Frontiers in Computer Science and Artificial Intelligence
ISSN: 2798-8048
DOI: 10.32996/fcsai
Journal Homepage: www.al-kindipublisher.com/index.php/fcsai

Multilevel Steganography of Unicode Using Modified Lempel–Ziv–Welch Algorithm and Least Significant Bit Algorithm

Prottasha Modak[1] ✉[1], Ashoke Kumar Modak[2], Madina Afrin[3], Pranto Halder[4], Shah Farhan Sadik[5], Niloy Hasan[6], Md. Manik Hossain[7], Shabbir Hossain Shaptak[8]

[1] *Department of Computer Science and Engineering, Anwer Khan Modern University, Dhaka, Bangladesh.*

[2] *Department of Orthopaedic Surgery, Kurmitola Multicare Specialized Hospital, Dhaka, Bangladesh.*

[3] *Department of Computer Science and Engineering, Khulna University of Engineering & Technology (KUET), Khulna, Bangladesh.*

[4] *Department of Computer Science and Engineering, Faculty of Science and Engineering, Teesta University, Rangpur 5404, Bangladesh.*

[5] *Department of Computer Science and Engineering, Faculty of Science and Engineering, Teesta University, Rangpur 5404, Bangladesh.*

[6] *Department of Computer Science and Engineering, Faculty of Science and Engineering, Teesta University, Rangpur 5404, Bangladesh.*

[7] *Department of Computer Science and Engineering, Faculty of Science and Engineering, Teesta University, Rangpur 5404, Bangladesh.*

[8] *Department of Computer Science and Engineering, Faculty of Science and Engineering, Teesta University, Rangpur 5404, Bangladesh.*

Corresponding Author: Prottasha Modak**, E-mail:** prottashamodak.akmu@gmail.com

ABSTRACT

Secure digital communication requires methods that protect not only the meaning of a message but, where appropriate, the fact that a message is being transmitted. Steganography addresses the latter problem by concealing payload data inside an ordinary carrier, while lossless compression can reduce the number of bits that must be embedded. This article examines a multilevel image-steganography workflow combining dictionary-based Lempel–Ziv–Welch (LZW) compression, a modified LZW representation and least significant bit (LSB) substitution. The central objective is to assess whether preprocessing a text payload can reduce its embedding burden while allowing exact recovery after extraction. In the described workflow, a text message and cover image are processed, compressed data are prepared for embedding, payload bits are inserted into image pixel values and the receiver applies inverse LSB extraction followed by decompression. The source thesis reports a worked example in which the 13-character string "mynameismeena" occupies 104 bits in an 8-bit-per-character representation. Standard LZW encoding is reported as 99 bits, while the modified representation is reported as 84 bits. These values correspond to approximately 4.8% and 19.2% reductions, respectively, relative to 104 bits; they do not support the source's statement that 84 bits represents an 80% reduction. The thesis also reports an image-size change from 81 KB to 13 KB and a combined text-plus-image representation change from 185 to 112 units, although the precise measurement conventions are not fully documented. The results therefore indicate the potential value of compression before embedding, but they do not establish security against steganalysis or robustness under image transformations. The article clarifies the workflow, presents the available results in tables, identifies methodological limitations and outlines a reproducible evaluation framework for Unicode payloads, image fidelity, capacity, execution time and extraction accuracy.

KEYWORDS

Image Steganography; Unicode; LZW Compression; Modified LZW; Least Significant Bit; Lossless Compression; Payload Capacity; Steganalysis.

ARTICLE INFORMATION

ACCEPTED: 18 October 2026 **PUBLISHED:** 21 November 2026 **DOI:** 10.32996/fcsai.2026.5.12.4

Copyright: © 2026 the Author(s). This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) 4.0 license (https://creativecommons.org/licenses/by/4.0/). Published by Al-Kindi Centre for Research and Development, London, United Kingdom.

