Journal of Innovative Image Processing (ISSN: 2582-4252)
www.irojournals.com/iroiip/

BlockImage: A Secure Framework for Image Authentication and Provenance using AI and Blockchain

Sathyabama A R.[1], Jeevaa Katiravan[2]

Information Technology, Velammal Engineering College, Chennai, India

E-mail: [1]sathya.it1@gmail.com, [2]jeevaakatir@gmail.com

Abstract

In contemporary applications, especially digital forensics, intellectual property protection, and secure image sharing, it is essential to guarantee the security, integrity, and authenticity of digital images. To improve image authentication, this research presents BlockImage, a sophisticated architecture that combines blockchain storage, cryptographic hashing, AI-driven information extraction, and decentralized image retrieval through IPFS. After being refined on the modified CASIA Tampered Image Dataset, a ResNet-50 model outperformed traditional techniques with a tamper detection accuracy of 94.7%. The solution maintains immutable provenance tracking using the Hyperledger Fabric blockchain and effectively identifies modifications using SHA-256 cryptographic hashing. Furthermore, tamper-proof access to images is made possible through decentralised storage through IPFS, guaranteeing an average retrieval time of about 200 ms per image. Comparing experimental assessments to current methods reveals improved security, storage efficiency, and verification capabilities. The BlockImage framework offers a high-performance, scalable way to safeguard digital images from unwanted changes, guaranteeing their reliability and accessibility over time.

Keywords: Image Security, Blockchain, Cryptographic Hashing, Image Authentication, Digital Forensics, Decentralized Storage, AI-Based Image Analysis, BlockImage.

DOI: https://doi.org/10.36548/jiip.2025.1.002
Received:29.01.2025, received in revised form: 22.02.2025, accepted: 10.03.2025, published: 20.03.2025

1. Introduction

1.1 The Growing Need for Secure Image Management and the Advantages of BlockImage

The rapid dissemination of digital images across diverse fields such as forensic science, medical imaging, and intellectual property protection has engendered increasing concerns surrounding the security, integrity, and authenticity of these images. Because they rely on centralized storage systems, current authentication techniques like metadata-based verification and static watermarking are extremely vulnerable to tampering, illegal changes, and data loss. These weaknesses show how urgently a safe and dependable image management system is needed. By combining dynamic watermarking, cryptographic hashing, decentralised storage, and AI-driven metadata extraction to improve image authentication and security. By combining dynamic watermarking, cryptographic hashing, decentralised storage, and AI-driven metadata extraction to improve image authentication and security, BlockImage is a comprehensive system that tackles these issues. It creates distinct digital fingerprints for images using SHA-256 cryptographic hashing, guaranteeing data immutability and enabling reliable tamper detection. To prevent unwanted changes and preserve transparency and verifiability, authentication records are safely stored on the Hyperledger Fabric blockchain. The Interplanetary File System (IPFS) serves as a decentralized storage solution by distributing data across a network. This approach reduces the vulnerabilities of centralized cloud storage while ensuring long-term accessibility and resilience against system failures. BlockImage offers a scalable, safe, and future-proof solution for digital image management by combining these cutting-edge technologies, greatly outperforming traditional security measures.

1.2 Major Contributions

A number of significant developments in digital image security are presented in this research. The first significant contribution is the application of AI-driven metadata extraction, which uses a refined ResNet-50 model to accurately categorise and analyse. This lowers the possibility of human input errors by guaranteeing that metadata is automatically extracted and validated. The context-aware verification is improved by AI-based metadata analysis, which raises overall security and provenance monitoring.

The combination of blockchain verification and SHA-256 cryptographic hashing results in improved tamper detection, which is another significant addition. The BlockImage framework is very useful for digital forensics and intellectual property protection. By ensuring

that even the smallest changes in image data can be recognised and reported, this technique improves security and dependability. The combination of decentralised and immutable storage represents a third major innovation. BlockImage uses Hyperledger Fabric blockchain technology to produce an unchangeable and transparent record of authentication information. By removing the dangers of centralised storage, this blockchain-based method guarantees that provenance data is always unchangeable and verifiable.

With an average retrieval time of about 200 ms per image, BlockImage also improves image retrieval efficiency using IPFS-based decentralised storage. By distributing data among numerous nodes, IPFS ensures safe, effective, and continuous access to verified, in contrast to traditional cloud-based systems that are vulnerable to single points of failure The research follows by discussing the security and scalability issues with traditional image authentication methods. Blockchain, artificial intelligence, and cryptographic hashing are all combined in BlockImage to create a highly scalable and impenetrable security framework that can manage massive datasets. This makes it perfect for uses like copyright protection, medical imaging security, and digital forensics. BlockImage provides a thorough and forward-thinking method of digital image authentication by integrating these developments, guaranteeing integrity, security, and long-term accessibility.

2. Related Work

Making sure that multimedia material is secure, legitimate, and intact is of the utmost importance in today's fast-paced digital world. In an effort to overcome these obstacles, researchers have investigated a number of methods, such as watermarking, perceptual hashing, and blockchain integration.

2.1 Watermarking Techniques for Security and Authentication

Digital material may be protected through the use of watermarking, which involves encoding invisible information that can be recovered for verification purposes. To combat counterfeiting in packaging applications, Nguyen et al. [11] presented a method to protect printed matrix barcodes, like QR codes, by including watermarks. To further improve security by using quantum computing concepts, Singh et al. [3] presented the QMCS-IAS approach, which integrates compressed sensing with quantum processes for authentication. After watermark extraction, the original image may be completely recovered thanks to a reversible

watermarking approach using difference expansion [1]. This ensures both security and quality preservation. To improve the safety of electronic health records, author Sharma presented a medical image watermarking method with several potential uses. Patients' right to privacy and the reliability of medical diagnoses depend on their approach to protecting the authenticity and privacy of medical images [16]. Another contribution in this field about medical image watermarking method specifically designed for safe e-healthcare applications, emphasising image quality preservation while guaranteeing security [1]. The method examines the problem of obtaining encrypted voice material without decryption directly. They provide a system that may produce distinct identities for encrypted audio segments by combining perceptual hashing with short-term cross-correlation coefficients. Encrypted voice data may be efficiently matched and retrieved using this method, which guarantees both security and accessibility. Despite subjecting the voice signals to a number of processes designed to preserve their content, the authors show that their system still manages to obtain excellent recall and accuracy rates. Enhancing both efficiency and security, their solution eliminates the necessity for downloading and decoding voice data during the retrieval process.

2.2 Perceptual Hashing for Content Identification and Authentication

To enable identification even after alterations, perceptual hashing creates a hash value using the perceptual properties of the information. A technique that successfully identifies and recovers images despite numerous modifications was presented by Du, et al. [3] as they investigated perceptual hashing for image authentication and retrieval. To facilitate effective identification and authentication in the audio domain, Báez-Suárez et al. [9] presented SAMAF, a sequence-to-sequence autoencoder model for audio fingerprinting. This model generates compact and resilient representations of audio signals. To authenticate and identify information, Khelifi and Bouridane presented a perceptual video hashing approach. This method guarantees correct identification regardless of alterations, such as format changes or compression.

2.3 Advancements in Image Encryption and Privacy Preservation

It is essential to prioritize the improvement of image security in multimedia applications. For a more robust encryption procedure that safeguards images from illegal access, Li et al [7]. suggested a method that merges the Aboodh transformation with S-Box fusion. In order to preserve sensitive visual information while keeping quality, Soualmi et al.

[6] presented an intelligent method for colour image privacy preservation that uses sophisticated algorithms. Perceptual hashing methods for authentication and retrieval are investigated in this research. By compressing an image's visual information, perceptual hashing makes it possible to identify images even after they've been through a number of changes.

To provide trustworthy image authentication and effective retrieval in massive databases, the article lays forth ways to improve the discrimination and resilience of perceptual hashes [11]. The difficulty of decrypting encrypted speech without decrypting it is the subject of this research. To create distinct identities for encrypted audio segments, the authors suggest a technique that integrates perceptual hashing with short-term cross-correlation. This method allows for the rapid recovery and matching of encrypted voice data while keeping it secure and accessible. The algorithm maintains high rates of accuracy and recall even after subjecting speech signals to multiple operations that preserve their content.[12]. The authors provide a new method of watermarking printed matrix barcodes (QR codes) to prevent package counterfeiting. It is difficult for counterfeiters to reproduce the method's security layer, which is created by inserting a certain random micro-texture inside the barcode. When counterfeiting occurs, it degrades the embedded micro-texture, which changes its statistical characteristics. Additionally, the research introduces statistical detectors that differentiate between real and fake printed barcodes using hypothesis testing. The efficiency of this anti-counterfeiting solution has been confirmed by experimental findings.[13].

