete Cosine Transform (DCT) converts spatial image data into a sum of cosine functions oscillating at different frequencies. It is widely used in image compression standards such as JPEG .

Working Mechanism:

- Image is divided into blocks (typically 8×8)
- Each block is transformed into frequency coefficients
- Coefficients are categorized into:
 - Low-frequency (visual content)
 - Mid-frequency (embedding region)

Figure 5: Working of Discrete Cosine Transform (DCT) in JPEG compression



Embedding Strategy: Data is typically embedded in mid-frequency coefficients to achieve a balance between imperceptibility and robustness.

Advantages:

- Robust against compression
- Maintains visual quality
- Efficient due to block-based processing

6.2.2 Discrete Wavelet Transform (DWT)

Definition: The Discrete Wavelet Transform (DWT) decomposes an image into multiple sub-bands at different resolutions, enabling localized frequency analysis [9].

Sub-band Decomposition:

- LL: Approximation (low frequency)
- LH: Horizontal details
- HL: Vertical details
- HH: Diagonal details

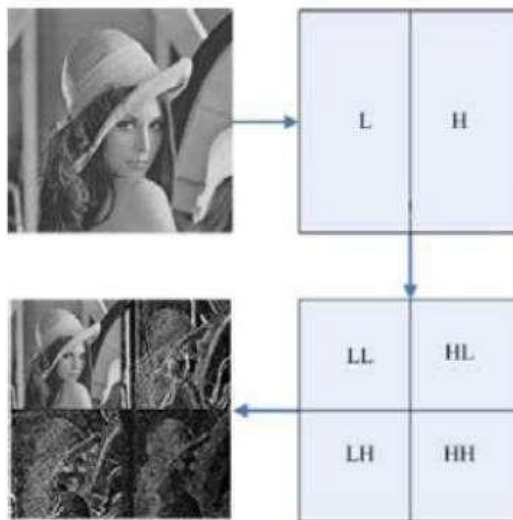


Figure 6: Working of Discrete Wavelet Transform (DWT): sub-band decomposition into LL, LH, HL, and HH components

Working Mechanism:

- Image is decomposed into hierarchical sub-bands
- Secret data is embedded in selected bands (commonly LH, HL, or HH)
- Inverse DWT reconstructs the stego image

6.2.3 Discrete Fourier Transform (DFT)

Definition: The Discrete Fourier Transform (DFT) represents an image in terms of sinusoidal components, capturing both amplitude and phase information [11].

Working Mechanism:

- Image is transformed into frequency spectrum
- Data is embedded into magnitude or phase components
- Inverse DFT reconstructs the image

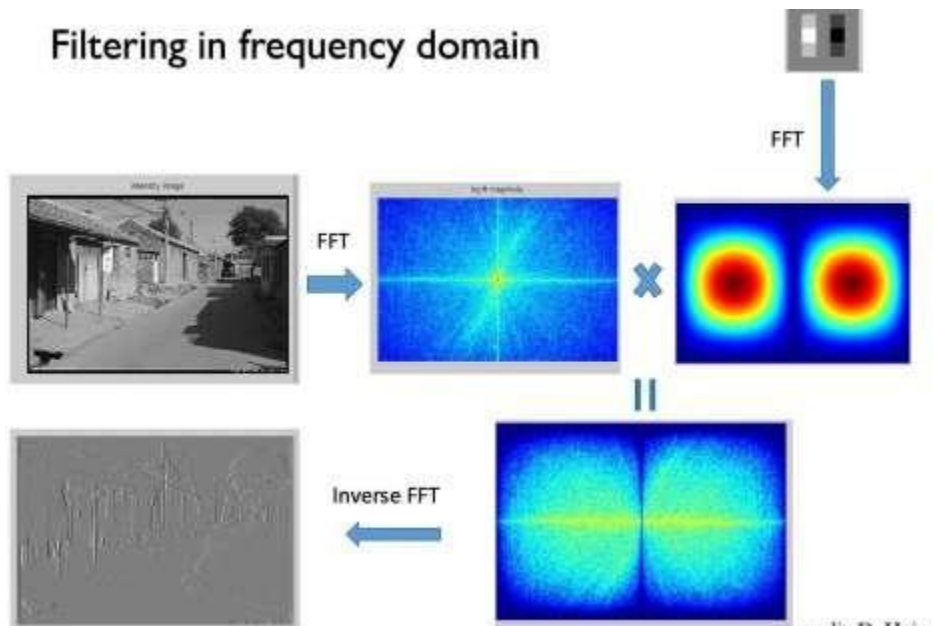


Figure 7 Working of Discrete Fourier Transform (DFT): filtering in the frequency domain

In this research paper we will be focusing on the DCT based steganography which is widely used and accepted as well as easier for understanding.

