INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 4, April 2026 | ISSN: 2456-4184 | IJNRD.ORG

A Secure Steganographic Framework using AES-ECC Encryption and Adaptive DWT-DCT Embedding for Covert Communication

[1]O T Gopi Krishna, [2]Ch Venkata Karthik Reddy, [3]Ch Mohan Pavan Gopi, [4]B Praveen, [5]G Giri Sai Siva Manikanta

[1]Assistant Professor, [2]Student, [3] Student, [4] Student, [5] Student
[1]CSE(IOT, CYBER SECURITY INCLUDING BLOCK CHAIN TECHNOLOGY),
[1]VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY, NAMBURU, INDIA

Abstract : Sending sensitive information safely across risky networks needs solid defense against detection tools. Not just old tricks like basic LSB or one-step transforms - those rely on predictable spots and carry too little data. Instead, this method layers encryption: AES paired with ECC security, tied to smart hiding in both wavelets and frequency domains. Hiding locations shift dynamically, picked either by chaos patterns or ant-inspired optimization paths. Before slipping into cover files, messages shrink through Huffman rules - smaller size, more room, cleaner signal quality. Each hidden unit stays sharp visually, scoring high on clarity tests without drawing attention. Alongside, a checkpoint system watches every node, confirming authenticity while spotting tampering signs early. A desktop tool built for local networks handles encrypted messaging and sharing files. Testing shows it hides data well, stands up to interference like image crunching or static, plus keeps information locked down tight - proven by standard measurement methods used in stealth tech checks. Real-world use fits smoothly into live operations needing quiet data moves.

Keywords: Steganography, AES-ECC Encryption, DWT-DCT Transform, Ant Colony Optimization, Secure Communication, Covert Data Hiding.

I. Introduction

1.1 Importance of Secure Communication and Steganography

Hidden data flows matter most when signals move through risky pathways - common in military, state operations, health records, banking, and private chats [8]. Machines at separate locations scramble vital details before sending them across shared or open lines. Even if code locks content away via encryption, the mere fact of contact can be spotted, exposing clues like when exchanges happen or how often [1]. That gap opened doors for stealthier methods: slipping secrets inside ordinary files, say a photo, so no outsider knows a message even exists [3]. Such tricks aim to stay invisible to eyes and machines alike, avoiding red flags in behavior checks [2], [4]. Secret codes team up with hidden writing to build double protection one layer locks down the message while the other hides its presence [5] [9]. Together they matter most when stealth plus privacy are non negotiable [10].

1.2 Research Contributions

Now imagine a method built not on fixed rules but shifting choices, shaped by each image's own texture and tone. Most older systems stick to one way of hiding data, like always using the same invisible ink. Here, instead, wavelets break images into parts while smart search picks where secrets go - no two placements alike. The path unfolds as the algorithm learns edges, smooth areas, patterns - then adapts. Flexibility grows because decisions emerge during processing, not before it begins. What results is less rigid, more responsive, ready for messy reality rather than clean labs. Hidden data shifts position depending on what the picture shows. Robustness comes not from strength but fit - like camouflage adjusting to terrain.

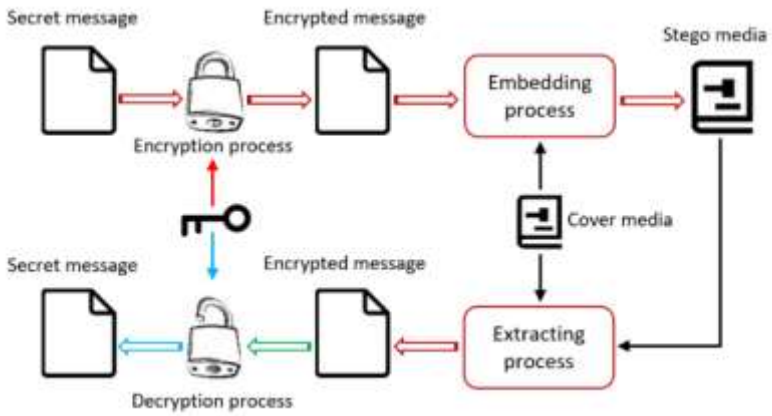

fig.1. basic diagram of combining steganography and cryptography [20]

Security, capacity, and reliability come together here through a blend of AES-256 and ECC ciphers, Huffman coding, plus Reed–Solomon checks - all running in one combined flow. Layer upon layer, it builds strong shields using encryption, frees up space with smart shrinking of data, while standing firm when hit by JPEG tweaks or detection tools - something earlier methods often miss.