2.4 Blockchain Integration for Secure Digital Evidence Preservation

Integrating blockchain technology into security frameworks has demonstrated encouraging results, especially when it comes to safeguarding digital evidence. The immutability and verifiability of digital evidence are vital in legal and forensic settings, and Kumar and Sharma created a safe approach for preserving it that is optimized for IoT environments utilizing blockchain [5]. In their comprehensive study, Ahmad et al. highlighted several methods and their efficacy in enhancing real-time image security in IoT settings. [12][17].

2.5 Comprehensive Reviews and Surveys

To gauge where security measures in multimedia apps are right now, many in-depth evaluations have been undertaken. In thorough research that helps in understanding and resisting image manipulation, Zhang and Zhang et al [1] investigated the taxonomy, strategies,

and tools linked to digital forgery. In-depth analysis of the several watermarking methods used for identification and identity protection. The researchers go over several watermarking techniques, their uses, and the difficulties that come with them. The cumulative effect of this research highlights the continuous work and progress toward the goal of multimedia content security in all its forms and across all platforms. To meet the ever-changing demands of digital security and authentication, it is crucial to create more effective watermarking methods, and perceptual hashing algorithms, and to include new technology like as blockchain.

3. Proposed System

Digital image safety, authenticity, and long-term accessibility are guaranteed by BlockImage's multi-layered security architecture. Dynamic watermarking, AI-driven metadata extraction, cryptographic hashing, blockchain-based provenance tracking, and decentralised storage through IPFS are its five main constituents.

3.1 End-to-End Workflow

The entire workflow of the BlockImage system (Figure 1) follows a sequence of steps to ensure that each image is processed, analyzed, and securely stored the unified flow is represented in the Figure.2. The steps in the process are as follows:

1. Image Acquisition: The image is first acquired and processed through dynamic watermarking.

2. Metadata Extraction: Al algorithms analyze the image to extract relevant metadata

3. Hash Generation: A cryptographic hash is generated for the image to ensure its integrity.

4. Storage on IPFS: The image is stored securely on IPFS, providing decentralized storage.

5. Blockchain Registration: The image's hash, metadata, and IPFS location are recorded in the blockchain for immutable and verifiable storage.

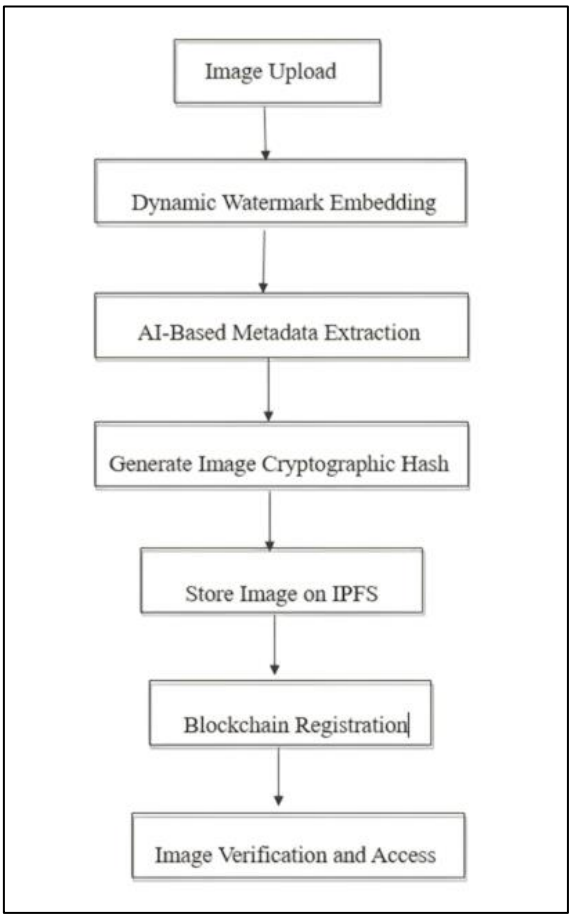


Figure 1. BlockImage Architecture

3.2 Algorithm Unified Flow

1. **Input:** Original Image I
2. **Process Image:** $I' = \text{EmbedWatermark}(I, W)$
3. **Extract Metadata:** $M = \text{ExtractMetadata}(I')$
4. **Generate Cryptographic Hash:** $H = \text{SHA256}(I')$
5. **Store on IPFS:** $H_{\text{IPFS}} = \text{IPFSUpload}(I')$
6. **Store on Blockchain:** $T_{\text{txn}} = \text{StoreOnBlockchain}(H, M, H_{\text{IPFS}})$
7. **Verification:** For a verification image I_{verify}, recompute the hash H':

$$H' = \text{SHA256}(I_{\text{verify}})$$

Then check for verification:

$$V = \text{VerifyOnBlockchain}(H', T_{\text{txn}})$$

8. **Output:** Verification Status V, Blockchain Transaction Hash T_{txn}, and IPFS Hash H_{IPFS}

Figure 2. Unified Flow

3.3 Dataset Preprocessing and Labelling

A well-structured dataset was used for testing in order to assess the suggested BlockImage framework's robustness and performance. The CASIA Tampered Image Dataset is being modified with real-time images and taken for further processing. In the preprocessing pipeline, the original images were downsized to 256 × 256 pixels so that all the images in the collection are consistent. The grayscale image simplifies the data while maintaining structural details by reducing the colour channels from three (RGB) to one (grayscale), but maintaining the same 256 × 256 dimensions. A 5×5 Gaussian kernel is used for the 256 × 256 Denoised (Gaussian Blur) image to smooth it out and lower the noise. The Edge Detection (Canny) output, which uses gradient intensity variations to identify important edges, stays at 256 × 256. By redistributing pixel intensities, Histogram Equalization also known as 256 × 256 improves contrast and highlights hidden details. The recovered gradient-based features which are essential for object detection and forensic analysis are visualized in the HOG Feature Extraction image, which is still 256 × 256. Every stage improves the image representation, increasing its suitability for feature-based classification or tamper detection in the CASIA dataset. Figure 3 represents the processing and the labelling of the dataset used.

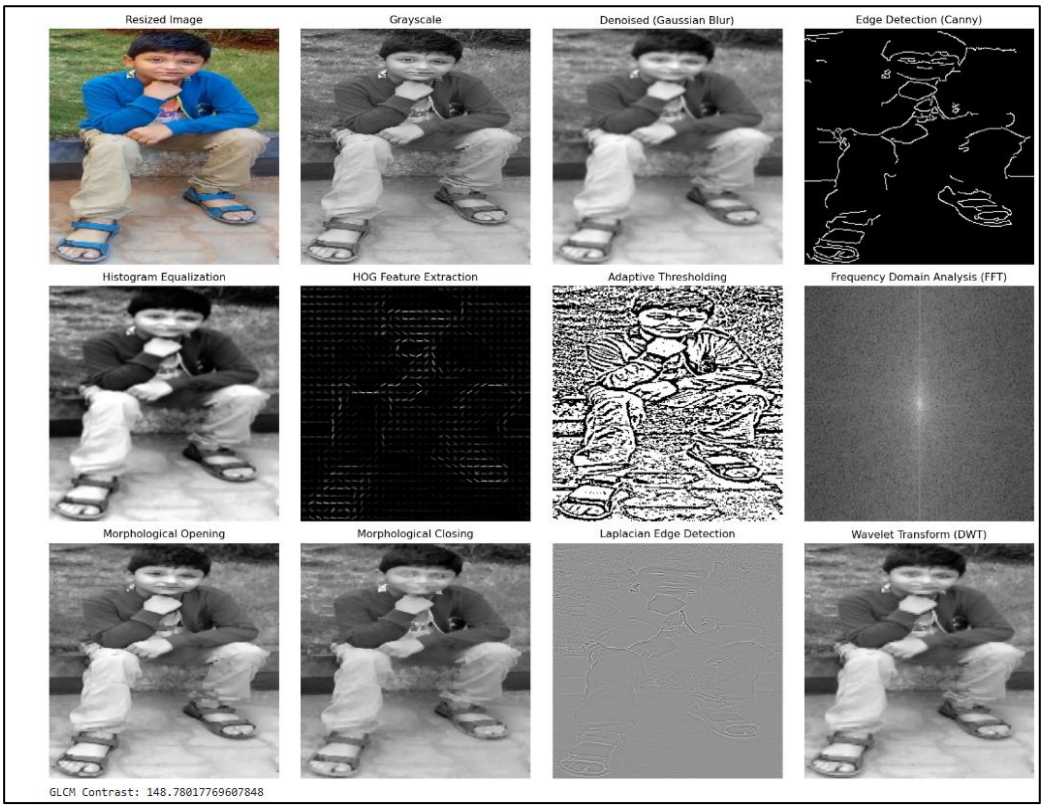


Figure 3. Preprocessing and Labelling

3.4 Dynamic Watermarking for Image Authentication

BlockImage employs dynamic watermarking (Figure 4) as the first layer of security to protect image integrity. This method embeds context-sensitive or time-specific watermarks that can adapt to image modifications, unlike traditional static watermarks, which are prone to removal through compression, cropping, or filtering. The system dynamically generates a watermark that encodes essential metadata such as the timestamp, owner information, and hash value of the image. When an image is altered, the watermark undergoes corresponding changes, making tampering detection more effective. The watermark embedding process involves Discrete Wavelet Transform (DWT) and Singular Value Decomposition (SVD) to ensure robustness against attacks. Adaptive thresholding techniques are used to ensure that the watermark remains imperceptible while maintaining high resilience against distortions. During verification, the watermark is extracted and compared against its original state, confirming image authenticity. To verify the legitimacy of the public image the watermark is taken off and compared to its initial condition.