7. DCT-Based Steganography

7.1 DCT-Based Steganography : Data hiding process

Discrete Cosine Transform (DCT)-based steganography is a transform domain technique where secret data is embedded into an image by modifying its frequency coefficients rather than directly altering pixel values [11].

7.1.1 Image in Spatial Domain (Input Stage)

The initial stage of DCT-based steganography begins with the acquisition of a digital image in its spatial domain representation. In this domain, an image is interpreted as a two-dimensional array of pixel values, where each pixel encodes colour information in terms of intensity levels. In a standard colour image, each pixel is represented using three colour components: (R, G, B). Each component typically occupies 8 bits, resulting in a total of 24 bits per pixel. For instance, a pixel with values:

$$R = 120, G = 65, B = 200$$

represents a specific colour formed by the combination of red, green, and blue intensities.

Conversion to Intensity Representation: Since transform domain techniques such as DCT operate more effectively on intensity variations rather than raw color information, the input image is first converted into a luminance-based representation. This is commonly achieved by transforming the RGB image into the YCbCr color space, where:

- Y represents luminance (brightness)
- Cb represents chrominance (blue difference)
- Cr represents chrominance (red difference)

The luminance component (Y) is computed as:

$$Y = 0.299R + 0.587G + 0.114B$$

This transformation is essential because the human visual system is significantly more sensitive to changes in brightness than to changes in colour. Therefore, operating on the luminance channel allows data to be embedded with minimal perceptual distortion [10].

Formation of Spatial Matrix: After conversion, the image is represented as a matrix of intensity values. For example, a small portion of the image may be represented as:

52	55	61	66	70	61	64	73
63	59	55	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

Each value in this matrix corresponds to the brightness of a pixel at a specific spatial location.

7.1.2 Transform Stage (DCT: Spatial → Frequency Domain)

At this stage, the previously obtained 8×8 brightness matrix is transformed from the spatial domain into the frequency domain using the Discrete Cosine Transform (DCT). This transformation does not alter the amount of information in the image; instead, it changes the way the information is represented.

Centering of Pixel Values: Before applying DCT, each pixel value is shifted by subtracting the average of brightness. In the above example of brightness matrix, the average obtained is:

$$avg = 76.0781 \approx 76$$

$$value = pixel - 76$$

This operation centres the data around zero, which improves the efficiency of the transform.

-24	-21	-15	-10	-6	-15	-12	-3
-13	-17	-21	14	33	9	-7	-4
-14	-17	-8	37	68	28	-10	-3
-13	-18	-5	46	78	30	-6	-7
9	-15	-8	28	50	12	-8	-6
-8	3	2	-6	1	-8	-18	-1
9	-5	-12	-17	-21	-15	-11	7
11	3	-7	-8	-11	0	2	18

Conceptual Working of DCT: The Discrete Cosine Transform converts this matrix into a set of frequency coefficients by comparing the pixel values with a set of pre-defined cosine-based patterns. Each coefficient $C(u, v)$ represents how strongly a particular pattern exists in the block. These patterns range from:

- Smooth (low frequency)
- Gradual variations (mid frequency)
- Rapid intensity changes (high frequency)

Mathematically, DCT computes each coefficient by:

1. Selecting a cosine pattern
2. Multiplying it with the pixel matrix (element-wise)
3. Summing all the values

2×2 DCT basis patterns: uniform, horizontal, vertical, and diagonal

Illustrative 2×2 Example for Understanding: To understand the computation clearly, consider a simplified 2×2 blocks:

$$\begin{pmatrix} -8 & -5 \\ 3 & -1 \end{pmatrix}$$

DCT evaluates this block against four basic patterns:

1. Uniform pattern (low frequency):

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

$$\text{Result: } (-8 \times 1) + (-5 \times 1) + (3 \times 1) + (-1 \times 1) = -11$$