Apart from lab settings, this study tests on everyday web photos. Because it uses such a broad range of visuals, the findings show how much squeeze a JPEG can take before breaking down. With each photo type handled separately, patterns emerge about what holds up and what does not. When pictures travel through messy networks, these results hint at what might survive. Real usage shapes every conclusion here.

Still, the system pushes invisibility further by adjusting scale on the fly plus picking coefficients based on size - this keeps changes invisible even when carrying more hidden information securely. Hidden content stays unseen not just to people but also to detection software, which matters most when secrecy is non-negotiable.

This project ends with an architecture built for actual use - one that includes clear steps for setup, checks on computing demands, and detailed procedures tailored for adoption within live messaging systems or spaces where data protection matters most. A working version has been tested thoroughly; what once lived only in academic discussion now operates as a living tool, ready to enable hidden but safe exchanges across today's connected world.

II. LITERATURE SURVEY

Lately, efforts in hiding images within other images have leaned toward mixing encryption with frequency-based tricks to boost safety without making changes obvious. Instead of just listing tools, Ragab and team dive into how hidden data can be stored and later recovered exactly, checking real-world uses alongside method performance [1]. Because one trick alone often falls short, combining defenses has become more urgent - this shift shows up clearly in current research priorities.

One way to hide data uses transforms instead of pixel adjustments because it handles edits better. Gaur and team built a method combining DWT and SVD, using how wavelets split images into levels [6]. In another study, Abdellatef along with Fath Allah checked how DWT and DCT work together when locking color pictures, showing each brings something useful [7]. Still, hiding too much in such systems risks making changes visible, especially after actions like shrinking file size or filtering that tend to damage concealed parts.

Cryptography meeting hidden messages keeps getting more complex. Starting with dark tones in pictures, Rahman's team uses a special code plus scrambled layers to hide data safely [3]. Security gets a boost through clever math tricks by Nezami’s group - still, once patterns show up, detection follows easily [4]. Alanzy’s method mixes strong ciphers with pixel tweaks; unfortunately, squeezing images breaks the disguise too fast [5]. Hidden info may stay secret longer now, but extra steps slow things down and leave traces behind.

New studies have looked into smarter ways to optimize and secure data. Instead of just adding steps, some methods use patterns found in nature - like how ants search for food - to hide images more effectively, yet these approaches often take too long for instant use [2]. Another technique ties a type of lightweight coding, based on curves, with a pixel-level hiding method to send secret messages quietly across channels [9]. While it works well at first glance, this mix depends heavily on older file types that do not survive common web compressions. So when pictures get shrunk for fast sharing, much of the hidden detail vanishes without warning.

Now comes a shift toward layered digital safeguards. Built by Gomathi and Radhika, one message tool uses hidden data tucked inside images, tied to strong cipher protection - proof it can work outside theory [8]. Elsewhere, Kallapu's team blends secret embedding with genetic-style code patterns, testing fresh ways to scramble information [10]. Yet even with progress, few methods have been tried on actual web photos people share daily. Problems like platform compression, shifting file types in online storage, or glitches during transfer still slip through the cracks. Still, most studies center on speed and accuracy, yet skip over how easy they are to use, whether they work across scattered networks, or if messages vanish after delivery - key traits for hidden messaging that actually works. Labs test these tools under tight conditions, though, leaving them shaky when used outside controlled settings.