1. Introduction

The growth of digital communication has increased the need to protect sensitive information while it is stored, transmitted and received. Cryptography transforms a readable message into ciphertext so that an unauthorized party cannot readily understand its contents. Steganography has a different immediate objective: it attempts to conceal the presence of the message by embedding it in an apparently ordinary carrier such as an image, audio recording, video or text document. The two approaches can be combined, because encryption can protect the payload if concealment fails, while steganography may reduce the attention that an overt encrypted message attracts. However, neither method alone guarantees complete security; the result depends on the threat model, implementation, key management, carrier choice and communication channel.[1,2]

Image steganography is widely studied because digital images contain large numbers of pixel values and often exhibit spatial redundancy. In a typical 24-bit RGB image, each pixel is represented by red, green and blue channel values, commonly eight bits per channel. Least significant bit (LSB) substitution embeds payload bits by replacing selected low-order bits in pixel components. Since changing a channel's least significant bit generally changes its numerical value by no more than one, the visual difference may be difficult to notice under low payload conditions. Yet simple LSB substitution has important limitations: statistical patterns may reveal embedding and image resizing, filtering, format conversion or lossy compression may damage hidden bits. The carrier format and subsequent processing therefore matter as much as the embedding operation itself.[3,4,5]

Compression can reduce the quantity of data that must be hidden. Lempel–Ziv–Welch (LZW) is a dictionary-based, lossless algorithm that builds references to recurring strings during encoding. Instead of transmitting every repeated sequence as individual characters, the encoder emits dictionary codes; the decoder reconstructs the same dictionary from the code stream. This property is useful for text and for certain image formats, although the actual compression ratio depends on the input, code width, dictionary rules and implementation. A short or high-entropy message may expand rather than shrink. The term "modified LZW" can refer to different designs, so an implementation must specify its codeword format, dictionary initialization, code growth, termination and decoder behavior before comparisons can be reproduced.[6,7]

The supplied thesis proposes combining compression with image-based hiding with particular emphasis on text compression, image compression, LSB embedding and inverse operations for recovery. Its reported example compares a standard LZW encoding with a modified dictionary representation for the string "mynameismeena." It also presents a sample image-size reduction and a combined text-and-image comparison. The motivation is reasonable: if compression reduces the payload length, fewer carrier bits may need to be changed, potentially improving the relationship between payload size and visual fidelity. Nevertheless, compression and steganography solve different problems. Compression is not encryption and a compressed message is not inherently confidential. Likewise, low visible distortion does not establish resistance to statistical detection.[1,3]

A further issue is Unicode. A character in Unicode is not necessarily stored in one byte. UTF-8 uses one to four bytes per Unicode scalar value, while UTF-16 commonly uses one or two 16-bit code units. A fair evaluation must preserve the original byte sequence and specify the encoding used before compression. A demonstration based on English ASCII characters cannot, by itself, establish performance for Bangla, Arabic, emoji or mixed-script messages. Dictionary-based compression may exploit repeated byte sequences across these texts, but performance must be measured rather than assumed.

This article organizes the proposed approach as a reproducible pipeline and critically interprets the quantitative information available in the thesis. The objectives are to: (i) explain the relationship between compression and LSB embedding; (ii) describe a sender–receiver workflow; (iii) report and recalculate the source's example measurements; (iv) distinguish observed source results from claims that remain unverified; and (v) propose evaluation criteria for future Unicode-aware testing. The contribution is therefore a transparent research-style reconstruction and analysis of the documented method, not a claim that new experiments were conducted for this manuscript.

1.1 Research questions and objectives

The article addresses three questions. First, does the reported modified dictionary representation use fewer bits than the standard LZW example for the selected short string? Second, what can the reported image and combined-size measurements legitimately show about compression before hiding? Third, what evidence would be needed to support stronger claims about imperceptibility, security, robustness and Unicode compatibility? The primary objective is to analyze the documented workflow without treating compression as encryption or interpreting a single worked example as a general benchmark.

2. Related work and technical background

Steganographic systems are commonly evaluated against three competing properties: capacity, imperceptibility and robustness. Capacity describes how much payload can be carried, often measured in bits per pixel (bpp), bits per channel or payload bytes. Imperceptibility concerns the degree to which the stego image differs perceptually or statistically from the cover. Robustness concerns whether hidden information survives operations such as recompression, cropping, filtering and transmission. A method can perform well on one dimension and poorly on another. Spatial-domain methods such as direct LSB substitution are relatively simple and can offer useful capacity, but they are typically fragile under image processing and may leave detectable statistical traces. Transform-domain methods may survive selected transformations better, but they can be more complex and may offer lower payload capacity.[8,9]