2. Horizontal variation pattern:

$$\begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}$$

$$\text{Result: } (-8 \times 1) + (-5 \times 1) + (3 \times -1) + (-1 \times -1) = -15$$

3. Vertical variation pattern:

$$\begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix}$$

$$\text{Result: } (-8 \times 1) + (-5 \times -1) + (3 \times 1) + (-1 \times -1) = 1$$

4. Diagonal (high frequency) pattern:

$$\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

$$\text{Result: } (-8 \times 1) + (-5 \times -1) + (3 \times -1) + (-1 \times 1) = -7$$

Thus, the frequency representation becomes:

$$\begin{pmatrix} -11 & -15 \\ 1 & -7 \end{pmatrix}$$

Extension to 8×8 Block: The same principle is applied to the 8×8 matrices, but instead of 4 patterns, 64 cosine patterns are used. Each coefficient $C(u, v)$ is computed using a cosine-based weighting function.

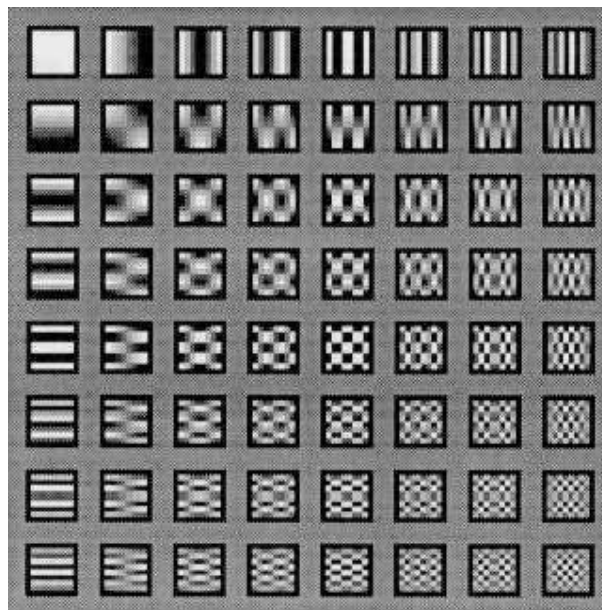


Figure 9: Complete set of 64 DCT basis functions (cosine patterns) for 8×8 block processing

The resulting DCT coefficient matrix for the given 8×8 block is approximately:

$$\begin{matrix} & \begin{matrix} -415 & -30 & -61 & 26 & 56 & -20 & -2 & 0 \end{matrix} \\ \begin{matrix} \text{I} \\ -47 \\ -49 \\ 12 \\ -8 \\ -1 \\ \text{h} \end{matrix} & \begin{pmatrix} 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ 7 & 77 & -25 & -29 & 10 & 5 & -6 & 2 \\ 12 & 34 & -14 & -10 & 6 & 2 & -3 & 3 \\ -7 & -13 & -4 & -2 & 2 & -3 & 3 & 1 \\ 3 & 2 & -6 & -2 & 1 & 4 & 2 & 1 \\ 0 & 0 & -2 & -1 & -3 & 4 & -1 & 2 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{pmatrix} \end{matrix}$$

7.1.3 Frequency Coefficients (Quantization) → Data Embedding

At this stage, the transformed 8×8 block is available in the form of frequency coefficients. These coefficients now serve as the medium for embedding the secret information. Instead of modifying pixel values directly, the embedding is performed by making controlled, minimal changes to selected coefficients.

Zig-Zag Conversion (2D $\rightarrow$ 1D Ordering): To systematically embed data, the 8×8 matrix is converted into a one-dimensional sequence using zig-zag traversal. This ordering arranges the coefficients from low frequency to high frequency.