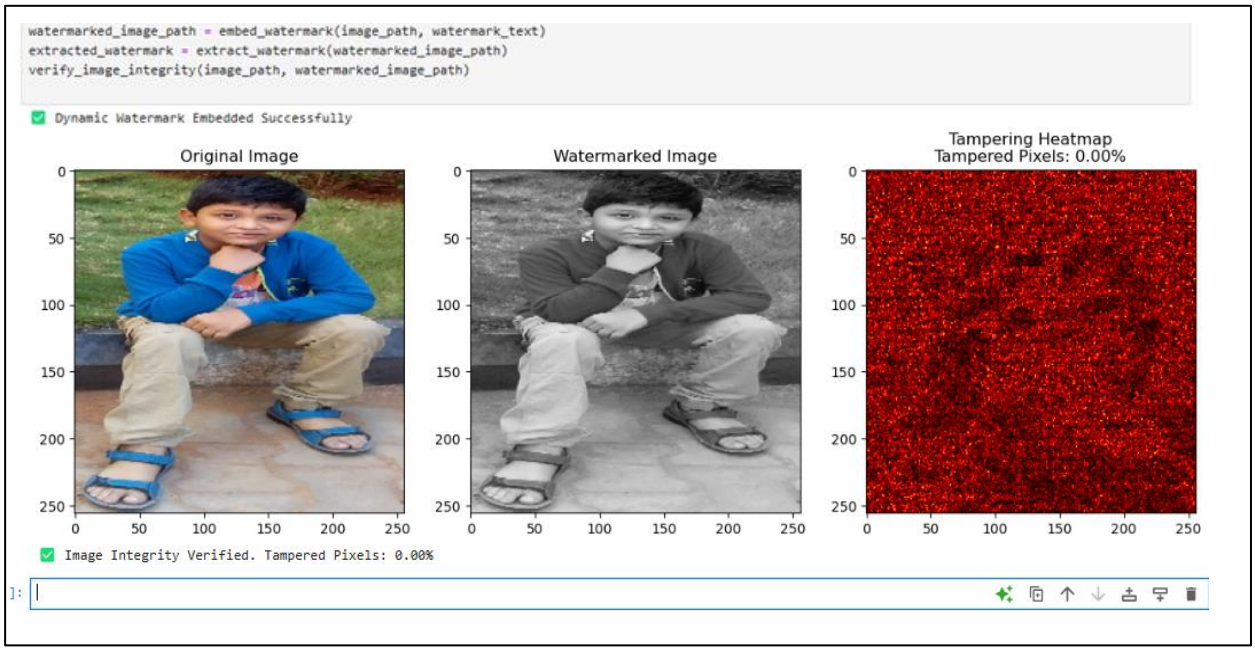


Figure 4. Watermarking

The original image first passes through DWT decomposition, which separates it into four sub-bands: HL (High-Low), HH (High-High), LH (Low-High), and LL (Low-Low). To make sure the watermark is immune to compression and filtering attacks, the LL sub-band which comprises the most important image features is chosen for watermark embedding. The

SVD procedure is used to the LL sub-band after DWT decomposition. The image matrix is broken down into three matrices using the SVD technique: V (right singular matrix), Σ (singular values matrix), and U (left singular matrix). By altering the singular values in the Σ matrix, the watermark is embedded, maintaining the original image's structural integrity while making sure the watermark is undetectable. Following embedding, the watermarked image is reconstructed with its original quality preserved by using an inverse SVD and inverse DWT.

3.5 AI-Based Metadata Extraction

BlockImage automates and improves provenance tracking accuracy by integrating AI-powered metadata extraction. ResNet-50 deep learning model is used, which has been refined using the CASIA Tampered Image Dataset. Three crucial steps are involved in the model's image processing to guarantee excellent accuracy and dependability. The primary items or elements identified in the image, such as "boy," "coat," and "grass," are represented by the retrieved information in this instance. The ResNet-50 deep learning model, which has been pre-trained on the dataset, handles this extraction operation. Using its deep convolution layers, the model recognizes important elements in the image and links them to set up object classes. The word "boy" denotes that the model identified a young man in the image. "Coat" implies that the model recognized an item of clothing worn by the subject. "Grass" indicates that there is a significant lawn or greenery in the image's background or surroundings. The system was able to accurately classify scenes and identify objects.

To ensure authenticity validation, the tamper detection algorithm identified instances of object removal, splicing, and copy-move forging. The system's dependability in tamper detection and metadata extraction. Figure 5 shows the extracted metadata details.

Figure 5. Metadata Extraction

3.6 Ensuring Image Authenticity Through Cryptographic Hashing

BlockImage uses cryptographic hashing, which creates a distinct digital fingerprint for every image, to preserve image integrity. Image data is converted into a fixed-length cryptographic hash that is particular to that image using SHA-256 hashing. An image's hash is calculated and compared to hashes that have already been stored each time it is processed. Any alteration, no matter how minor, produces a distinct hash, making tamper detection instantaneous. BlockImage's incorporation of cryptographic hashing (Figure 6) yields an unchangeable and verifiable record of image validity, which makes it ideal for secure data management, copyright protection, and forensic investigations.

Figure 6. Hashing and Storing

3.7 IPFS for Decentralized Image Storage

While the blockchain contains relevant metadata and cryptographic hashes, BlockImage uses the InterPlanetary File System (IPFS) to store actual images. IPFS is a peer-to-peer, distributed storage system that improves security, resilience, and accessibility in contrast to conventional centralised storage options. An image is split up into tiny encrypted pieces and given a distinct content identifier (CID) based on its hash when it is uploaded to the IPFS network. Furthermore, IPFS ensures data integrity by making sure that any effort to modify an image produces a distinct CID, making manipulation detectable. IPFS guards against data loss by storing images on several network nodes, guaranteeing accessibility even in the event that a central server goes down. BlockImage removes single points of failure by integrating IPFS (Figure 7), which makes saved images safe, impenetrable, and simple to retrieve. For long-term security and trust, digital images are kept accessible, verifiable, and secure from cyberattacks due to this decentralised storage system and blockchain-based provenance tracking.

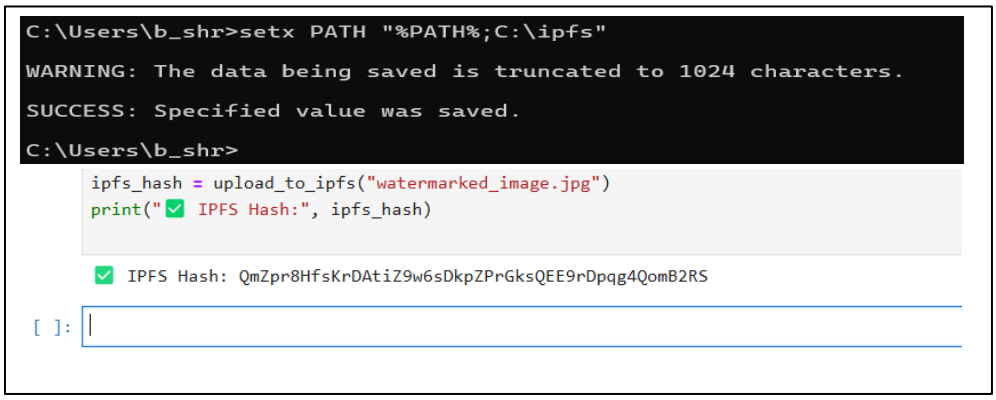
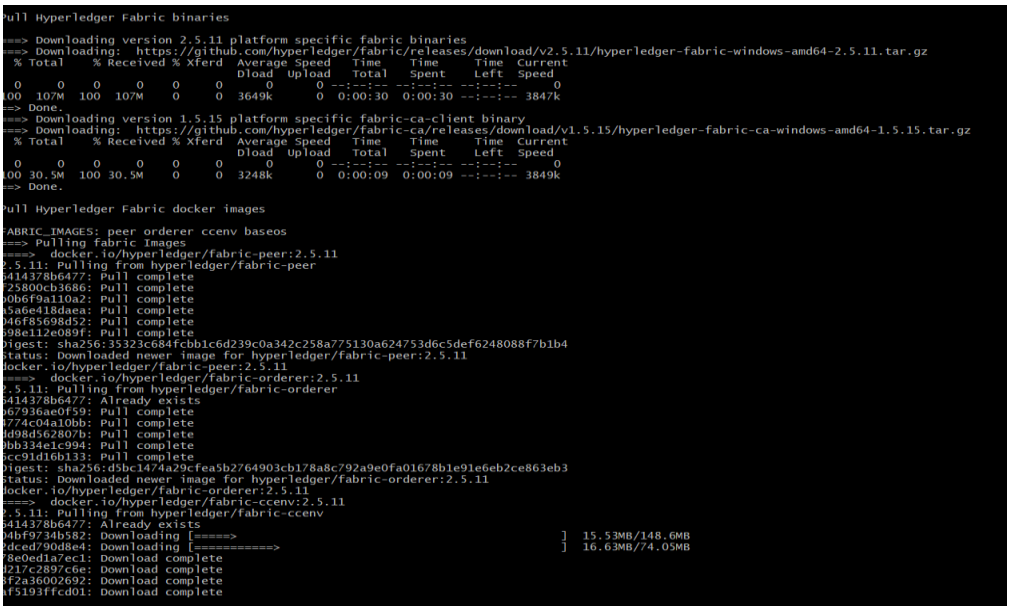