LZW is an adaptive dictionary coder. In a conventional byte-oriented implementation, the initial dictionary contains entries for all possible single-byte values and longer strings are added as they are encountered. During decoding, the same additions are reconstructed from the received codes, avoiding the need to transmit the entire dictionary. The compression gain depends on repeated sequences and code width. When the input is short, the overhead of code representation can reduce the advantage; for a longer message containing repeated words or phrases, the algorithm may perform differently. A modified LZW method may seek to reduce code lengths by representing a dictionary reference together with a symbol or by using variable-width indexes. Its superiority should be established on identical data and with all overhead included. [6,7]

For image payload embedding, the carrier is represented as a sequence of channel bytes. The binary payload must include enough framing information for the receiver to identify the message length and, where applicable, encoding, compression mode and integrity check. A basic LSB encoder replaces a carrier bit with a payload bit. The decoder reads those positions in the same order. If the system uses a secret key to pseudo-randomize positions, both parties need the same key and a specified key-to-position mapping. A key is not a substitute for encryption and an undocumented "random" process cannot be reproduced or analyzed. The supplied thesis describes LSB hiding and inverse LSB extraction but does not document a complete cryptographic key protocol or a security analysis. [3,4,5]

3. Methods

This study uses a document-based analytical method. The primary source is the supplied 2019 undergraduate thesis on multilevel steganography using modified LZW and LSB. The method and quantitative examples were extracted from its algorithm description and result-analysis chapter. No new experiments using images or text corpora and no software benchmarks were conducted for this article. Therefore, the results below are a critical reconstruction of the reported examples rather than a new experimental validation. Where the thesis does not define a measurement unit or experimental protocol, that limitation is stated rather than resolved through assumption.

3.1 Proposed sender workflow

The sender begins with a text message and a cover image. First, the text is converted into a defined byte representation; for Unicode data, the encoding (for example, UTF-8) must be recorded. Second, the byte sequence is compressed with either conventional LZW or the specified modified variant. Third, the cover image is loaded into a lossless pixel representation and the payload is framed with a length field and optional integrity metadata. Fourth, the compressed payload is converted to a bitstream and embedded in selected pixel-channel LSBs. The output is saved as a stego image in a format that preserves the altered pixel values. If image compression is performed as part of the workflow, it must be clarified whether it is lossless and whether the embedding occurs before or after that compression: lossy compression after embedding can destroy the payload.

3.2 Proposed receiver workflow

The receiver reads the stego image without resizing or color conversion, extracts payload bits from the same pixel positions, parses the framing metadata and reconstructs the compressed byte stream. The matching decompressor then restores the original byte sequence. Finally, the receiver decodes the bytes using the agreed character encoding. The correct success criterion is byte-for-byte equality between the original payload and recovered payload, not merely visual similarity of displayed text. An implementation should also reject malformed lengths, incomplete payloads, invalid dictionary codes and failed integrity checks. The source describes inverse LSB and inverse LZW operations, but does not provide all protocol details needed for independent replication.

3.3 Evaluation measures

For payload compression, the compression ratio is defined here as encoded size divided by original size and percentage size reduction is defined as 100 × (1 − encoded size/original size). A ratio below 1 indicates size reduction; a ratio above 1 indicates expansion. For image fidelity, mean squared error (MSE) between the cover and stego images can be calculated over all pixel channels, while peak signal-to-noise ratio (PSNR) is calculated from MSE and the maximum possible pixel value. For 8-bit channels, the peak value is 255. Higher PSNR generally indicates lower numerical distortion, but it does not prove undetectability. Payload capacity should be reported in bits and bpp. Runtime should be measured under a documented environment and repeated over multiple trials. Unicode recovery should be tested across representative scripts and exact byte equality should be checked.

3.4 Scope and assumptions

The reported source examples use a 13-character English string and an image-size example identified as 81 KB before compression and 13 KB after compression. The source also mentions an execution time of 478.717609 seconds for an image-compression output, but does not supply enough information about hardware, image dimensions, software environment, repeated runs or timing boundaries to generalize that value. The article treats these figures as source-reported observations. It does not assume that "KB" and "bits" are interchangeable; the original tables mix labels and units, so they are reproduced with explanatory notes and recalculated ratios where possible.