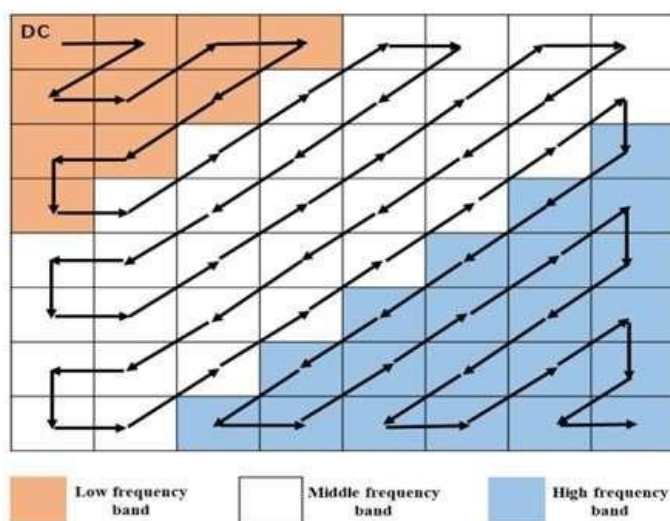


Figure 10: Zig-zag traversal pattern showing low, middle, and high frequency bands

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Figure 11: Zig-zag index ordering matrix for 8×8 block

The sequence becomes:

$$\begin{array}{cccccccc} & -415 & -30 & -61 & 26 & 56 & -20 & -2 & 0 \\ & 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ \mathbf{I} & -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ & -49 & 12 & 34 & -14 & -10 & 6 & 2 & 2 \\ & 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ & -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ & -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ \mathbf{h} & 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{array}$$

Selection of Embedding Region: Not all coefficients are suitable for data hiding. The selection is based on the position of coefficients in the zig-zag sequence:

- The initial coefficients (low frequency) are avoided because they carry most of the visual information.
- The final coefficients (high frequency) are avoided because they are often removed during compression.

Therefore, the embedding is performed in the mid-frequency range, typically indices 6 to 49.

Preparation of Secret Data: The secret message “python” is converted into a binary sequence:

```
01110000 01111001 01110100
01101000 01101111 01101110
```

This forms a continuous bit stream. Each bit in this sequence will be embedded into one coefficient.

Embedding Rule: The embedding is performed using a simple parity rule:

- Even coefficient → represents bit 0
- Odd coefficient → represents bit 1

To embed a bit, the coefficient is adjusted minimally:

- If the coefficient already matches the required parity, it is left unchanged
- Otherwise, it is modified by ± 1

Illustrative Embedding Example: Consider a subset of selected coefficients: [27, -61, 7, -49, 12]. Suppose the bits to embed are: 0 1 1 0 1.

Table 3: Embedding Process Example

Coefficient	Bit	Action	Result
27	0	odd → make even	26
-61	1	already odd	-61
7	1	already odd	7
-49	0	odd → make even	-48
12	1	even → make odd	13

Updated values: [26, -61, 7, -48, 13]. Only minimal changes are introduced, ensuring that distortion remains imperceptible.

Propagation Across Blocks: If the message length exceeds the capacity of one block, embedding continues sequentially across multiple 8×8 blocks. Each block contributes a fixed number of bits depending on the number of usable coefficients.

Reconstruction of Modified Coefficient Matrix: After embedding, the modified sequence is placed back into its original positions using inverse zig-zag mapping, reconstructing the 8×8 coefficient matrix:

	-415	-30	-61	26	56	-20	-2	0
	4	-22	-61	10	13	-7	-9	5
1	-47	7	77	-25	-29	10	5	-6
	-49	12	34	-14	-10	6	2	2
	12	-7	-13	-4	-2	2	-3	3
1	-8	3	2	-6	-2	1	4	2
	-1	0	0	-2	-1	-3	4	-1
h	0	0	-1	-4	-1	0	1	2

7.1.4 Inverse Transform (IDCT → Stego Image Reconstruction)

After embedding the secret data into the selected mid-frequency coefficients, the modified frequency matrix must be converted back into the spatial domain to obtain the final stego image. This reconstruction is performed using the Inverse Discrete Cosine Transform (IDCT), which reverses the transformation applied earlier.

Reconstruction Principle: The IDCT reconstructs pixel values by combining all frequency components (coefficients) with their corresponding cosine patterns. For a pixel located at position (x, y) , the value is computed as:

$$pixel(x, y) = \sum_u \sum_v [C(u, v) \times \text{cosine pattern}(u, v, x, y)]$$

Where $C(u, v)$ are the modified DCT coefficients and the cosine patterns correspond to different spatial frequencies.

Step-by-Step Reconstruction:

Step 1: Pattern Contribution. Each coefficient generates a spatial pattern:

- $C(0,0)$ contributes a constant value across the block

- $C(0,1)$ introduces horizontal variation
- $C(1,0)$ introduces vertical variation
- Higher indices introduce finer details

All these contributions are summed to reconstruct each pixel.

