Journal of International Commercial Law and Technology
Print ISSN: 1901-8401

Website: https://www.jiclt.com/

Article

StegoHash: A Dual-Layer Tamper Detection Model Using Steganography and SHA-256 Hashing

Article History:

Name of Author:
Dr. Ruchika Sharma[1], Ms. Aparna Raj Singh [2], Ms. Prerna Pathak[3] and Rakshit Sharma[4]

Affiliation:
[1]Assistant Professor, Jagan Institute of Management Studies, New Delhi, India.
[2]Assistant Professor, Jagan Institute of Management Studies, New Delhi, India.
[3]Student UG-IT, JaganNath Community College, New Delhi, India.
[4]Student UG-IT, JaganNath Community College, New Delhi, India.

Corresponding Author:
Dr. Ruchika Sharma

How to cite this article: Ruchika Sharma, *et, al.* StegoHash: A Dual-Layer Tamper Detection Model Using Steganography and SHA-256 Hashing. *J Int Commer Law Technol.* 2025;6(1):1183–1190.

Received: *13-10-2025*
Revised: *23-10-2025*
Accepted: *09-11-2025*
Published: *26-11-2025*

Abstract: In a time where digital communication rules, the demand for an unbreakable and authentic data exchange is greater than ever. Conventional security measures such as encryption create confidentiality in messages, but they fail to hide the fact that data is being transmitted. Steganography, however, effectively hides the fact that a message is being sent but fails to include integrity checking. This paper introduces StegoHash, a dual-layer security system that fuses SHA-256 cryptographic hashing and Least Significant Bit Steganography. Under this procedure, the message is hashed initially using SHA-256 , then both the original message and its corresponding digest are embedded inside a cover image using LSB. On delivery to the target device, the embedded message and its hash are extracted, followed by the calculation of a new hash for the message to compare with the extracted one. If a comparison between the two hashes indicates a match, the message is guaranteed to be authentic; otherwise, tampering has taken place. This model can effectively fuse cryptographic hash checking to ensure integrity with Steganography to ensure confidentiality, thereby offering a lightweight and effective dual-layer data security system.

Keywords: Steganography, Message Digest, SHA-256 Hash, Cryptography, Tamper Detection , Stego Integrity, StegoHash, LSB.

1. INTRODUCTION

The swift development of computer-mediated communication and cloud-based data exchange necessitate a maximum focus on preserving the confidentiality and integrity of exchanged data. Classical cryptographic methods encrypt data so that it becomes inaccessible to an unauthorized party; yet, they do not temper the possibility of detecting the transmission of sensitive information. Steganography, as opposed to that, conceals the fact of transmitting a message, and thus, it provides secret data transmission in a seemingly harmless carrier like an image or audio signal [1].

To address the dual issues of confidentiality and authenticity, the current study suggests StegoHash, a hybrid security system using SHA-256 hashing to generate a secure fingerprint, or message digest, of data [13]. It then embeds the original message and the associated hash within an image using Least Significant Bit Steganography. The proposed StegoHash model presents a dual-layered protection

mechanism—data concealment by Steganography and data integrity verification by hashing[14].

The rest of the introduction explores the basic building blocks and technologies comprising StegoHash, starting with cryptography and then Steganography, in the same order of implementation is proposed through this research.

1.1 Cryptography

Cryptography is a specialized discipline committed to the development of secure methods intended to protect information against unauthorized access, modification, or disclosure [14].

In the context of modern digital communications, it is crucial to provide confidentiality, integrity, and authenticity [7]. Cryptographic techniques employ mathematical algorithms to transform plaintext into ciphertext or generate unique signatures that verify the authenticity of information. While encryption and decryption are the most common cryptographic processes, the scope further extends to mechanisms like hashing, digital signatures, and key exchange systems. Full encryption is not used in the context of the current paper instead, a cryptographic hash function is used to generate a unique signature of the message, allowing it to be verified for authenticity. Such signatures are known as message digest [3].

Fig 1. Original Image before LSB embedding

Fig 2. Stego Image after LSB embedding

This figure no 1. compares the original image and the stego image after LSB embedding, demonstrating that no visible distortion occurs.