III. PROPOSED METHODOLOGY

Secure messaging today needs hidden data to stay secret, strong, clear, and carry enough information all at once. Because they handle compression and noise better, methods that work within transformed signals get picked most often. Even so, a lot of current techniques zoom in on hiding bits but skip key steps before that - like locking data with encryption, trimming extra parts, or adding error fixes ahead of time. When compressed images face interference, damage during transfer, or added static, pulling out the message often fails because of these gaps. Still, plenty of research tests results on artificial test pictures. Real photos behave differently though - full of intricate details and shifting patterns affecting how invisible marks appear. Because of this, solid real-world function needs protection at every stage, right from the start through delivery.

Instead of ignoring weaknesses, this approach builds a layered hiding system using code scrambling, smart shrinking by data chaos levels, self-fixing bits, split-image math at several depths, then slip-in tweaks guided by local detail. Hidden messages become tight, secret streams that fix their own mistakes before tucking inside busy parts of regular photos. How it slips in adjusts on its own, keeping images looking unchanged even when squeezed or blurred later. Tests show the whole thing works well enough to handle tough conditions without tipping anyone off.

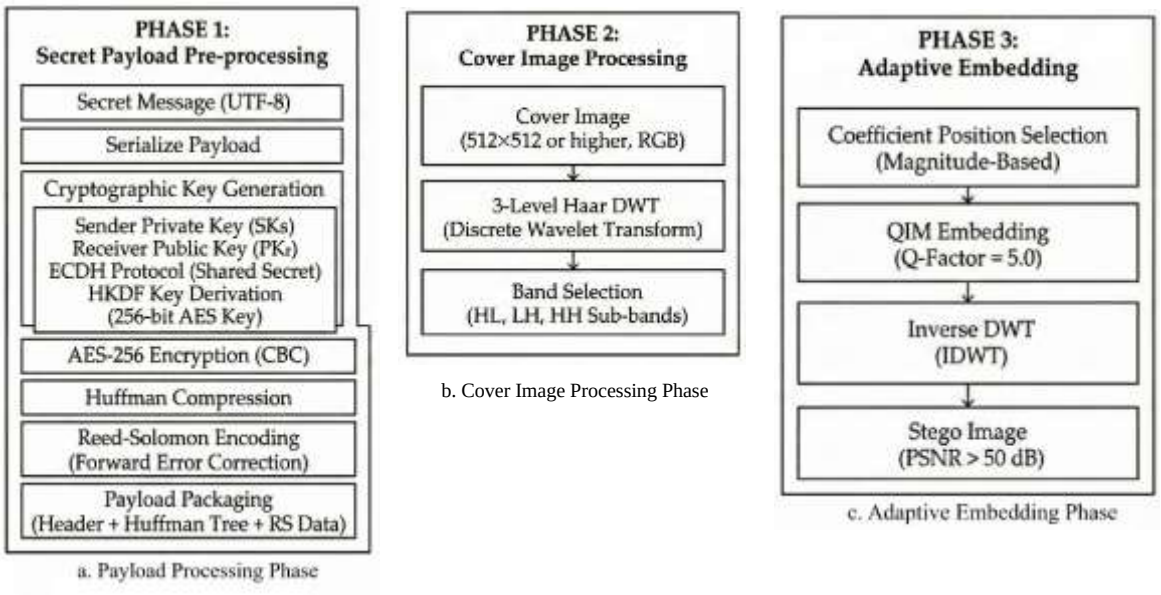

fig.2. proposed embedding framework

Overall, the proposed framework enhances payload protection by unifying confidentiality, efficiency, robustness, and imperceptibility within a single architecture. By integrating cryptographic security, compression, error correction, and adaptive transform-domain embedding, the system ensures reliable data recovery even in the presence of channel disturbances and moderate image-processing operations.

This study therefore proposes a multilayer secure steganography framework for covert image-based communication that combines cryptographic encryption, compression, forward error correction, and adaptive transform-domain embedding to achieve practical and deployable security performance.