Step 2: Pixel Value Formation. For every pixel position:

1. Multiply each coefficient with its corresponding cosine basis value
2. Sum all 64 contributions
3. Obtain the reconstructed intensity value

This process is repeated for all 64 pixels in the block.

Step 3: Effect of Embedded Data.

Since embedding modifies coefficients only by ± 1 , the resulting change in pixel values is very small and distributed across the block. This ensures that no abrupt change occurs in any single pixel and the modification remains imperceptible to the human eye [10].

Step 4: Reversing Centering. After IDCT, the earlier subtracted shift is reversed:

$$pixel = reconstructed\ value + 128$$

Step 5: Value Normalization. The reconstructed values are constrained within the valid pixel range:

$$0 \leq pixel \leq 255$$

Any value outside this range is clipped to the nearest valid boundary.

Illustrative 2x2 Example: Using the modified coefficient matrix:

$$\begin{pmatrix} -11 & -15 \\ 2 & -7 \end{pmatrix}$$

Each pixel is reconstructed by combining contributions from all four coefficients. The resulting values approximate the original spatial block with slight variations introduced due to embedding.

Block Reconstruction to Image: The above process is applied to each 8×8 blocks independently. Once all blocks are reconstructed, they are combined to form the full luminance (Y) component while the chrominance components (Cb and Cr) remain unchanged. Finally, the YCbCr image is converted back to RGB format, producing the final stego image (R, G, B).

7.2 Data Extraction Process (Transform Domain)

The data extraction process in DCT-based steganography is the reverse of the embedding procedure. The objective is to retrieve the hidden bit stream from the stego image by analysing the modified frequency coefficients. Since the embedding was performed in the transform domain using controlled modifications (parity-based changes), the same transformation steps are reapplied to recover the embedded information accurately. The extraction pipeline follows:

Stego Image $\rightarrow$ Block Division $\rightarrow$ DCT $\rightarrow$ Zig-Zag $\rightarrow$ Read Coefficients $\rightarrow$ Extract Bits $\rightarrow$ Reconstruct Message

7.2.1 DCT Reapplication

The stego image, which appears visually identical to the original image, is first converted into the same working domain used during embedding:

1. Converted from RGB to YCbCr color space
2. The luminance (Y) component is selected
3. Divided into non-overlapping 8×8 blocks

Each block is then centered by subtracting 128:

$$value = pixel - 128$$

The Discrete Cosine Transform (DCT) is applied to each block, producing a frequency coefficient matrix. Due to minimal embedding changes (± 1), the structure of coefficients remains intact, enabling reliable extraction.

7.2.2 Zig-Zag Traversal

The obtained 8×8 coefficient matrix is converted into a one-dimensional sequence using the same zig-zag traversal applied during embedding. This ensures that coefficients are accessed in the exact same order. From this sequence, the same mid-frequency region is selected: indices 6 to 49. These indices correspond to the coefficients where data was embedded.

7.2.3 Bit Extraction

Each selected coefficient is analysed to retrieve the embedded bit using the parity rule:

- Even value $\rightarrow 0$
- Odd value $\rightarrow 1$

Table 4: Bit Extraction Example

Coefficient Value	Type	Bit
26	Even	0
-61	Odd	1
7	Odd	1
-48	Even	0
13	Odd	1

Resulting bit sequence: 01101. This process continues across all selected coefficients and across multiple blocks until the complete bit stream is recovered.

7.2.4 Message Reconstruction

The extracted bit stream is grouped into 8-bit segments:

```
01110000 -> p
01111001 -> y
01110100 -> t
01101000 -> h
01101111 -> o
01101110 -> n
```

Reconstructed message: **python.**

To determine the end of the message, one of the following methods is typically used:

- A predefined message length is stored before embedding begins
- A special delimiter or termination symbol is appended to the hidden message
- A fixed embedding capacity is agreed upon by the sender and receiver

This completes the transform-domain extraction workflow. By applying the same block processing, DCT conversion, zig-zag traversal, and parity checking used during embedding, the hidden message can be recovered accurately while preserving the visual quality of the stego image.