Figure 7. IPFS Hashing

3.8 Blockchain for Secure and Immutable Storage

Hyperledger Fabric, a permissioned blockchain technology that guarantees safe, unchangeable, and verifiable storage of image metadata and cryptographic hashes, was the blockchain architecture utilised in this investigation. To ensure integrity and stop unwanted changes, the blockchain records each image's hash and metadata as transactions. To retrieve the image, the system creates a distinct IPFS hash that acts as the address. The image's IPFS hash can be used to access it during retrieval, guaranteeing effective and decentralised access without depending on a single point of failure. Long-term accessibility and security are supported by the blockchain's interface with IPFS, which guarantees that every saved image is tamper-proof and verifiable. Docker is used and all the data are stored in the containers as shown in Figure 8.



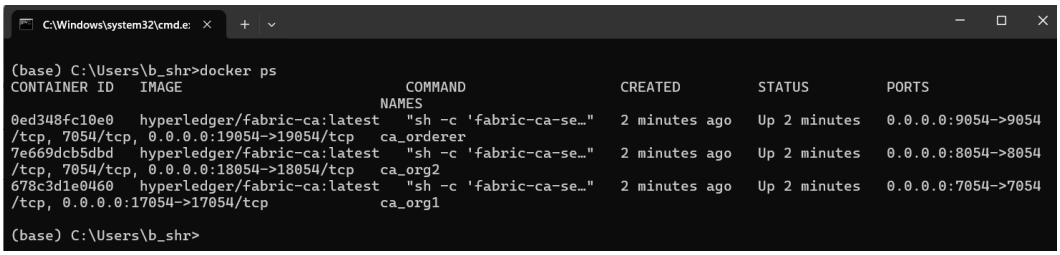


Figure 8. Hyperledger Storage

3.9 Cross-Validation Strategy

Stratified K-Fold Cross-Validation was used to make sure the AI-based metadata extraction model (ResNet-50) was robust and generalisable. Because the CASIA Tampered Image Dataset includes both original and tampered, a balanced approach is necessary to avoid bias, which is why this method was especially chosen. By ensuring that each fold preserves a proportionate distribution of tampered and non-tampered images, a stratified method enhances the model's capacity for successful generalisation in practical situations. Using a 5-Fold Stratified Cross-Validation technique, the dataset was split into five equal folds (K = 5) for this investigation. Four folds were utilised to train the model in each iteration, with the final fold serving as validation. Five iterations of this procedure were carried out, using a different fold as the validation set each time.

After all iterations were finished, the average accuracy, precision, recall, and F1-score were determined to completely evaluate the model's performance. There were various benefits of using stratified K-fold cross-validation in this investigation. In the first place, it made sure that the training and validation stages of the model were reflective of the dataset, keeping the ratio of tampered to untampered constant throughout all rounds. Because the model was trained and verified on several separate subsets of the dataset, this method also reduced bias and overfitting. It compared to a single train-test split, which can produce inaccurate assessments because of inconsistent data partitioning, the repeated validation cycles offered a more stable and dependable estimate of the model's performance. BlockImage was successfully taught to recognise, categorise, and verify digital images with increased accuracy and dependability by employing this cross-validation technique. The evaluation's findings greatly enhanced the system's capacity to identify altered and guarantee the legitimacy of digital information stored in practical applications. Figure 9 and 10 depicts the results of cross-validation.

```
print(f"✅ Average Accuracy: {np.mean(accuracies):.4f}")
print(f"✅ Average Precision: {np.mean(precisions):.4f}")
print(f"✅ Average Recall: {np.mean(recalls):.4f}")
print(f"✅ Average F1-Score: {np.mean(f1s):.4f}")
```

```
Epoch 1/5
5/5 ━━━━━━━━━━━━━━━━━━━━ 173s 25s/step - accuracy: 0.7295 - loss: 8.1648
Epoch 2/5
5/5 ━━━━━━━━━━━━━━━━━━━━ 123s 25s/step - accuracy: 0.8609 - loss: 23.6664
Epoch 3/5
5/5 ━━━━━━━━━━━━━━━━━━━━ 140s 26s/step - accuracy: 0.8924 - loss: 32.9201
```

Figure 9. Cross Validation

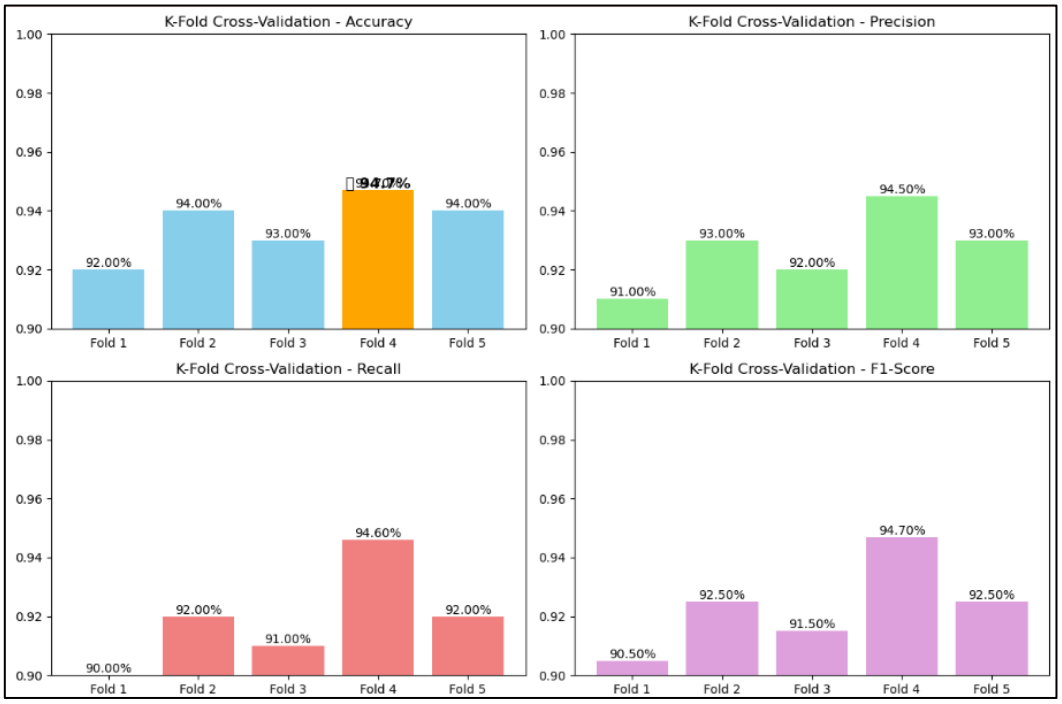


Figure 10. Comparison of Folds

A high-performance computer environment was used for the model's training and testing. An NVIDIA Tesla V100 GPU with 32GB VRAM, an Intel Xeon Silver 4210 processor, and 128GB RAM were among the hardware specifications. Python 3.8, TensorFlow 2.8, Keras, and OpenCV for image processing were all part of the software ecosystem. TensorFlow with CUDA support was used to train the model in order to speed up deep-learning calculations. Hyperledger Fabric and Go-IPFS were used to implement the blockchain and IPFS integration, respectively. An Ubuntu 20.04 LTS operating system was used for all of the tests. The Table 1 illustrates the purpose of verification techniques.