1.1.1 Message Digest

A message digest is a fixed-length, irreversible result of executing a cryptographic hash function on an input of variable size. It serves as a distinctive identifier of the original message and is utilized heavily for verifying data integrity. Small changes to the original input—i.e., changing one character—result in a very different digest. This property makes message digest useful for deployment in tamper detection, file authentication, and password checking. The main purpose of using a message digest in steganographic security systems is to provide a method for ascertaining whether a hidden message has been modified while in transit or when stored.

1.1.2 SHA-256 Hash Function

SHA-256 (Secure Hash Algorithm - 256-bit) is a cryptographic hash function that was created as a member of the SHA-2 family. It produces a 256-bit (64-character hexadecimal) hash from input data and is said to be very secure and collision-resistant. The algorithm is also deterministic because the same input always produces the same hash output. Of greater significance, it shows the avalanche effect, which states that even when a one-bit change is made in the input, the output is very dissimilar. SHA-256 has been widely employed in blockchain, digital signatures, certificate creation, and secure communication because it is highly resistant to cryptographic attacks [13]. In the StegoHash model presented, SHA-256 is employed to create a hash of the message prior to embedding it in the image. This allows the receiver to check if the message is tampered with, thus ensuring the integrity of the concealed data. [17]

1.1.3 Tamper Detection

Tamper detection is the method of detecting unauthorized modification of data during transmission or storage [5]. In StegoHash, this is done by comparing the hash computed through the use of the image with a newly computed hash of the concealed message [8]. On discovering both hash values are equal, it concludes that the message has not been tampered with and is hence genuine. A discrepancy in the hash values, however, suggests potential tampering. This detection method is a light-weight and efficient replacement for encryption-based authentication, particularly in applications where the importance of maintaining data integrity cannot be overemphasized. By incorporating tamper detection using SHA-256 hashing in steganographic embedding, the framework offers overall protection to concealed communications without sacrificing

efficiency [9][13][17] .

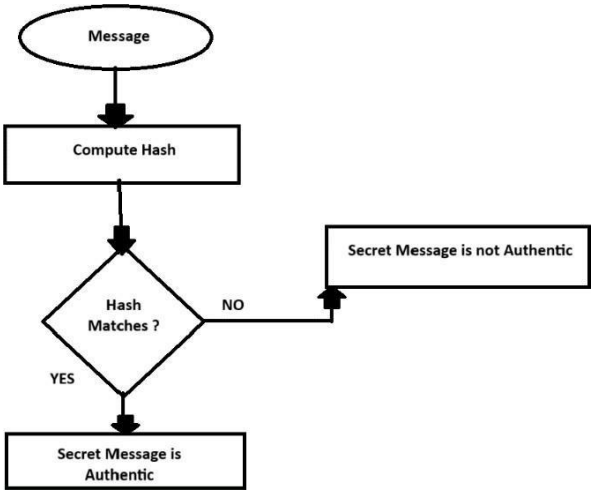

Fig 3. Tamper detection

The diagram shows the verification phase of StegoHash. After extracting the hidden message, a new hash is computed and compared with the embedded hash. If the hash matches, the message is confirmed to be authentic. If the hash does not match, it indicates that the message has been altered or tampered with.

1.2 Steganography

Steganography is the practice of hiding information in an apparently harmless carrier, thus making the fact that data is being transmitted imperceptible to unwanted viewers. Whereas cryptography is about the modification of content to preserve confidentiality; Steganography is about hiding the message. In digital technology, Steganography is used mainly in multimedia media such as images, audio, video, and text. Among all the media, digital images are the most widely used, because they have large file size, redundancy in pixels by nature, and low visual effect when the changes are minimal [18].

Digital images consist of discrete pixels that represent color information in terms of binary data [4]. In a common 24-bit image, each pixel contains three 8-bit components representing the Red, Green, and Blue (RGB) color channels [1]. Least Significant Bit technique is known as one of the simplest but most efficient methods for embedding data in digital images. This technique includes modifying the least significant bits of pixel color values to embed binary data. Since modifications to the LSB do not noticeably impact the overall pixel value, these modifications are imperceptible to the human eye [4]. For example, a pixel value of 11001000, when modified to 11001001, has a difference of just one unit and cannot be noticed by the eye in the output image. This can be achieved through the application of this method, whereby an encrypted message—transliterated to binary form—is concealed within the LSB of the image pixels. Three bits of data can typically be conveyed per pixel (one per color component), and in a typical-sized image, this translates to a substantial payload size. Secondly, since the changes are dispersed across a large collection of pixels, visual quality of the image is preserved. That is, the cover image, now transformed into a stego image, will appear natural and unaltered to the human eye, as well as to standard digital inspection.