8. Results

8.1 Imperceptibility Performance

Imperceptibility is the primary quality requirement of any invisible watermarking system. Table 1 presents representative PSNR, SSIM, and MSE values from the literature for LSB, DCT, and DWT methods under standard no-attack conditions [8, 9, 10, 11].

Table 5: Imperceptibility Performance: LSB vs. DCT vs. DWT

Method	PSNR	SSIM	MSE	Payload	Notes
LSB (1-bit/channel)	47–60	0.999–1.000	< 0.5	3.0	Best PSNR; fragile to attacks
LSB (2-bit/channel)	42–47	0.997–0.999	1.5–3.5	6.0	Higher capacity; more detectable

DCT ($\alpha = 0.1$)	49–52	0.998–0.999	0.4–0.9	0.1–0.5	Good quality and robustness
DCT ($\alpha = 0.3$)	44–48	0.996–0.999	1.0–2.5	0.1–0.5	More robust but lower PSNR
DWT (1-level)	49–57	0.998–1.000	0.2–0.7	0.2–1.0	Excellent quality
DCT-DWT Hybrid	46–68	0.999–1.000	< 0.5	0.1–0.5	Best quality in secure settings

LSB watermarking achieves the highest PSNR under no-attack conditions because it modifies pixel values by at most ± 1 . DWT and DCT methods also maintain PSNR values well above 40 dB, confirming perceptual transparency. SSIM values above 0.998 across all methods indicate negligible structural distortion.

8.2 Robustness Against Attacks

Robustness is evaluated using BER and NC after applying attacks to the watermarked image. Table [tab:robustness] summarizes representative BER values from benchmark studies [8, 9, 10, 11].

Table 5: Imperceptibility Performance: LSB vs. DCT vs. DWT

Attack	LSB (%)	DCT (%)	DWT (%)	DCT-DWT (%)
No Attack	0.0	0.0	0.0	0.0
JPEG (QF=80)	~50	2–8	3–6	0–2
JPEG (QF=50)	~50	10–20	8–15	0–5
Gaussian Noise	~50	5–15	3–10	1–5
Salt & Pepper	~50	3–10	2–8	0–3
Median Filter	~50	5–12	3–8	0–3
Rotation (5°)	~50	15–30	10–20	0–5
Cropping (10%)	~50	8–20	5–15	0–5
Scaling (0.5 $\times$)	~50	10–25	8–18	0–8

LSB watermarking fails under most attacks, with BER approaching 50%, equivalent to random guessing. DCT and DWT methods perform significantly better, while hybrid DCT-DWT consistently provides the strongest robustness.

8.3 Steganalysis Resistance

Statistical steganalysis methods attempt to detect hidden data through image anomalies. LSB embedding is highly vulnerable because it distorts least-significant bit-plane statistics, making detection possible through RS analysis and chi-square attack. Transform-domain methods are more resistant because watermark information is distributed across frequency coefficients rather than concentrated in pixel planes [12].

Modern CNN-based steganalysers such as SRNet and ZhuNet can detect LSB embedding with accuracies above 95% for single-bit modifications. In contrast, DCT and DWT watermarking at moderate strengths often remain below 60% detection accuracy. Content-adaptive watermarking offers the strongest resistance, with detection rates approaching random chance [7,15].

8.4 Payload Capacity Analysis

There is a fundamental trade-off between payload capacity, imperceptibility, and robustness. Table 2 compares representative payload capacities for a standard 512×512 colour image.

Table 7: Payload Capacity Comparison

Method	Bits per Pixel	Total Payload
LSB (1 LSB/channel)	3.0	786,432
LSB (2 LSB/channel)	6.0	1,572,864
DCT (1 bit/block)	0.016	4,096
DCT (mid-frequency)	0.69	180,224
DWT (1-level)	~0.5	~131,072

LSB provides the highest raw capacity, whereas transform-domain methods embed lower but significantly more robust payloads. For practical watermarking applications such as authentication codes, DCT and DWT capacities are sufficient.

8.5 Computational Complexity

LSB watermarking runs in $O(N)$ time, where N is the number of pixels, with negligible memory overhead. DCT watermarking typically runs in $O(N \log N)$ due to repeated block transforms. DWT watermarking requires approximately $O(NL)$ time, where L is the number of decomposition levels. For modern systems, these overheads are acceptable for AI-generated image pipelines.