4. Results

The source's reported results cover a worked text example, a modified dictionary representation, an image-size comparison and a combined text-plus-image comparison. The values are useful as demonstrations of the proposed sequence, but they should not

be interpreted as statistically representative performance results. The following tables preserve the source values and add transparent arithmetic based on the stated quantities.

4.1 Standard LZW text example

The thesis uses the string "mynameismeena," described as 13 characters. At eight bits per character the uncompressed size is 13 × 8 = 104 bits. The reported LZW output contains 11 codes represented at nine bits each, giving 99 bits. This is a reduction of five bits. Under the definition used in this article, the encoded-to-original ratio is 99/104 = 0.952 and the percentage reduction is approximately 4.8%. The example shows exact recovery of the original text in the described encoding–decoding demonstration. It does not establish that LZW will always compress short text: code widths and dictionary behavior are implementation-dependent and short messages may yield little gain.

Measure	Source-reported value	Recalculated interpretation
Input string	mynameismeena	13 characters as described
Uncompressed representation	13 × 8 = 104 bits	104 bits under the 8-bit character assumption
LZW encoded output	11 codes × 9 bits = 99 bits	99 bits
Encoded/original ratio	0.95	99/104 ≈ 0.952
Size reduction	Not explicitly calculated	(104−99)/104 × 100 ≈ 4.8%
Recovery	Reported as identical to original	A successful worked example; not a corpus-wide result

4.2 Modified LZW example

The thesis's modified representation expresses the dictionary output as a sequence of index–character pairs. For the same input string, the table lists dictionary entries including M, Y, N, A, ME, I, S, MEE and NA. The source then reports an encoded length of 84 bits, compared with the 104-bit unencoded representation. The corresponding ratio is 84/104 ≈ 0.808 and the reduction is 20/104 × 100 ≈ 19.2%. Thus, the figure supports a reduction of about one-fifth for this example, not an 80% reduction. The source's calculation appears to confuse the percentage of original size remaining with the percentage saved. A fair comparison also requires accounting for all framing, dictionary and codeword overhead; the available description does not fully specify whether all such overhead is included.

Measure	Standard LZW	Modified representation
Unencoded input	104 bits	104 bits
Reported encoded size	99 bits	84 bits
Encoded/original ratio	99/104 ≈ 0.952	84/104 ≈ 0.808
Size reduction	≈ 4.8%	≈ 19.2%
Difference in encoded size	—	15 bits fewer than the reported LZW output
Interpretation	Small reduction in this example	Better result for this one example; general advantage unproven

4.3 Image compression example

The thesis reports that an image called "Meenaa" changes from 81 KB to 13 KB after LZW compression. The table labels the values as "Bits," but the surrounding description identifies them as kilobytes. To avoid falsely changing the unit, this article reports them as source-described size units and notes that the likely intended unit is KB. Based on the numerical values, 13/81 ≈ 0.1605, which means the compressed file is approximately 16.0% of its original size, corresponding to an 84.0% reduction in file size. This is a substantial reported reduction but it cannot be independently verified from the textual description alone. Image compressibility depends on pixel data, image format, color depth and the exact encoder. The source's stated runtime of 478.717609 seconds is retained as reported, but cannot be compared fairly without implementation and hardware details.

Image measure	Source-reported value	Calculated interpretation
Original image size	81 KB (table heading says bits)	Unit inconsistency in source
Compressed image size	13 KB (table heading says bits)	Unit inconsistency in source
Encoded/original ratio	0.160	13/81 ≈ 0.1605
Size reduction	Not explicitly stated	Approximately 84.0% if both sizes are in KB
Reported execution time	478.717609 seconds	Environment and timing protocol not specified

4.4 Combined text and image comparison

The source compares 104 units for original text and 81 units for the original image, giving a combined total of 185 units. The compressed comparison lists 99 for text and 13 for image, totaling 112 units. Using these stated figures, 112/185 ≈ 0.605, meaning the combined encoded representation is approximately 60.5% of the original total. The reduction is therefore approximately 39.5%. The thesis table marks the ratio as "60%," which is closer to the percentage remaining than the percentage saved. This combined comparison mixes text bits with image file sizes as presented in the source and is not dimensionally valid unless the units are normalized. It should therefore be read as an illustrative comparison, not as a rigorous overall compression measurement.