Yet, it should be noted that the LSB method works best with lossless or uncompressed image formats like BMP or PNG [12]. Lossy formats like JPEG, in contrast, employ compression methods that can change pixel values and thus obliterate the steganographic data [4]. For optimal message extraction and acceptable image quality, file types and pixel selection are absolutely crucial. In spite of its simplicity, the LSB method offers a simple yet effective solution to steganographic data transmission, particularly when used in concert with other mechanisms like cryptographic hash verification or such as our StegoHash model.

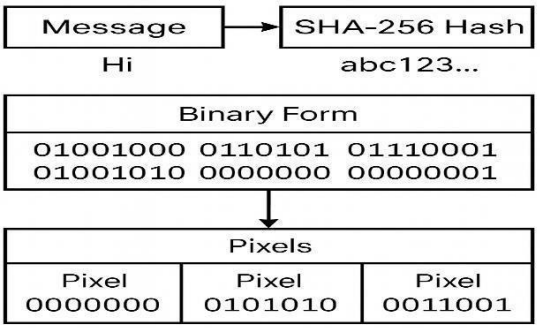

Fig 4. Binary Stream Structure

This diagram shows how StegoHash embeds data. The message is first hashed with SHA-256, and the message plus hash are converted into binary. These binary bits are then inserted into the least significant bits of image pixels. Each pixel stores a few bits, producing a stego-image that looks unchanged but secretly contains both the message and its integrity-checking hash.

1.2.1 Image Security

Image security in Steganography means that the stego image can retain its original visual appearance and structural integrity even when hiding the secret data. This is to ensure that the image will not arouse suspicion about the existence of concealed information inside the image. The main aim is not to create perceptual and statistical alterations that may reflect tampering or concealed communication.

Image security is realized in LSB based Steganography by altering only the least significant bits of pixel values, which make very minimal contributions to the general visual appearance of the image.[9][1]

To quantify image security, a number of measures are used, most prominently Peak Signal-to-Noise Ratio (PSNR) and the Structural Similarity Index Measure (SSIM) [9]. PSNR measures similarity between the cover image and stego image in terms of pixel-level alteration, with higher PSNR relating to higher quality [12]. Conversely, SSIM measures perceived visual difference by analyzing brightness, contrast, and structural detail change. In the correct methodology, Least Significant Bit Steganography achieves a PSNR value greater than 50 dB and an SSIM value of almost 1.0, both representing high image fidelity with minimal distortion. The image security component is thus of paramount significance to the success of any steganographic model, such as the StegoHash model proposed here, where message and hash concealment both must be visually imperceptible.[6][4][8]

1.3 Steganography and stegohash integrity

In steganographic systems, stego integrity refers to the assurance that the embedded message incorporated into the cover medium is not altered, remains whole, and can be authenticated upon reception [9]. While traditional systems of Steganography seek to hide data existence, they do not contain a straightforward process for determining whether the embedded message has been corrupted, either inadvertently or through malicious attack. To address this shortcoming, modern systems incorporate cryptographic hash functions and data embedding algorithms. The incorporation of both allows the receiver to confirm that the extracted message is the same as that which the sender originally embedded [3][17].

The StegoHash model is a new dual-layer security model specifically designed to provide data confidentiality as well as integrity. This model utilizes the best of LSB Steganography and SHA-256 cryptographic hashing to provide these goals. In the model, the original message is initially passed to the SHA-256 algorithm to create a fixed-length hash value, an electronic fingerprint of the message[21]. The message and hash are concatenated and translated into a binary stream. This combined stream is then embedded into the least significant bits of an electronic image, resulting in a stego image that visually cannot be differentiated from the original. When the stego image is received, the embedded information is extracted and then gets separated into message sndd its hash value. The retrieved message is then subjected to the SHA-256 algorithm to create a new hash value. The new hash created is then compared to the hash value retrieved from the stego image. A similarity in the two hash values serves as confirmation that the message is original and not altered in its transmission. A difference in them, however, confirms that the message has been altered. Such a feature of detecting tampering makes StegoHash, a very efficient system for secure communication since not only is the information concealed but its integrity is also verified without the resort to conventional encryption systems.