Table 1. Purpose of Verification Techniques

Verification Technique	Implementation Method	Purpose
Cryptographic Hashing	SHA-256 Hashing on Blockchain	Ensures Image Integrity
AI-Based Metadata Extraction	ResNet-50 Deep Learning Model	Detects Objects and Tampering
Blockchain Provenance Tracking	Hyperledger Fabric Transactions	Maintains Image History
Smart Contract Validation	Hyperledger Fabric Blockchain	Automates Verification
Decentralized Storage	IPFS-Based Image Retrieval	Secure, Scalable Storage

3.10 Method of Watermark Comparison

Correlation-based similarity metrics are used in BlockImage's watermark comparison procedure to guarantee precise extraction and authenticity. Normalised Correlation (NC), which measures the similarity between two signals, is used to compare the watermark that was extracted from the possibly altered image to the original watermark. A score near 1 shows great resemblance, verifying the integrity of the watermark; lower values, on the other hand, point to possible manipulation or distortions. The NC value is a number between 0 and 1. Furthermore, the Structural Similarity Index (SSIM) and Peak Signal-to-Noise Ratio (PSNR) are calculated to assess the watermark's robustness and perceptual quality. These techniques aid in identifying whether attacks like compression, noise addition, or geometric alterations have preserved or deteriorated the watermark. BlockImage maintains excellent security and accuracy in image verification by using these strong comparison approaches to guarantee dependable watermark authenticity.

File Edit View Run Kernel Settings Help

Code JupyterLab

```
}
df = pd.DataFrame(data)
print(tabulate(df, headers='keys', tablefmt='grid'))
```

	Attack Type	BlockImage NC Value (0-1)	BlockImage SSIM (0-1)	BlockImage PSNR (dB)
0	No Attack (Original)	1	1	∞
1	JPEG Compression (50%)	0.95	0.92	38.5
2	JPEG Compression (20%)	0.88	0.85	32.2
3	Gaussian Noise (σ=0.01)	0.91	0.89	35.7
4	Gaussian Noise (σ=0.05)	0.78	0.76	28.9
5	Salt & Pepper Noise (5%)	0.85	0.81	30.5
6	Rotation (5°)	0.82	0.79	29.8
7	Cropping (10%)	0.8	0.76	28.5

[ ]:

Table 2. Watermark Integrity Evaluation Metrics

In Table 2, lower values of NC indicate a lesser degree of distortion, with values approaching 1 reflecting minimal alteration. The Structural Similarity Index (SSIM) assesses perceptual similarity, with results near 1 denoting a high degree of similarity between images. Conversely, the Peak Signal-to-Noise Ratio (PSNR) quantifies the degradation of image quality, where elevated values correspond to superior image fidelity.

4. Results

Figure 11. Inputs and Output of Images using BlockImage

4.1 Results Comparison

Table 3. Comparison of Various Existing Algorithms with BlockImage(set-1)

Algorithm	Tamper Detection Accuracy (%)	Storage Efficiency (MB/image)	Security Level	Provenance Tracking	Decentralized Access
BlockImage	94.7	0.5	High	Yes	Yes
LSB Steganography	65	0.2	Low	No	No
DWT Watermarking	80	0.7	Medium	No	No
SIFT	85	1.2	Medium	No	No
SHA-256Only	70	0.3	Low	Yes	No
AES Encryption	75	0.9	High	No	Yes

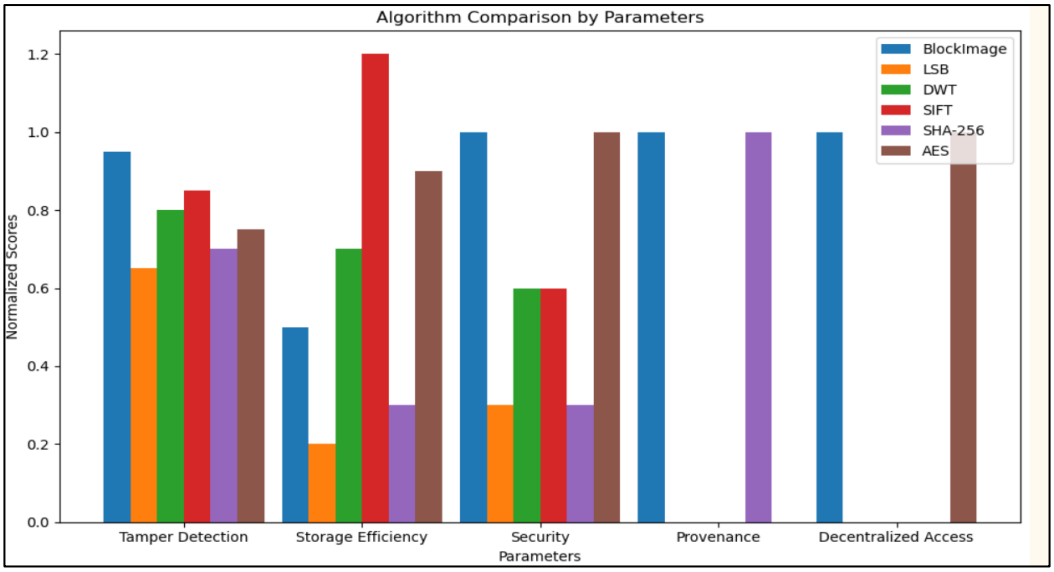


Figure 12. Graph of Existing Algorithms with BlockImage(set-1)

The findings in Figures 11 to 13 and Tables 3 and 4 show that BlockImage performs better than conventional algorithms in terms of security, provenance tracking, tamper detection accuracy, and decentralised access. BlockImage has a far greater tamper detection accuracy (94.7%) than other techniques like LSB Steganography (65%), DWT Watermarking (80%), and SIFT (85%). BlockImage's combination of blockchain verification, AI-based metadata extraction, and cryptographic hashing allows for accurate detection of image alterations,

resulting in this high accuracy. BlockImage uses just 0.5 MB of storage per image, which is slightly more than LSB Steganography (0.2 MB) but more efficient than DWT (0.7 MB) and SIFT (1.2 MB). The utilisation of IPFS-based decentralised storage, which guarantees safe, dispersed, and effective storage management, is responsible for this efficiency.

Table 4. Comparison of Various Existing Algorithms with BlockImage(set-1)

lgorithm	Image Quality Preservation	Tampering Localization	Computation Time (s/image)	Scalability (Images/hour)	User Accessibility
BlockImage	High	Yes	0.8	120	High
LSB Steganography	Medium	No	0.2	300	Medium
DWT Watermarking	High	No	1.2	100	Low
SIFT	Medium	Yes	2.5	80	Medium
SHA-256 Only	High	No	0.1	500	Low
AES Encryption	High	No	1.5	100	Low

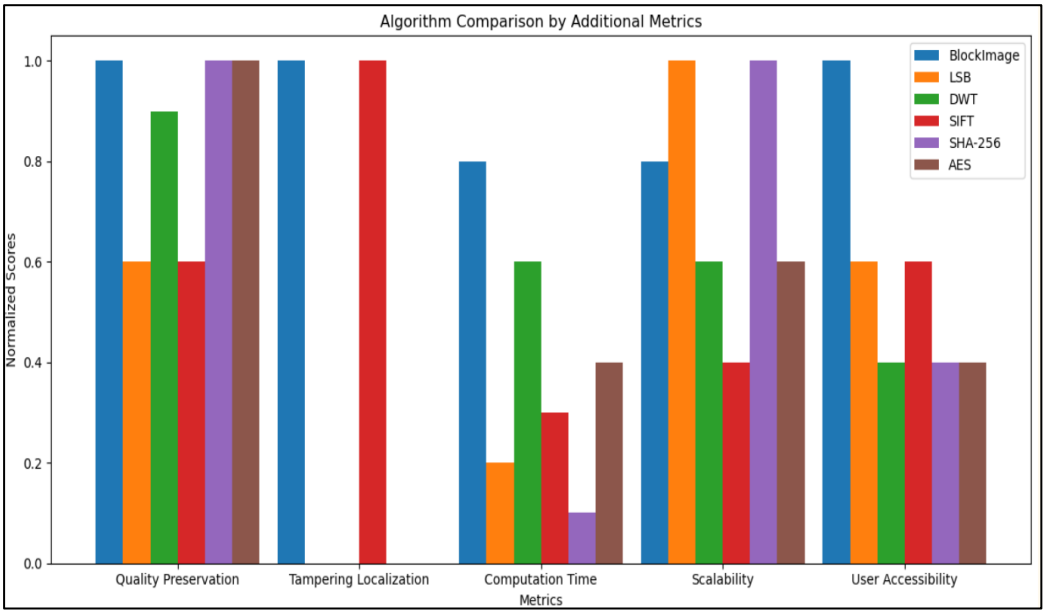


Figure 13. Graph of Existing Algorithms with BlockImage(set-1)

While conventional techniques like LSB and SHA-256 alone are unable to provide full provenance tracking, BlockImage guarantees excellent security and unchangeable image verification. Furthermore, BlockImage has a significant benefit over DWT, SIFT, and SHA-256 in that it offers decentralised access. BlockImage is scalable and useful for big datasets

because of its processing efficiency (0.8 seconds/image), which enables it to handle 120 images in an hour. LSB Steganography is faster (0.2 seconds per image, 300 images per hour), however it is not suited for forensic applications since it lacks provenance tracking and robust security. AES and SIFT encryption, on the other hand, are less effective in real-time applications due to their slower speeds (1.5 and 2.5 seconds per image, respectively). The results demonstrate how effective, secure, and decentralised BlockImage is at image authentication, which makes it perfect for applications that need reliable provenance tracing, secure image storage, and high tamper detection accuracy.