Component	Original value	Compressed value	Comment
Text	104 bits	99 bits	Reported standard LZW example
Image	81 KB (likely)	13 KB (likely)	Table incorrectly labels values as bits
Combined numerical total	185	112	Arithmetic based on the source's displayed values
Compressed/original ratio	—	112/185 ≈ 0.605	About 60.5% remains
Reduction	—	≈ 39.5%	Only meaningful if units are compatible

4.5 Embedding and extraction

The thesis reports that compressed text was hidden in a compressed image and that inverse LSB extraction followed by inverse LZW recovered "mynameismeena." This is evidence of a successful illustrative round trip in the reported demonstration. However, the source does not provide a machine-readable cover image, stego image, payload bit count, per-channel embedding map, PSNR/MSE table for this exact example or tests after common image transformations. Consequently, no quantitative claim about imperceptibility, capacity, detection resistance or robustness can be drawn from the extraction example alone.

Outcome	Evidence available in the source	What can be concluded
Exact text recovery	Recovered string is reported as mynameismeena	Successful example is reported
Visual quality	Qualitative statements that LSB changes are small	No verified numerical fidelity result for this exact test
Steganalysis resistance	No detection experiment reported for the described example	Not established
Robustness	No systematic recompression/cropping/filtering tests	Not established
Unicode coverage	Example uses English ASCII text	Unicode-wide performance not established

5. Discussion

The purpose of the proposed system is to reduce the payload before embedding and then recover the original message through inverse operations. The source's worked text example offers limited support for the first part of this purpose: the modified representation is reported as 84 bits compared with 99 bits for standard LZW and 104 bits for the unencoded string. Recalculation shows that the modified output is about 19.2% smaller than the unencoded representation, while the standard LZW output is about 4.8% smaller. For this specific example, the modified representation therefore has a lower reported encoded size. However, the result is based on a single short English string and cannot establish general superiority across languages, file lengths or content distributions.[4,5]

A critical methodological issue is the distinction between compression ratio and percentage reduction. An encoded-to-original ratio of 0.808 means that approximately 80.8% of the original size remains; it does not mean that 80% has been saved. The actual reduction is approximately 19.2%. Similarly, the combined numerical comparison of 112 against 185 means about 60.5% of the original size remains and a reduction of approximately 39.5% is achieved, assuming the values share a common unit. Since the source combines text bit counts and image-size values that appear to be KB, the total is not a valid physical quantity without normalization. These corrections do not negate the idea of preprocessing a payload but they are necessary for a defensible interpretation of the results.

The second purpose is to conceal the message in an image while maintaining visual quality. LSB substitution is attractive because changing the least significant bit typically introduces only a small numerical difference in a channel value. Nevertheless, the perceptual impact depends on the number and distribution of modified bits, the texture and content of the cover image and the embedding strategy. A high payload can introduce detectable patterns even if a person does not notice a visible difference. The source's qualitative description of the stego image is therefore not sufficient evidence of imperceptibility. A rigorous assessment

should calculate MSE and PSNR for the cover and stego images and supplement these metrics with statistical steganalysis or a documented detection experiment.[6,8,10]

The third purpose is reliable extraction. The source states that inverse LSB and inverse LZW operations recover the original text in the example. Exact round-trip recovery is essential, but a robust protocol must specify the order of payload bits, image channel traversal, termination length, codeword interpretation and character encoding. Without a payload length field or another unambiguous delimiter, a receiver may not know where the compressed message ends. Without integrity protection, accidental corruption may produce a plausible but incorrect decoded result. The system should verify equality at the byte level and should detect truncated or malformed data rather than silently outputting corrupted text.[11,12]

Unicode introduces an additional requirement that is central to the article's title. The source's demonstrated string contains only basic Latin characters, which are compatible with a single-byte ASCII representation. Unicode text must first be serialized into a specified encoding. UTF-8 is a practical choice because it represents a wide range of scripts while preserving text as bytes; however, the number of bytes is not necessarily equal to the number of visible characters. Compression ratios should therefore be computed from the actual encoded byte length, not from a general assumption of eight bits per character. A valid evaluation should include Bangla and other scripts, combining marks, emoji, mixed-script messages and normalization-sensitive text. Recovered bytes should be compared before any normalization that could mask errors.