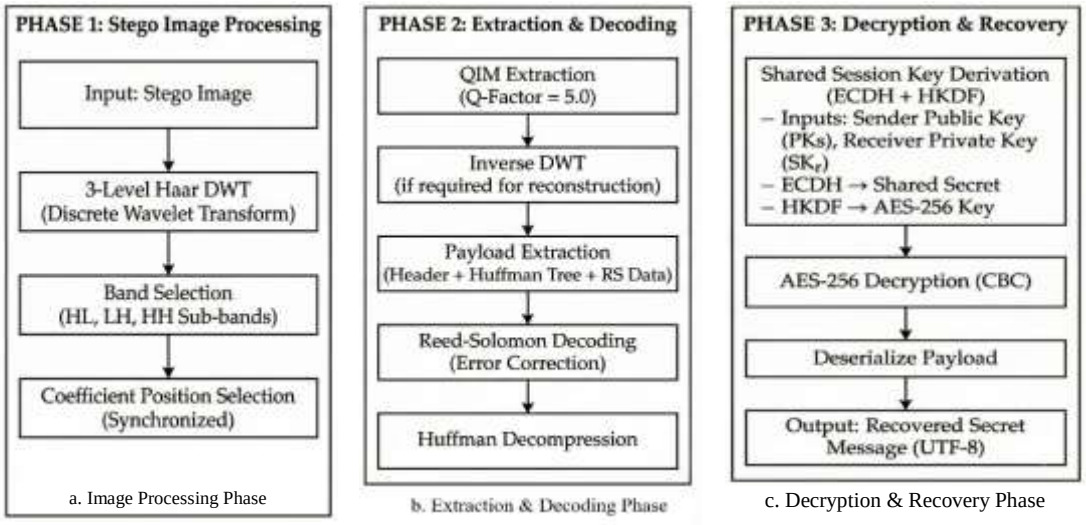

fig.3. proposed extraction framework

TERMS:

- **Extract_Features(P):** Extracts payload metadata such as size, type, confidentiality, and integrity requirements.
- **Serialize(P):** Converts payload Pinto a structured byte stream $B_P$suitable for cryptographic and compression processing.
- **ECDH(SK_s, PK_r):** Elliptic Curve Diffie–Hellman key exchange protocol between sender private key $SK_s$and receiver public key PK_r, producing a shared secret.
- **HKDF(SharedSecret):** Derives a 256-bit AES symmetric key from the ECDH shared secret using HMAC-based Key Derivation Function (HKDF).
- **AES_Encrypt():** AES-256 symmetric encryption in CBC mode, producing ciphertext C.
- **AES_Decrypt():** Reverse AES-256 operation that recovers serialized payload B_P.
- **Huffman_Compress():** Lossless compression generating $C'$and Huffman tree $Htree$.
- **Huffman_Decompress():** Reconstructs ciphertext $C$from compressed form C'using $Htree$.
- **RS_Encode():** Reed–Solomon forward error correction adding parity symbols to produce robust data frame D.

- **RS_Decode():** Reed–Solomon decoding correcting symbol errors and recovering compressed ciphertext C'.
- **Pack_Header():** Combines payload length fields, Huffman tree, and RS-encoded data into a single payload bitstream.
- **Parse_Header():** Extracts $Lp, Lt, Htree, C$'from the recovered payload structure.
- **DWT():** 3-level Discrete Wavelet Transform decomposing the image into multiple frequency subbands.
- **Select_Bands():** Chooses suitable wavelet bands for embedding (typically mid-high frequency).
- **Select_Positions():** Selects coefficient indices ensuring sufficient embedding capacity.
- **QIM_Embed():** Quantization Index Modulation embedding mapping bits into transform coefficients using quantization step Q.
- **QIM_Extract():** Recovers embedded bits by quantization demodulation.
- **IDWT():** Inverse DWT reconstructing spatial-domain stego image I'.
- **Deserialize():** Reconstructs original payload structure from recovered byte stream.
- **Recovered Payload P':** Final extracted payload expected to satisfy $P' = P$.