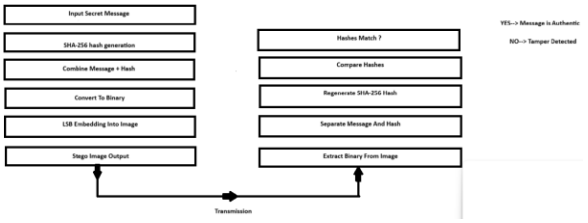

Fig 5. StegoHash Flow Diagram

This figure represents the complete workflow of the StegoHash model, showing how the message and its hash are processed, embedded, and verified from sender to receiver.

2. LITERATURE REVIEW

Over the past few years, many researchers have worked on making communication more secure by using either cryptography, steganography, or both the methods. For example, Sharma and Kumar [14] discussed how both techniques can be combined, while they previously focused on data hiding through recursive equations [1]. Even though these studies improved confidentiality and integrity, researchers like Chroni et al. [5] pointed out that problems still exist, especially when it comes to detecting tampering and making data embedding more reliable.

A lot of research has been done using cryptographic hash functions along with image-based steganography such as the one discussed by Nazir and the techniques reviewed by Provos and Honeyman [1], showed how MD5 hashes were added into images to check if the data gets changed. This approach got success at that time, but MD5 is no longer trusted because attackers can create collisions, which makes the hash unreliable.

Later on, Sharma and Kumar [16] introduced a method that uses recursive equations to demonstrate where the data should be hidden inside an image. This technique improved randomness in pixel selection and made it harder for attacker to predict where the data is hidden, but the model still relied on traditional hashing methods like MD5 which worked mainly on large uncompressed formats like BMP,

which limits its practical use.

Some researchers, such as Haghighi et al. and Zain & Clarke, suggested encrypting the message first (using AES, RSA, etc.) and then hiding the encrypted data using techniques like LSB [9], DCT, or DWT. These approaches improve confidentiality, but still lack in heavy process and make it slow. And due to this reason,they are not ideal for lightweight systems or the situations where fast processing is mandate.

Forgác et al. also introduced machine-learning based approaches for stego-image detection and for improving embedding strategies. Their study was more advanced compared to traditional techniques and the methods are complex too which becomes difficult to implement for a simple or student-level applications.

After going through all this literature, one clear limitation keeps appearing that there is no simple, lightweight method that can hide data and also confirm whether the message was changed, without using full encryption every time.

To overcome this, our proposed StegoHash model tries to provide a balanced solution. Instead of using outdated hashing methods, StegoHash uses the modern SHA-256 function to create a reliable message digest [13]. It then hides both the message and its hash using a basic but effective LSB steganography method [9]. This way, any kind of tampering [can be spotted easily, and the image still looks normal to the human eye. The whole process also stays lightweight and does not require heavy encryption or excessive computing power.

3. METHODOLOGY

The suggested StegoHash model is a dual-layer protection system intended to ensure confidentiality and integrity of an embedded message. The procedure entails the embedding of an additional cryptographic hash along with the cover message into a digital image using the Least Significant Bit Steganography method. The combined methodology enables safe transmission of embedded messages while enabling detection of any unauthorized alteration. The procedure is split into two: the sender side, where the message is encoded and embedded into an image, and the recipient side, where the message is extracted and verified.

First, the sender enters the plain text message to be embedded. A SHA-256 hash of the message is calculated before embedding. The hash is used as an electronic fingerprint at the time of receipt for testing originality of the message. The original message and its hash are concatenated to create one string. The string is then translated into a binary bitstream and embedded into the LSB of the pixels of a cover image in an ordered fashion [9]. The LSB method is used because it is easy, provides high capacity, and can preserve visual similarity between original and stego images. Unlike previous steganographic methods that either hide plain messages or employ antiquated hash functions such as MD5, StegoHash employs a more secure, collision-resistant SHA-256 hash [3][17]. Additionally, by hiding a message together with its hash, the model also benefits from the aspect of tamper detection ,which many simple LSB methods lack [5]. The new image, now referred to as the stego image, not only contains the confidential material but also the capability to authenticate its origin, all without revealing anything about tampering to third parties.