9. Discussion

Implications for AI-Generated Image Watermarking

The comparative study leads to several practical conclusions:

- LSB watermarking is unsuitable for real-world social media sharing because JPEG recompression destroys embedded bits.
- DCT watermarking is highly suitable for JPEG-dominant channels because mid-frequency coefficients often survive compression at quality factors above 60.
- DWT watermarking is preferred in applications demanding high perceptual quality such as photography and medical imaging.
- Hybrid DCT-DWT methods provide the strongest overall robustness and are recommended for high-security scenarios [11].
- Diffusion-based regeneration attacks reveal limitations in classical watermarking systems, motivating multi-layer strategies combining watermarking, perceptual hashing, and provenance standards such as C2PA [1,2].

10. Conclusion

Invisible watermarking provides an effective mechanism for identifying and authenticating AI-generated images without introducing visible artifacts. Spatial-domain techniques such as LSB are simple to implement and offer high payload capacity, but they are generally vulnerable to compression, noise, and image processing attacks. In contrast, transform-domain techniques such as DCT and DWT offer greater robustness and are better suited for real-world deployment where images may be modified, compressed, or redistributed.

This comparative study shows that transform-domain watermarking is a stronger choice when robustness and reliability are priorities, while spatial-domain methods remain useful for lightweight or low-complexity applications. As AI-generated media continues to improve in realism, robust watermarking will remain an important tool for authenticity verification, intellectual property protection, and responsible AI content governance [4,3].

ACKNOWLEDGEMENT

- We express our sincere gratitude to AISSMS Institute of Information Technology, Pune, for providing the academic environment, resources, and encouragement necessary to carry out this research work.
- We are especially thankful to our respected guide Dr. Deeplaxmi Zingade for her valuable guidance, continuous support, technical insights, and motivation throughout every stage of this project. Her mentorship played a vital role in shaping the direction and successful completion of this study.
- We also extend our appreciation to the faculty members of the Department of Artificial Intelligence & Data Science for their suggestions and encouragement.

References

- [1] Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.
- [2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.
- [3] S. J. Nightingale and H. Farid, "AI-synthesized faces are indistinguishable from real faces and more trustworthy," *Proceedings of the National Academy of Sciences*, vol. 119, no. 8, e2120481119, 2022.
- [4] L. Verdoliva, "Media forensics and deepfakes: An overview," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 910–932, 2020.
- [5] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, "HiDDeN: Hiding data with deep networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 657–672, 2018.
- [6] Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker, *Digital Watermarking and Steganography*, 2nd ed. Morgan Kaufmann, 2007.
- [7] S. Katzenbeisser and F. A. P. Petitcolas, *Information Hiding: Techniques for Steganography and Digital Watermarking*. Artech House, 2000.
- [8] R. Panigrahi and N. Padhy, "An effective steganographic technique for hiding the image data using the LSB technique," *Cyber Security and Applications*, vol. 3, Art. no. 100069, 2025.
- [9] N. Saikrishna and M. G. Resmipriya, "An Invisible Logo Watermarking using Arnold Transform," *Procedia Computer Science*, vol. 93, pp. 808–815, 2016.
- [10] Y. Wang, Y. Luo, Z. Wang, and H. Pan, "A Hidden DCT-Based Invisible Watermarking Method for Low-Cost Hardware Implementations," *Electronics*, vol. 10, no. 12, 2021.
- [11] M. Chandra, "A DFT-DWT Domain Invisible Blind Watermarking Technique for Copyright Protection of Digital Images," in *Advances in Wavelet Theory and Their Applications in Engineering, Physics and Technology*, D. Baleanu, Ed. InTech, 2012.
- [12] P. C. Mandal, I. Mukherjee, G. Paul, and B. N. Chatterji, "Digital image steganography: A literature survey," *Information Sciences*, vol. 609, pp. 1451–1488, Sep. 2022.
- [13] H.-W. Tseng and H.-S. Leng, "A reversible modified least significant bit (LSB) matching revisited method," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 5, no. 1, pp. 1–8, 2009.
 - a. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with CLIP latents," *arXiv preprint arXiv:2204.06125*, 2022.
- [14] K. Ghazanfari, S. Ghaemmaghami, and S. R. Khosravi, "LSB++: An improvement to LSB+ steganography," in *TENCON 2011—2011 IEEE Region 10 Conference*, IEEE, 2011.