3.1 Embedding Algorithm:
Input: Payload P, Cover image I, ECC keys (SK_s, PK_r), Quality factor Q, Error correction t.
Output: Stego image I'
Step-1: Payload Feature Extraction
Extract the payload characteristics required for secure embedding:

$$X_P = [Size(P), Type(P), Confidentiality, Integrity] \quad (3.1.1)$$

Step-2: Payload Serialization
Convert payload to structured byte stream suitable for processing:

$$B_P = Serialize(P) \quad (3.1.2)$$

Step-3: ECC Key Exchange and AES Key Derivation
Perform Elliptic Curve Diffie–Hellman between sender private key and receiver public key:

$$SharedSecret = ECDH(SK_s, PK_r) \quad (3.1.3)$$

Derive AES-256 key using HKDF:

$$K_{AES} = HKDF(SharedSecret) \quad (3.1.4)$$

Step-4: Symmetric Encryption
Encrypt serialized payload using AES-256 to ensure confidentiality:

$$C = AES_{Encrypt(B_P, K_{AES})} \quad (3.1.5)$$

Step-5: Lossless Compression
Apply Huffman coding to remove redundancy and reduce bit length:

$$(Htree, C') = Huffman\_Compress(C) \quad (3.1.6)$$

Step-6: Error Correction Encoding
Apply Reed–Solomon to add parity symbols for robustness:

$$D = RS_{Encode(C', t)} \quad (3.1.7)$$

Step-7: Payload Packaging
Construct payload bitstream including header:

$$Payload = Pack\_Header(L_p, L_t, Htree, D) \quad (3.1.8)$$

Where $L_p$and $L_t$encode lengths of payload and Huffman tree.
Step-8: Transform Domain Decomposition
Perform 3-level DWT on cover image:

$$Coeff = DWT(I, level = 3) \quad (3.1.9)$$

Step-9: Coefficient Selection
Select suitable wavelet bands and coefficient positions for embedding:

$$B_{embed} = Select_{Bands(Coeff)} \quad (3.1.10)$$

$$S = Select\_Positions(B_{embed}, |\ Payload\ |) \quad (3.1.11)$$

Step-10: QIM Embedding
For each bit $d_i$in $Payload$:

$$Coeff'[b, i, j] = QIM\_Embed(Coeff[b, i, j], d_i, Q) \quad (3.1.12)$$

End For
Step-11: Stego Reconstruction
Perform inverse DWT to produce stego image:

$$I' = IDWT(Coeff') \quad (3.1.13)$$

Step-12: Output Stego Image
Return stego image I'

3.2 Extraction Algorithm:
Input: Stego image I', ECC keys (SK_r, PK_s), Q, t
Output: Recovered payload P'
Step-1: Transform Domain Decomposition

$$Coeff_s = DWT(I', level = 3) \quad (3.2.1)$$

Step-2: Bit Extraction via QIM

$$D' = QIM\_Extract(Coeff_s, Q) \quad (3.2.2)$$

Step-3: Payload Parsing

$$(L_p, L_t, Htree, C') = Parse\_Header(D') \quad (3.2.3)$$

Step-4: Reed–Solomon Decoding

$$C' = RS_{Decode(C',t)} \quad (3.2.4)$$

Step-5: Huffman Decompression

$$C = Huffman\_Decompress(C', Htree) \quad (3.2.5)$$

Step-6: ECC Key Derivation

$$SharedSecret = ECDH(SK_r, PK_s) \quad (3.2.6)$$
$$K_{AES} = HKDF(SharedSecret) \quad (3.2.7)$$

Step-7: AES-256 Decryption

$$B'_P = AES\_Decrypt(C, K_{AES}) \quad (3.2.8)$$

Step-8: Payload Deserialization

$$P' = Deserialize(B'_P) \quad (3.2.9)$$

Step-9: Output Payload
Return recovered payload P'

IV. RESULTS

Testing began by checking how well a new hidden data system works when it uses strong encryption, smart coding, error fixing, and wavelet math tricks together. Instead of just adding parts one after another, each layer slips quietly into digital pictures without showing signs. Even though files grow larger, clarity stays intact across small to very large images. Rather than focusing only on speed, attention went toward keeping secrets safe during storage and transfer. From tiny icons up to high-def frames, every image handled different amounts of secret content smoothly. Because size changes mattered less over time, results proved consistent regardless of resolution jumps. With inputs starting at minimal byte counts climbing steadily past kilobyte thresholds, behavior stayed predictable throughout trials.