Once the receiver receives the stego image, it extracts the embedded bitstream by scanning the LSBs of the pixels in the same order used for embedding. The bitstream is split into two: the embedded SHA-256 hash and the original message. The retrieved message is hashed by passing the SHA-256 function to it. If the hash received matches the retrieved hash, the receiver confirms that the message is original and unchanged. If the hashes do not match, it indicates that the message has been altered or corrupted during transmission, thus making its authenticity invalid.

This technique does away with the necessity for sophisticated encryption techniques or extensive key management systems, but still offers a high degree of security. It is characterized by its lightness, efficiency, and effectiveness and is appropriate for applications that require covert transmission and integrity verification, including secure messaging, confidential reporting, and academic certification data transmission, among others. With the integration of cryptographic hashing and steganographic embedding, StegoHash does away with significant limitations in earlier models and offers an effective, secure solution for tamper-resistant data hiding [3][17].

4. IMPLEMENTATION

We are implementing our approach using Python 3 due to its simplicity, versatility, and the availability of powerful libraries for image processing and cryptography. The model is divided into two main operational modules: the embedding module for the sender and the extraction and verification module for the receiver. Both modules perform different operations to hide data, verify integrity, and detect tampering.

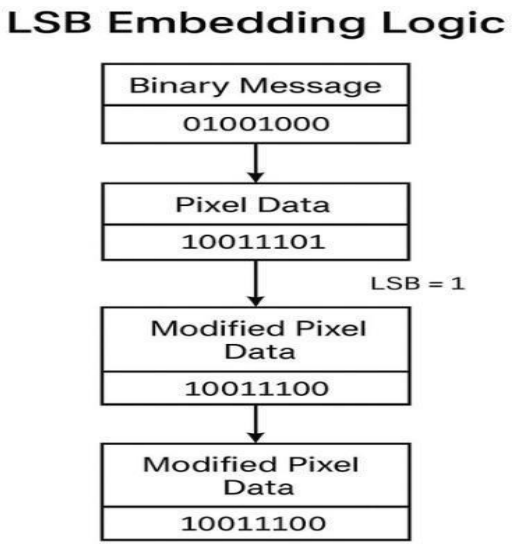

Fig 6. LSB Embedding Logic Image

The diagram shows how a bit from the binary message is embedded into an image pixel using Least Significant Bit substitution. The original pixel value is represented in binary, and its last bit is replaced with the next bit from the secret message. This produces a modified pixel whose visual appearance remains unchanged, but carries one hidden bit of data. Repeating this process across pixels embeds the entire message inside the image.

4.1 Sender Side – Process of Integration

The process begins with a user-input text message. The message is hashed using the hashlib.sha256() function to generate a 256-bit SHA-256 hash in hexadecimal format. This hash is a unique identifier of the message. The message and hash are combined into a single string and converted into a binary stream. Every character of the message-hash string is converted into its equivalent 8-bit ASCII representation, and this gives us the complete binary stream.

The binary stream is then embedded in a cover image through the use of the Least Significant Bit method. The method utilizes the Python Imaging Library (PIL) to read and manipulate the image. The cover image is first accessed, enabling sequential access to the Red, Green, and Blue values of each pixel. In each color channel, the least significant bit of the byte is replaced with a corresponding bit of the binary stream. This is done until the binary stream has been fully embedded. The modified pixels are then used to create a new image — commonly known as the stego image — that is perceptually indistinguishable from the original but contains hidden data.

4.2 Recipient Side – Extraction and Verification

On the receiver side, the stego image is read through the same PIL module. The LSBs of the image pixels are read sequentially to reconstruct the concealed binary stream. The binary data are read in 8-bit blocks and demodulated into ASCII characters, thereby reconstructing the original message+hash string. The last 64 characters of this string are the SHA-256 hash, and the remaining part is the original message.

To verify the integrity of the message, the recipient uses Python's hashlib.sha256() to create a new hash of the message retrieved. The new hash is then compared to the hash derived from the image. When both hashes are equal, it is used to verify that the message has not been altered. When the hashes are not equal, the system sends a Tampering alert, which indicates that the data was altered in transit or storage.