The reported image compression result also needs careful interpretation. LZW can be effective for some image data and file formats, but it is not a universal compressor that guarantees smaller outputs for every image. The observed reduction from 81 KB to 13 KB may depend on the image's redundancy and the exact representation. More importantly, compression and embedding order must be explicitly designed. If the cover image is converted into a compressed file and then pixel-level LSB embedding is attempted on the compressed bitstream, the algorithm is no longer directly operating on pixel-channel values unless the image is first decoded to a pixel array. A sound implementation should distinguish file compression from pixel-domain embedding and preserve a lossless carrier representation after embedding. Saving a modified pixel array as JPEG may alter the embedded bits because JPEG is generally lossy.[13,14]

Security claims should be bounded by a clear threat model. LSB embedding by itself does not encrypt the payload and compression does not provide confidentiality. A person who suspects the presence of hidden data may use statistical tests or compare files to detect modifications. If confidentiality is required, the payload should be encrypted with a modern authenticated-encryption scheme before embedding; the steganographic layer then serves as concealment rather than as the sole security mechanism. If a secret key is used to select embedding locations, the key-generation and position-selection methods should be specified and the system should avoid describing a non-cryptographic pseudo-random sequence as encryption.[1,6]

Finally, the reported runtime of 478.717609 seconds cannot establish that the method is fast or slow without information about the hardware, software, image dimensions and exact timed operation. It may represent a complete execution, a plotting operation or another implementation-specific process; the source does not clarify this. A reproducible benchmark should separate text compression, image processing, embedding, extraction and decompression times; report machine specifications and software versions; and run multiple trials. Results should include averages and variation, not only a single high-precision number. The same benchmark inputs and baseline algorithms should be used to avoid attributing differences to unrelated conditions.

Overall, the source supports a promising proof-of-concept idea: compress a text payload, embed it in a carrier image using LSB substitution and reverse the operations at the receiver. It does not yet demonstrate the broader claims of Unicode-wide effectiveness, improved security, robustness to image manipulation or general superiority over conventional LZW. The most valuable next step is not to add stronger claims, but to implement a transparent test suite that measures payload reduction, image fidelity, extraction accuracy, runtime and resistance to detection under controlled conditions.

6. Limitations and future research

Several limitations restrict the strength of the present findings. The source uses one short text example for its principal comparison, does not specify a representative corpus and does not fully define the modified codeword format. Image-size units are inconsistent between the table headings and the narrative. The combined text-plus-image table adds values that appear to use different units. The source does not report enough information to reproduce the image-compression runtime or to calculate the exact image fidelity of the same cover/stego pair. Nor does it provide a systematic steganalysis evaluation. These are reporting and validation gaps rather than proof that the proposed approach cannot work.

Future experiments should define a complete specification for both standard and modified LZW, including initial dictionaries, codeword widths, dictionary limits, reset behavior, byte order and end-of-stream markers. The evaluation corpus should contain messages of varied length and redundancy in UTF-8, including Bangla, English, Arabic, emoji and mixed-script samples. For each message, the original bytes, compressed bytes and recovered bytes should be stored and hashed. Multiple cover images with varied dimensions, textures and color distributions should be tested at several payload rates. The output format should preserve the modified pixel values and every file-size comparison should use the same unit and include metadata and framing overhead.

Image-quality analysis should report MSE, PSNR and payload in bits per pixel. A stronger security evaluation should compare the stego images against cover images using documented statistical or machine-learning steganalysis with training and test data

separated to avoid leakage. Robustness experiments should measure extraction success after lossless format conversion, resizing, cropping, filtering and lossy recompression, while recognizing that basic spatial LSB methods are expected to be fragile under many such operations. Runtime tests should be repeated under controlled hardware and software conditions. Finally, comparisons should include uncompressed LSB, standard LZW plus LSB and the proposed modified LZW plus LSB using the same payloads and carrier images.