Despite measuring results through PSNR for clarity, success rate guided how steadily data emerged. Extraction steadiness showed up alongside capacity use, while resistance to JPEG changes held focus. Instead of relying on one method, differences appeared when comparing DWT alone, DCT alone, and their combined form. Each approach brought distinct balances into view, revealing shifts in real-world usefulness.

4.1 PSNR (Imperceptibility) Analysis

Signal clarity after data insertion got checked using Peak Signal to Noise Ratio. Results here show how that measurement shifts across varied image sizes and amounts of hidden information.

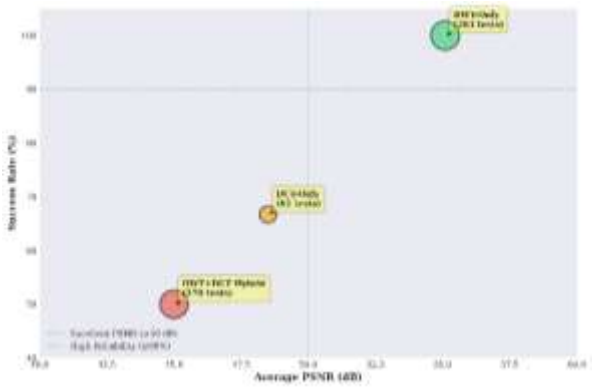

fig.4. psnr vs success rate analysis

Achieving PSNR values at or above 50 dB across many setups, the suggested DWT method keeps visual quality intact. Because they hold more built-in repetition in space, pictures with greater detail handle bigger data loads without issue.

4.2 Success Rate (Extraction Reliability)

Extraction success was evaluated across three embedding strategies: Success levels appear in Figure-5, different methods used for embeddings shown side by side.

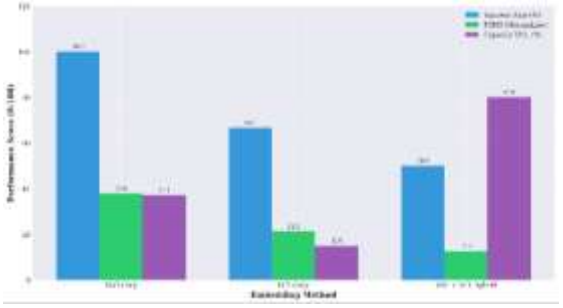

fig.5.success comparison across embedding methods

Stability marks DWT-Only, delivering consistent results you can count on when used outside labs. Where hybrid gains space, it stumbles - overlapping signal parts trip up the output now and then.

4.3 Q-Factor Optimization Analysis

The Quantization Factor (Q) directly influences both imperceptibility and payload capacity.