4.3 Modules and Tools Used in Python

- PIL.Image: Used for pixel-level image data processing and loading
- hashlib: To create SHA-256 hash of the message
- os, math, and bitstring (optional): They are used for file sizes, conversions, and stream formatting.

```
# Generate SHA-256 [13] hash
import hashlib
from PIL import Imagei
message = "Hi, my name is Tina. This is a secret message."
hash_val = hashlib.sha256(message.encode()).hexdigest()
combined = message + hash_val
binary_data = ''.join(format(ord(char), '08b') for char in combined)

img = Image.open('cover.png')
pixels = list(img.getdata())
new_pixels = []
idx = 0
for pixel in pixels:
  r, g, b = pixel
  if idx < len(binary_data):
    r = (r & ~1) | int(binary_data[idx])
    idx += 1
  if idx < len(binary_data):
    g = (g & ~1) | int(binary_data[idx])
    idx += 1
  if idx < len(binary_data):
    b = (b & ~1) | int(binary_data[idx])
    idx += 1
  new_pixels.append((r, g, b))
img.putdata(new_pixels)
img.save('stego_image.png')
```

The code performs the embedding stage of the StegoHash method. It first generates a SHA-256 hash

of the secret message and appends it to the original text. This combined data is converted into binary and embedded into the least significant bits of the image's RGB pixels. Each pixel's LSBs are replaced sequentially with bits of the binary stream, producing a stego-image that visually appears unchanged. By embedding both the message and its hash, the method ensures hidden communication with built-in integrity verification, allowing tamper detection during extraction.

4.4 Advantages

- The StegoHash model has a number of advantages over traditional steganographic and cryptographic models, particularly where confidentiality as well as integrity are both necessary without leading to strict computational requirements.
- First, the system combines two robust mechanisms — SHA-256 hashing and LSB Steganography; to provide a dual-layer security. This not only hides the message but also makes it verifiable. On any attempt to alter any part of the hidden message, the integrity check will fail, thus making it a reliable system for tampering detection.
- Second, employing SHA-256, a cryptographically secure, collision-resistant hash algorithm, guarantees that even subtle variations in the message will result in a completely different hash, thus identifying unauthorized modifications. StegoHash is thus more secure than previous versions that employed weak hashing algorithms like MD5 or SHA-1.
- Thirdly, the Least Significant Bit technique allows for high data embedding capacity with zero or no perceptible distortion. The image changes are not discernible, thus preserving the image quality integrity while allowing safe transmission of data.
- Fourth, the design is lean and efficient, requiring neither advanced encryption and decryption algorithms nor large-scale key management infrastructures. It can be implemented with basic Python libraries and can be used in resource-constrained environments, like educational laboratories or small secure communication environments.
- Finally, StegoHash's modularity makes it very easy to extend. It can be applied to other file formats, can be enhanced with new, more secure hash functions as they become available, or can even be integrated into bigger secure communication systems.

4.5 Limitations

While StegoHash is a good and realistic model of secure message hiding with integrity checking, it is not flawless.

The model that is used currently uses LSB embedding, which is susceptible to aggressive image processing such as compression, resizing, or format conversion. If a stego image is saved in a lossy format such as JPEG, the embedded message may be lost or severely corrupted.

Additionally, StegoHash is dependent on the recipient being aware of the correct extraction order and image type. Any modification in the order or pixel alteration can lead to incorrect extraction of the message or its hash.

Lastly, while SHA-256 provides adequate protection for integrity, it does not encrypt the message. Thus, the system provides verification but not confidentiality from a cryptographic point of view. In highly sensitive applications, this model is utilized alongside encryption techniques to provide full-spectrum security.

5. CONCLUSION

StegoHash model offers a highly efficient and simple-to-implement solution for protecting data hiding and tamper detection in electronic communication [16][5]. Utilizing the cryptographic security of SHA-256 and the imperceptibility of Least Significant Bit Steganography ,the model offers a dual-layer protection scheme for guaranteeing confidentiality and integrity of the transmitting message [13][9][1]. The research process utilized in this paper demonstrates that data can be concealed in an image in a way that the eye is unable to perceive, and authenticity verification can be done through cryptographic hash comparison.