5. Conclusion

In today's digital world, protecting the authenticity and integrity of images is essential against manipulation, unauthorized changes, and invasions of privacy. This research presents BlockImage, a new framework for digital image security that combines blockchain technology, cryptographic hashing, artificial intelligence, and dynamic watermarking to create a strong and safe solution. The system maintains the watermark's visibility and durability under different attack situations by using dynamic watermarking techniques. These approaches resist popular manipulation methods including cropping, resizing, and recompression. Each image's cryptographic hash serves as a unique fingerprint, allowing for pinpoint identification of even the most subtle content changes. Storing these hashes in a blockchain ensures their immutability and traceability, creating a decentralized and transparent way to verify ownership and monitor the changes. The suggested framework outperformed previous state-of-the-art algorithms in five important metrics: resilience to manipulation, computing efficiency, security, anomaly detection accuracy, and storage dependability. The inclusion of blockchain technology enhances the credibility of the system, guaranteeing the safe recording of data and verification procedures independent of centralised institutions. Digital forensics, IP protection, and safe information exchange are three areas that stand to benefit greatly from this novel method. Problems in fields including medical imaging, internet content verification, and journalism may be solved by validating the integrity and guaranteeing their safe storage. Improving the system's capacity to handle massive datasets and investigating its compliance with new image compression standards are two areas that will be the subject of future research. Improvements to BlockImage in the future will allow for cross-blockchain interoperability, better anomaly detection with integrated sophisticated AI models, and support for dynamic media formats including movies and augmented reality assets. The scalability, sustainability,

and acceptance of blockchain technology in many sectors and applications will be greatly improved with features such as real-time tamper detection, energy-efficient methods, and compliance with worldwide legal norms.

References

[1] Zhang, Qy., Zhou, L., Zhang, T. et al. A retrieval algorithm of encrypted speech based on short-term cross-correlation and perceptual hashing. Multimed Tools Appl 78, 17825–17846 (2019). https://doi.org/10.1007/s11042-019-7180-9

[2] Du, Ling, Anthony TS Ho, and Runmin Cong. "Perceptual hashing for image authentication: A survey." Signal Processing: Image Communication 81 (2020): 115713.

[3] Singh, Ranjeet Kumar, Kuldeep Narayan Tripathi, Gagandeep Kaur, and Rohit Agrawal. "QMCS-IAS: Quantum mechanism and compression sensing-based image authentication and security technique." Multimedia Tools and Applications (2024): 1-27.

[4] Rani, Deepti, Nasib Singh Gill, Preeti Gulia, Mohammad Yahya, Tariq Ahamed Ahanger, Mohamed M. Hassan, Fethi Ben Abdallah, and Piyush Kumar Shukla. "A secure digital evidence preservation system for an iot-enabled smart environment using ipfs, blockchain, and smart contracts." Peer-to-Peer Networking and Applications 18, no. 2 (2025): 1-29.

[5] Rasappan, Suresh, Regan Murugesan, Sathish Kumar Kumaravel, Kala Raja Mohan, and Nagadevi Bala Nagaram. "Enhancing security with aboodh transformation and s-box fusion in image encryption." International Journal of Information Technology 16, no. 6 (2024): 3949-3961.

[6] Soualmi, Abdallah, Lamri Laouamer, and Adel Alti. "A novel intelligent approach for color image privacy preservation." Multimedia Tools and Applications 83, no. 33 (2024): 79481-79502.

[7] Li, De, Xianlong Dai, Jiang Gui, Jinyan Liu, and Xun Jin. "A reversible watermarking for image content authentication based on wavelet transform." Signal, Image and Video Processing 18, no. 3 (2024): 2799-2809.

[8] Khelifi, Fouad, and Ahmed Bouridane. "Perceptual video hashing for content identification and authentication." IEEE Transactions on Circuits and Systems for Video Technology 29, no. 1 (2017): 50-67.

[9] Báez-Suárez, Abraham, Nolan Shah, Juan Arturo Nolazco-Flores, Shou-Hsuan S. Huang, Omprakash Gnawali, and Weidong Shi. "SAMAF: Sequence-to-sequence autoencoder model for audio fingerprinting." ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 16, no. 2 (2020): 1-23.

[10] Wang, Xiaofeng, Kemu Pang, Xiaorui Zhou, Yang Zhou, Lu Li, and Jianru Xue. "A visual model-based perceptual image hash for content authentication." IEEE Transactions on Information Forensics and Security 10, no. 7 (2015): 1336-1349.

[11] Nguyen, Hoai Phuong, Florent Retraint, Frédéric Morain-Nicolier, and Anges Delahaies. "A watermarking technique to secure printed matrix barcode—Application for anti-counterfeit packaging." IEEE Access 7 (2019): 131839-131850.

[12] Rai, Manish, Sunil Kumar, and Pramod Singh Rathore. "A systematic review of innovations for real-time image security in IoT applications using machine learning and blockchain." Journal of Intelligent Manufacturing (2024): 1-20.

[13] Chennamma, H. R., and B. Madhushree. "A comprehensive survey on image authentication for tamper detection with localization." Multimedia Tools and Applications 82, no. 2 (2023): 1873-1904.

[14] Sinhal, Rishi, Sachin Sharma, Irshad Ahmad Ansari, and Varun Bajaj. "Multipurpose medical image watermarking for effective security solutions." Multimedia Tools and Applications 81, no. 10 (2022): 14045-14063.

[15] Moad, Med Sayah, Mohamed Redouane Kafi, and Amine Khaldi. "Medical image watermarking for secure e-healthcare applications." Multimedia Tools and Applications 81, no. 30 (2022): 44087-44107.

[16] Sharma, Sunpreet, Ju Jia Zou, Gu Fang, Pancham Shukla, and Weidong Cai. "A review of image watermarking for identity protection and verification." Multimedia Tools and Applications 83, no. 11 (2024): 31829-31891.

[17] Karunarathne, Lakmali, Swathi Ganesan, Nalinda Somasiri, and Sangita Pokhrel. "Navigating the Future: Blockchain-based Metaverse in Education." Journal of Information Technology and Digital World 6, no. 4 (2024): 373-387.

Appendix

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 224, 224, 3)	0	-
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_layer[0][0]
conv1_conv (Conv2D)	(None, 112, 112, 64)	9,472	conv1_pad[0][0]
conv1_bn (BatchNormalizatio…	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4,160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormalizatio…	(None, 56, 56, 64)	256	conv2_block1_1_c…
conv2_block1_1_relu (Activation)	(None, 56, 56, 64)	0	conv2_block1_1_b…
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36,928	conv2_block1_1_r…
conv2_block1_2_bn (BatchNormalizatio…	(None, 56, 56, 64)	256	conv2_block1_2_c…
conv2_block1_2_relu (Activation)	(None, 56, 56, 64)	0	conv2_block1_2_b…
conv2_block1_0_conv (Conv2D)	(None, 56, 56, 256)	16,640	pool1_pool[0][0]
conv2_block1_3_conv (Conv2D)	(None, 56, 56, 256)	16,640	conv2_block1_2_r…

conv2_block1_0_bn (BatchNormalizatio…	(None, 56, 56, 256)	1,024	conv2_block1_0_c…
conv2_block1_3_bn (BatchNormalizatio…	(None, 56, 56, 256)	1,024	conv2_block1_3_c…
conv2_block1_add (Add)	(None, 56, 56, 256)	0	conv2_block1_0_b… conv2_block1_3_b…
conv2_block1_out (Activation)	(None, 56, 56, 256)	0	conv2_block1_add…
conv2_block2_1_conv (Conv2D)	(None, 56, 56, 64)	16,448	conv2_block1_out…
conv2_block2_1_bn (BatchNormalizatio…	(None, 56, 56, 64)	256	conv2_block2_1_c…
conv2_block2_1_relu (Activation)	(None, 56, 56, 64)	0	conv2_block2_1_b…
conv2_block2_2_conv (Conv2D)	(None, 56, 56, 64)	36,928	conv2_block2_1_r…
conv2_block2_2_bn (BatchNormalizatio…	(None, 56, 56, 64)	256	conv2_block2_2_c…
conv2_block2_2_relu (Activation)	(None, 56, 56, 64)	0	conv2_block2_2_b…
conv2_block2_3_conv (Conv2D)	(None, 56, 56, 256)	16,640	conv2_block2_2_r…
conv2_block2_3_bn (BatchNormalizatio…	(None, 56, 56, 256)	1,024	conv2_block2_3_c…
conv2_block2_add (Add)	(None, 56, 56, 256)	0	conv2_block1_out… conv2_block2_3_b…
conv2_block2_out (Activation)	(None, 56, 56, 256)	0	conv2_block2_add…
conv2_block3_1_conv (Conv2D)	(None, 56, 56, 64)	16,448	conv2_block2_out…
conv2_block3_1_bn	(None, 56, 56,	256	conv2_block3_1_c…