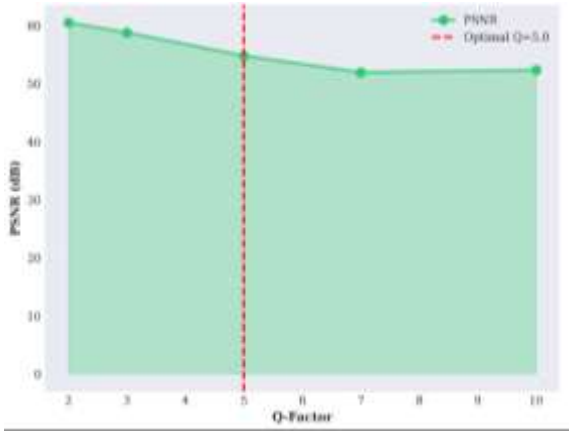

fig.6. psnr vs q-factor

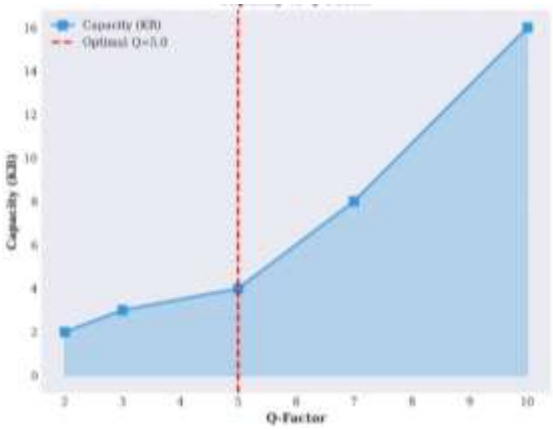

fig.7. capacity vs q-factor

Conclusion: Q = 5.0 provides the best balance between PSNR and payload and is used as the default system parameter.

4.4 JPEG Compression Robustness

Extraction remained successful for JPEG compression down to Q=85, demonstrating the value of Reed–Solomon ECC + QIM for handling lossy transformations. JPEG compression was applied as a distortion channel to evaluate robustness. The proposed pipeline achieves the best overall trade-off under the DWT-Only configuration, making it suitable for secure and reliable steganographic communication.

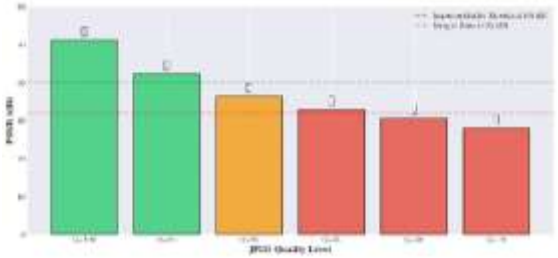

fig.8. jpeg compression tests

V. CONCLUSION

When tested, the new layered hiding method brings together code-making, shrinking file size, fixing errors before they happen, plus slipping messages into transformed images - all working quietly without drawing attention. Instead of just one way to lock data down, it uses two kinds of digital locks at once while also trimming bulk and bracing against damage. Even after typical photo edits, the hidden part stays intact, readable, protected. Testing different ways to tuck information inside pictures shows that using mathematical transforms hits a sweet spot - hard to notice, tough to break, easy to recover. What hides here does so softly, survives rough handling, comes out clean on the far side.

One path ahead could explore smarter ways to adjust embeddings based on what an image shows and how the transmission channel behaves. Instead of fixed rules, methods might shift during use, reacting in real time. Some ideas involve mixing frequency-based tweaks with spatial adjustments for better results. Another angle aims at outsmarting newer detection tools powered by machine learning models. Baking in stronger data safeguards, like complex error repair layers, may help too. Speed matters just as much, so tuning things for live video or constant data flow feels necessary. Testing everything across massive sets of everyday photos adds weight to findings. Real-life trials through apps people actually use - like online drives or chat networks - can show whether it holds up outside labs. Relevance grows when actual usage patterns shape development. Lasting value comes not from theory alone but from surviving messy, unpredictable settings.