Compared to typical steganographic methods that are interested only in concealment, StegoHash has a robust integrity test, and hence the recipient is capable of verifying if the received message has been tampered with or not. The implemented design, which is based on Python, proved to be efficient, lightweight, and highly adjustable in real-world scenarios such as secure internal communication, confidential document transfer, and academic credential protection.

With its effectiveness, minimal computational load, and good verification quality, StegoHash presents an attractive solution to modern digital security issues. Prospects for further development include extending the model to other types of documents, encrypted content Steganography, or employing adaptive embedding techniques to enhance steganalysis and image forgery protection further.

REFERENCES

1. R. Sharma, V. Kumar, "Implementation of Steganography Using Recursive Equation Approach", International Journal of Computer & Mathematical Sciences, vol. 4, pp. 15–19, 2015.
2. R. Sharma, V. Kumar, "Recursive Equation Approach of Information Hiding for Authentication of Digital Data", Journal of Algebraic Statistics, vol. 13, no. 2, pp. 813–819, 2022.
3. Nazir, S , Kaleem, M, "Embedded Hash Codes for Image Similarity Detection and Tamper Proofing", ETECTE, 2023.
4. R. Forgác, M. Očkay, "Steganography Approach to Image Authentication Using PCNN", Computing and Informatics, vol. 42, pp. 591-614, 2023.
5. M. Chroni, "Tamper Detection and Localization in Forensic Images", ACM Digital Library, 2023.
6. Y. Lin, C.C.Chang, "Tampering Detection in Absolute Moment Block Truncation Coding (AMBTC) Compressed Images", Electronics (MDPI), 2025.
7. J. M. Zain, M. Clarke, "Reversible Region of Non-Interest (RONI) Watermarking for Authentication of DICOM Images", IJCSNS, vol. 7, no. 9, 2007.
8. B. Bolourian Haghighi, A. H. Taherinia, "TRLF: An Effective Semi-fragile Watermarking Method for Tamper Detection and Recovery based on LWT and FNN", arXiv, 2018.
9. S. Chakraborty, A. S. Jalal, "LSB Based Non Blind Predictive Edge Adaptive Image Steganography", arXiv, 2022.
10. A. Khaldi, "Diffie-Hellman Key Exchange through Steganographied Images", Law State and Telecommunications Review, vol. 10, no. 1, pp. 147–160, 2018.
11. J. Tian, "High capacity reversible data embedding and content authentication", ICASSP'03, vol. 3, pp. III-517, 2003 .
12. H. Kaur, "A Survey on Different Techniques of Steganography", MATEC Web of Conferences, 2016.
13. NIST, "FIPS PUB 180-4 Secure Hash Standard (SHS)", 2015.
14. R. Sharma, V. Kumar, "Hybrid Content Security Model Using Steganography and Cryptography in Cloud Storage", European Economic Letters, vol. 15, issue 4, 2025.
15. T. S. Gouda, M. Ravishankar, "Design and Development of Image Forensic Techniques for Forgeries, Watermarking, Cryptographic Integrity Verification and Blockchain", Journal of Neonatal Surgery, vol. 14, issue 11S, 2025.
16. R. Sharma, V. Kumar, "Information Hiding Using Linear Recursion", International Journal of Scientific & Engineering Research, vol. 11, issue 6, pp. 314–317, 2020.
17. E. P. Michel, "A Mixed Steganographic and Fragile Watermarking Approach for Tamper Detection Using SHA-256", Lecturas Matematicas, 2020.
18. P. C. Mandal, I. Mukherjee, "Digital Image Steganography: A Literature Survey", Elsevier Journal, 2022.
19. S. Nazir, "Embedded Hash Codes for Image Similarity Detection and Tamper-Proofing", ETECTE, 2022.
20. Kusum Lata, S. Rathee, "A Survey of Chaos-based Cryptography and Steganography Techniques", Journal of Propulsion Technology, vol. 45, no. 3, 2024.
21. R. Sharma, "Hybrid Content Security Model Using Steganography and Cryptography in Cloud Storage", European Economic Letters, vol. 15, no. 4, pp. 554–559, 2025.