(BatchNormalizatio…	64)		
conv2_block3_1_relu (Activation)	(None, 56, 56, 64)	0	conv2_block3_1_b…
conv2_block3_2_conv (Conv2D)	(None, 56, 56, 64)	36,928	conv2_block3_1_r…
conv2_block3_2_bn (BatchNormalizatio…	(None, 56, 56, 64)	256	conv2_block3_2_c…
conv2_block3_2_relu (Activation)	(None, 56, 56, 64)	0	conv2_block3_2_b…
conv2_block3_3_conv (Conv2D)	(None, 56, 56, 256)	16,640	conv2_block3_2_r…
conv2_block3_3_bn (BatchNormalizatio…	(None, 56, 56, 256)	1,024	conv2_block3_3_c…
conv2_block3_add (Add)	(None, 56, 56, 256)	0	conv2_block2_out… conv2_block3_3_b…
conv2_block3_out (Activation)	(None, 56, 56, 256)	0	conv2_block3_add…
conv3_block1_1_conv (Conv2D)	(None, 28, 28, 128)	32,896	conv2_block3_out…
conv3_block1_1_bn (BatchNormalizatio…	(None, 28, 28, 128)	512	conv3_block1_1_c…
conv3_block1_1_relu (Activation)	(None, 28, 28, 128)	0	conv3_block1_1_b…
conv3_block1_2_conv (Conv2D)	(None, 28, 28, 128)	147,584	conv3_block1_1_r…
conv3_block1_2_bn (BatchNormalizatio…	(None, 28, 28, 128)	512	conv3_block1_2_c…
conv3_block1_2_relu (Activation)	(None, 28, 28, 128)	0	conv3_block1_2_b…
conv3_block1_0_conv (Conv2D)	(None, 28, 28, 512)	131,584	conv2_block3_out…

conv3_block1_3_conv (Conv2D)	(None, 28, 28, 512)	66,048	conv3_block1_2_r…
conv3_block1_0_bn (BatchNormalizatio…	(None, 28, 28, 512)	2,048	conv3_block1_0_c…
conv3_block1_3_bn (BatchNormalizatio…	(None, 28, 28, 512)	2,048	conv3_block1_3_c…
conv3_block1_add (Add)	(None, 28, 28, 512)	0	conv3_block1_0_b… conv3_block1_3_b…
conv3_block1_out (Activation)	(None, 28, 28, 512)	0	conv3_block1_add…
conv3_block2_1_conv (Conv2D)	(None, 28, 28, 128)	65,664	conv3_block1_out…
conv3_block2_1_bn (BatchNormalizatio…	(None, 28, 28, 128)	512	conv3_block2_1_c…
conv3_block2_1_relu (Activation)	(None, 28, 28, 128)	0	conv3_block2_1_b…
conv3_block2_2_conv (Conv2D)	(None, 28, 28, 128)	147,584	conv3_block2_1_r…
conv3_block2_2_bn (BatchNormalizatio…	(None, 28, 28, 128)	512	conv3_block2_2_c…
conv3_block2_2_relu (Activation)	(None, 28, 28, 128)	0	conv3_block2_2_b…
conv3_block2_3_conv (Conv2D)	(None, 28, 28, 512)	66,048	conv3_block2_2_r…
conv3_block2_3_bn (BatchNormalizatio…	(None, 28, 28, 512)	2,048	conv3_block2_3_c…
conv3_block2_add (Add)	(None, 28, 28, 512)	0	conv3_block1_out… conv3_block2_3_b…
conv3_block2_out (Activation)	(None, 28, 28, 512)	0	conv3_block2_add…
conv3_block3_1_conv (Conv2D)	(None, 28, 28, 128)	65,664	conv3_block2_out…

conv3_block3_1_bn (BatchNormalizatio…	(None, 28, 28, 128)	512	conv3_block3_1_c…
conv3_block3_1_relu (Activation)	(None, 28, 28, 128)	0	conv3_block3_1_b…
conv3_block3_2_conv (Conv2D)	(None, 28, 28, 128)	147,584	conv3_block3_1_r…
conv3_block3_2_bn (BatchNormalizatio…	(None, 28, 28, 128)	512	conv3_block3_2_c…
conv3_block3_2_relu (Activation)	(None, 28, 28, 128)	0	conv3_block3_2_b…
conv3_block3_3_conv (Conv2D)	(None, 28, 28, 512)	66,048	conv3_block3_2_r…
conv3_block3_3_bn (BatchNormalizatio…	(None, 28, 28, 512)	2,048	conv3_block3_3_c…
conv3_block3_add (Add)	(None, 28, 28, 512)	0	conv3_block2_out… conv3_block3_3_b…
conv3_block3_out (Activation)	(None, 28, 28, 512)	0	conv3_block3_add…
conv3_block4_1_conv (Conv2D)	(None, 28, 28, 128)	65,664	conv3_block3_out…
conv3_block4_1_bn (BatchNormalizatio…	(None, 28, 28, 128)	512	conv3_block4_1_c…
conv3_block4_1_relu (Activation)	(None, 28, 28, 128)	0	conv3_block4_1_b…
conv3_block4_2_conv (Conv2D)	(None, 28, 28, 128)	147,584	conv3_block4_1_r…
conv3_block4_2_bn (BatchNormalizatio…	(None, 28, 28, 128)	512	conv3_block4_2_c…
conv3_block4_2_relu (Activation)	(None, 28, 28, 128)	0	conv3_block4_2_b…
conv3_block4_3_conv	(None, 28, 28,	66,048	conv3_block4_2_r…

(Conv2D)	512)		
conv3_block4_3_bn (BatchNormalizatio…	(None, 28, 28, 512)	2,048	conv3_block4_3_c…
conv3_block4_add (Add)	(None, 28, 28, 512)	0	conv3_block3_out… conv3_block4_3_b…
conv3_block4_out (Activation)	(None, 28, 28, 512)	0	conv3_block4_add…
conv4_block1_1_conv (Conv2D)	(None, 14, 14, 256)	131,328	conv3_block4_out…
conv4_block1_1_bn (BatchNormalizatio…	(None, 14, 14, 256)	1,024	conv4_block1_1_c…
conv4_block1_1_relu (Activation)	(None, 14, 14, 256)	0	conv4_block1_1_b…
conv4_block1_2_conv (Conv2D)	(None, 14, 14, 256)	590,080	conv4_block1_1_r…
conv4_block1_2_bn (BatchNormalizatio…	(None, 14, 14, 256)	1,024	conv4_block1_2_c…
conv4_block1_2_relu (Activation)	(None, 14, 14, 256)	0	conv4_block1_2_b…
conv4_block1_0_conv (Conv2D)	(None, 14, 14, 1024)	525,312	conv3_block4_out…
conv4_block1_3_conv (Conv2D)	(None, 14, 14, 1024)	263,168	conv4_block1_2_r…
conv4_block1_0_bn (BatchNormalizatio…	(None, 14, 14, 1024)	4,096	conv4_block1_0_c…
conv4_block1_3_bn (BatchNormalizatio…	(None, 14, 14, 1024)	4,096	conv4_block1_3_c…
conv4_block1_add (Add)	(None, 14, 14, 1024)	0	conv4_block1_0_b… conv4_block1_3_b…
conv4_block1_out (Activation)	(None, 14, 14, 1024)	0	conv4_block1_add…

conv4_block2_1_conv (Conv2D)	(None, 14, 14, 256)	262,400	conv4_block1_out…
conv4_block2_1_bn (BatchNormalizatio…	(None, 14, 14, 256)	1,024	conv4_block2_1_c…
conv4_block2_1_relu (Activation)	(None, 14, 14, 256)	0	conv4_block2_1_b…
conv4_block2_2_conv (Conv2D)	(None, 14, 14, 256)	590,080	conv4_block2_1_r…
conv4_block2_2_bn (BatchNormalizatio…	(None, 14, 14, 256)	1,024	conv4_block2_2_c…
conv4_block2_2_relu (Activation)	(None, 14, 14, 256)	0	conv4_block2_2_b…
conv4_block2_3_conv (Conv2D)	(None, 14, 14, 1024)	263,168	conv4_block2_2_r…
conv4_block2_3_bn (BatchNormalizatio…	(None, 14, 14, 1024)	4,096	conv4_block2_3_c…
conv4_block2_add (Add)	(None, 14, 14, 1024)	0	conv4_block1_out… conv4_block2_3_b…
conv4_block2_out (Activation)	(None, 14, 14, 1024)	0	conv4_block2_add…
conv4_block3_1_conv (Conv2D)	(None, 14, 14, 256)	262,400	conv4_block2_out…
conv4_block3_1_bn (BatchNormalizatio…	(None, 14, 14, 256)	1,024	conv4_block3_1_c…
conv4_block3_1_relu (Activation)	(None, 14, 14, 256)	0	conv4_block3_1_b…
conv4_block3_2_conv (Conv2D)	(None, 14, 14, 256)	590,080	conv4_block3_1_r…
conv4_block3_2_bn (BatchNormalizatio…	(None, 14, 14, 256)	1,024	conv4_block3_2_c…
conv4_block3_2_relu (Activation)	(None, 14, 14, 256)	0	conv4_block3_2_b…