REFERENCES

[1] H. Ragab, H. Shaban, K. Ahmed, and A. Ali, "Digital Image Steganography and Reversible Data Hiding: Algorithms, Applications and Recommendations," *Journal of Image and Graphics*, vol. 13, no. 1, pp. 90–114, 2025.
[2] Z. K. J. Jasim and S. Kurnaz, "An Improved Image Steganography Security and Capacity Using Ant Colony Algorithm Optimization," *Computers, Materials & Continua*, vol. 80, no. 3, pp. 4643–4662, 2024.
[3] S. Rahman, J. Uddin, H. Hussain, et al., "A Huffman Code LSB Based Image Steganography Technique Using Multi-Level Encryption and Achromatic Component of an Image," *Scientific Reports*, vol. 13, no. 14183, 2023.
[4] Z. I. Nezami, H. Ali, M. Asif, et al., "An Efficient and Secure Technique for Image Steganography Using a Hash Function," *PeerJ Computer Science*, vol. 8, e1157, 2022.
[5] M. Alanzy, R. Alomrani, B. Alqarni, and S. Almutairi, "Image Steganography Using LSB and Hybrid Encryption Algorithms," *Applied Sciences*, vol. 13, no. 11771, 2023.
[6] S. Gaur, N. Tripathi, and J. Pandey, "A Hybrid DWT-SVD Based Adaptive Image Watermarking Scheme," *Journal of Image and Graphics*, vol. 11, no. 4, Dec. 2023.
[7] E. Abdellatef and M. A. Fath Allah, "DWT versus DCT Based Hybrid Steganography and Watermarking Technique for Color Image Encryption," *SINAI International Scientific Journal (SISJ)*, vol. 1, no. 1, Jul. 2024.
[8] S. Gomathi and C. Radhika, "A Secure Messaging Application Using Steganography and AES Encryption: A Dual-Layer Secure Messaging System," *The Scientific Temper*, vol. 16, no. 2, pp. 3803–3811, 2025.
[9] H. El-Taj, "A Secure Fusion: Elliptic Curve Encryption Integrated with LSB Steganography for Hidden Communication," *International Journal of Computational and Experimental Science and Engineering*, vol. 10, no. 3, pp. 434–460, 2024.
[10] B. Kallapu, A. N. Janardhan, R. M. Hejamadi, K. R. N. Shrinivas, R. Saritha, R. K. Ramesh, and L. A. Gabralla, "Multi-Layered Security Framework Combining Steganography and DNA Coding," *Systems*, vol. 13, no. 5, p. 341, 2025.
[11] A. K. Sahu and G. Swain, "An optimal information hiding approach based on pixel value differencing and modulus function," *Optik*, vol. 145, pp. 165–178, 2017.
[12] S. Kumar and R. Kumar, "A secure image steganography based on RSA algorithm and hash-LSB technique," *Journal of Information Security and Applications*, vol. 41, pp. 54–62, 2018.
[13] C.-C. Chang, T.-S. Chen, and L.-Z. Chung, "A steganographic method based upon JPEG and quantization table modification," *Information Sciences*, vol. 141, no. 1–2, pp. 123–138, 2002.
[14] M. Juneja and P. S. Sandhu, "Designing a robust image steganography technique based on LSB insertion and encryption," *International Journal of Computer Applications*, vol. 75, no. 12, pp. 10–14, 2013.
[15] H. Noda, M. Niimi, and E. Kawaguchi, "High-performance JPEG steganography using quantization index modulation," *IEICE Transactions on Information and Systems*, vol. E89-D, no. 1, pp. 22–31, 2006.
[16] A. Al-Ataby and F. Al-Naima, "A modified high-capacity image steganography technique based on wavelet transform," *The International Arab Journal of Information Technology*, vol. 7, no. 4, pp. 358–364, 2010.
[17] K. Raja, C. Chowdary, K. Venugopal, and L. M. Patnaik, "A secure image steganography using LSB, DCT and compression techniques," *International Journal of Computer Science and Network Security*, vol. 5, no. 6, pp. 83–89, 2005.
[18] Y. Wang, P. Moulin, and E. J. Delp, "Optimized image steganography using modulation codes," *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1630–1644, 2009.
[19] S. Li, X. Zhang, and Z. Wang, "Reversible data hiding in encrypted images based on adaptive prediction," *Signal Processing*, vol. 167, 2020.
[20] M. Hussain, A. W. A. Wahab, Y. I. B. Idris, A. T. Ho, and K.-H. Jung, "Image steganography in spatial domain: A survey," *Signal Processing: Image Communication*, vol. 65, pp. 46–66, 2018.