7. Conclusion

This article reconstructed and critically analyzed a multilevel steganography workflow that combines LZW-style lossless compression with LSB image embedding. The documented process compresses a text payload, embeds its bits in an image, extracts the payload at the receiver and decompresses it to recover the original message. In the thesis's worked example, a 104-bit representation of "mynameismeena" is reported to become 99 bits with standard LZW and 84 bits with a modified representation. Recalculation gives reductions of approximately 4.8% and 19.2%, respectively. The source also reports an image-size change from 81 KB to 13 KB, but the table's unit labels and the combined-size comparison require caution. A reported 112-to-185 combined numerical ratio corresponds to about 60.5% remaining and a 39.5% reduction only if the quantities are comparable.

The results illustrate why compression before embedding can be useful, but they do not establish universal compression gains, security against steganalysis, robustness to image transformations or compatibility with all Unicode text. LSB substitution may preserve visual quality at modest payloads, yet it is vulnerable to statistical detection and operations that alter pixel values. Compression is not encryption and a secure deployment should protect the payload independently. Future work should define the modified algorithm precisely and evaluate multilingual byte streams, normalized units, exact recovery, image fidelity, capacity, runtime and steganalysis resistance under reproducible conditions. The most defensible conclusion is that the proposed combination is a plausible proof of concept whose broader performance and security claims require systematic experimental validation.

Declarations

Data availability

The numerical examples in this manuscript are drawn from the supplied thesis. No new experimental dataset was generated for this article.

Ethics statement

No human participants, personal data or animal subjects were involved in the document-based analysis.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations or those of the publisher, the editors and the reviewers.

References

[1] Frączek W, Mazurczyk W, Szczypiorski K. Multi-Level Steganography: Improving Hidden Communication in Networks. Journal of Universal Computer Science. 2012;18(14):1967–1986. https://doi.org/10.3217/jucs-018-14-1967.

[2] Kajal I, Rohil H, Kajal A. LZW based Image Steganography Using Kekre's Algorithm. International Journal of Computer Science and Information Technologies. 2014;5(2).

[3] Malik A, Sikka G, Varma HK. A High Capacity Text Steganography Scheme Based on LZW Compression and Color Coding. 2016.

[4] Smadi MA, Al-Haija QA. A Modified Lempel–Ziv–Welch Source Coding Algorithm for Efficient Data Compression. Journal of Theoretical and Applied Information Technology. 2014;61(1).

[5] Azad MK, Sharmeen R, Ahmad S, Kamruzzaman SM. An Efficient Technique for Text Compression. In: Proceedings of IMB2005. 2005. arXiv: 1009.4981.

[6] Ogundele TJ, Adetunmbi AO. Development of Multi-Level System of Steganography. Journal of Computer Science and Technology. 2013;13(1).

[7] Wei WY. An Introduction to Image Compression. Master's thesis, Graduate Institute of Communication Engineering, National Taiwan University, Taipei, Taiwan; 2008.

[8] Hamid N, Yahya A, Ahmad RB, Al-Qershi OM. Image Steganography Techniques: An Overview. International Journal of Computer Science and Security. 2012;6(3):168–187.

[9] Nagaria B, Parikh A, Mandliya S, Shrivastav N. Steganographic Approach for Data Hiding Using LSB Techniques. International Journal of Advanced Computer Research. 2012;2(4), Issue 6.

[10] Ahmed JM, Ali ZM. Information Hiding Using LSB Technique. International Journal of Computer Science and Network Security. 2011;11(4).

[11] Abu Taleb SA, Musafa HMJ, Khtoom AM, Gharaybih IK. Improving LZW Image Compression. European Journal of Scientific Research. 2010;44(3):502–509.

[12] Ibrahim W, Al-Mazaydeh A. Image Steganography Using LSB and LSB+Huffman Code. Journal of Computer Applications. 2014;99(5). https://doi.org/10.5120/17370-7896.

[13] Rezaei F, Ma T, Hempel M, Peng D, Sharif H. An anti-steganographic approach for removing secret information in digital audio data hidden by spread spectrum methods. In: Proceedings of the IEEE International Conference on Communications. 2013:2117–2122. https://doi.org/10.1109/ICC.2013.6654839.

[14] Modak P, Najmin S, Sharmin T, Badhon JA. Multilevel Steganography of Unicode Using Modified Lempel-Ziv-Welch Algorithm and Least Significant Bit Algorithm. Undergraduate thesis. Department of Computer Science and Engineering, Bangladesh Army University of Science and Technology; 2019.