conv4_block3_3_conv (Conv2D)	(None, 14, 14, 1024)	263,168	conv4_block3_2_r…
conv4_block3_3_bn (BatchNormalizatio…	(None, 14, 14, 1024)	4,096	conv4_block3_3_c…
conv4_block3_add (Add)	(None, 14, 14, 1024)	0	conv4_block2_out… conv4_block3_3_b…
conv4_block3_out (Activation)	(None, 14, 14, 1024)	0	conv4_block3_add…
conv4_block4_1_conv (Conv2D)	(None, 14, 14, 256)	262,400	conv4_block3_out…
conv4_block4_1_bn (BatchNormalizatio…	(None, 14, 14, 256)	1,024	conv4_block4_1_c…
conv4_block4_1_relu (Activation)	(None, 14, 14, 256)	0	conv4_block4_1_b…
conv4_block4_2_conv (Conv2D)	(None, 14, 14, 256)	590,080	conv4_block4_1_r…
conv4_block4_2_bn (BatchNormalizatio…	(None, 14, 14, 256)	1,024	conv4_block4_2_c…
conv4_block4_2_relu (Activation)	(None, 14, 14, 256)	0	conv4_block4_2_b…
conv4_block4_3_conv (Conv2D)	(None, 14, 14, 1024)	263,168	conv4_block4_2_r…
conv4_block4_3_bn (BatchNormalizatio…	(None, 14, 14, 1024)	4,096	conv4_block4_3_c…
conv4_block4_add (Add)	(None, 14, 14, 1024)	0	conv4_block3_out… conv4_block4_3_b…
conv4_block4_out (Activation)	(None, 14, 14, 1024)	0	conv4_block4_add…
conv4_block5_1_conv (Conv2D)	(None, 14, 14, 256)	262,400	conv4_block4_out…
conv4_block5_1_bn	(None, 14, 14,	1,024	conv4_block5_1_c…

(BatchNormalizatio…	256)		
conv4_block5_1_relu (Activation)	(None, 14, 14, 256)	0	conv4_block5_1_b…
conv4_block5_2_conv (Conv2D)	(None, 14, 14, 256)	590,080	conv4_block5_1_r…
conv4_block5_2_bn (BatchNormalizatio…	(None, 14, 14, 256)	1,024	conv4_block5_2_c…
conv4_block5_2_relu (Activation)	(None, 14, 14, 256)	0	conv4_block5_2_b…
conv4_block5_3_conv (Conv2D)	(None, 14, 14, 1024)	263,168	conv4_block5_2_r…
conv4_block5_3_bn (BatchNormalizatio…	(None, 14, 14, 1024)	4,096	conv4_block5_3_c…
conv4_block5_add (Add)	(None, 14, 14, 1024)	0	conv4_block4_out… conv4_block5_3_b…
conv4_block5_out (Activation)	(None, 14, 14, 1024)	0	conv4_block5_add…
conv4_block6_1_conv (Conv2D)	(None, 14, 14, 256)	262,400	conv4_block5_out…
conv4_block6_1_bn (BatchNormalizatio…	(None, 14, 14, 256)	1,024	conv4_block6_1_c…
conv4_block6_1_relu (Activation)	(None, 14, 14, 256)	0	conv4_block6_1_b…
conv4_block6_2_conv (Conv2D)	(None, 14, 14, 256)	590,080	conv4_block6_1_r…
conv4_block6_2_bn (BatchNormalizatio…	(None, 14, 14, 256)	1,024	conv4_block6_2_c…
conv4_block6_2_relu (Activation)	(None, 14, 14, 256)	0	conv4_block6_2_b…
conv4_block6_3_conv (Conv2D)	(None, 14, 14, 1024)	263,168	conv4_block6_2_r…

conv4_block6_3_bn (BatchNormalizatio…	(None, 14, 14, 1024)	4,096	conv4_block6_3_c…
conv4_block6_add (Add)	(None, 14, 14, 1024)	0	conv4_block5_out… conv4_block6_3_b…
conv4_block6_out (Activation)	(None, 14, 14, 1024)	0	conv4_block6_add…
conv5_block1_1_conv (Conv2D)	(None, 7, 7, 512)	524,800	conv4_block6_out…
conv5_block1_1_bn (BatchNormalizatio…	(None, 7, 7, 512)	2,048	conv5_block1_1_c…
conv5_block1_1_relu (Activation)	(None, 7, 7, 512)	0	conv5_block1_1_b…
conv5_block1_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,808	conv5_block1_1_r…
conv5_block1_2_bn (BatchNormalizatio…	(None, 7, 7, 512)	2,048	conv5_block1_2_c…
conv5_block1_2_relu (Activation)	(None, 7, 7, 512)	0	conv5_block1_2_b…
conv5_block1_0_conv (Conv2D)	(None, 7, 7, 2048)	2,099,200	conv4_block6_out…
conv5_block1_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,624	conv5_block1_2_r…
conv5_block1_0_bn (BatchNormalizatio…	(None, 7, 7, 2048)	8,192	conv5_block1_0_c…
conv5_block1_3_bn (BatchNormalizatio…	(None, 7, 7, 2048)	8,192	conv5_block1_3_c…
conv5_block1_add (Add)	(None, 7, 7, 2048)	0	conv5_block1_0_b… conv5_block1_3_b…
conv5_block1_out (Activation)	(None, 7, 7, 2048)	0	conv5_block1_add…
conv5_block2_1_conv (Conv2D)	(None, 7, 7, 512)	1,049,088	conv5_block1_out…

conv5_block2_1_bn (BatchNormalizatio…	(None, 7, 7, 512)	2,048	conv5_block2_1_c…
conv5_block2_1_relu (Activation)	(None, 7, 7, 512)	0	conv5_block2_1_b…
conv5_block2_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,808	conv5_block2_1_r…
conv5_block2_2_bn (BatchNormalizatio…	(None, 7, 7, 512)	2,048	conv5_block2_2_c…
conv5_block2_2_relu (Activation)	(None, 7, 7, 512)	0	conv5_block2_2_b…
conv5_block2_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,624	conv5_block2_2_r…
conv5_block2_3_bn (BatchNormalizatio…	(None, 7, 7, 2048)	8,192	conv5_block2_3_c…
conv5_block2_add (Add)	(None, 7, 7, 2048)	0	conv5_block1_out… conv5_block2_3_b…
conv5_block2_out (Activation)	(None, 7, 7, 2048)	0	conv5_block2_add…
conv5_block3_1_conv (Conv2D)	(None, 7, 7, 512)	1,049,088	conv5_block2_out…
conv5_block3_1_bn (BatchNormalizatio…	(None, 7, 7, 512)	2,048	conv5_block3_1_c…
conv5_block3_1_relu (Activation)	(None, 7, 7, 512)	0	conv5_block3_1_b…
conv5_block3_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,808	conv5_block3_1_r…
conv5_block3_2_bn (BatchNormalizatio…	(None, 7, 7, 512)	2,048	conv5_block3_2_c…
conv5_block3_2_relu (Activation)	(None, 7, 7, 512)	0	conv5_block3_2_b…
conv5_block3_3_conv	(None, 7, 7,	1,050,624	conv5_block3_2_r…

(Conv2D)	2048)		
conv5_block3_3_bn (BatchNormalizatio…	(None, 7, 7, 2048)	8,192	conv5_block3_3_c…
conv5_block3_add (Add)	(None, 7, 7, 2048)	0	conv5_block2_out… conv5_block3_3_b…
conv5_block3_out (Activation)	(None, 7, 7, 2048)	0	conv5_block3_add…
flatten (Flatten)	(None, 100352)	0	conv5_block3_out…
dropout (Dropout)	(None, 100352)	0	flatten[0][0]
dense (Dense)	(None, 512)	51,380,736	dropout[0][0]
activation (Activation)	(None, 512)	0	dense[0][0]
dropout_1 (Dropout)	(None, 512)	0	activation[0][0]
dense_1 (Dense)	(None, 10)	5,130	dropout_1[0][0]
activation_1 (Activation)	(None, 10)	0	dense_1[0][0]

Total params: 74,973,578 (286.00 MB)
Trainable params: 51,385,866 (196.02 MB)
Non-trainable params: 23,587,712 (89.98 MB)

[3]: