Embedded hash codes for image similarity detection and tamper proofing

Nazir, Sajid; Kaleem, Mohammad

Published in:
2022 International Conference on Emerging Trends in Electrical, Control, and Telecommunication Engineering (ETECTE)

DOI:
10.1109/ETECTE55893.2022.10007355

Publication date:
2023

Document Version
Author accepted manuscript

Embedded Hash Codes for Image Similarity Detection and Tamper Proofing

Sajid Nazir
Department of Computing
Glasgow Caledonian University
Glasgow, UK
sajid.nazir@gcu.ac.uk

Mohammad Kaleem
Department of Electrical and Computer Engineering
COMSATS University
Islamabad, Pakistan
mkaleem@comsats.edu.pk

***Abstract*— The image and video traffic on the Internet has increased due to the ease and abundance of the image capture and sharing services. However, this can sometimes result in violations of copyright and image tampering for various purposes. Finding differences between images has importance for image retrieval, aiding forensics, disease diagnosis, and environmental changes etc. Many similarity measures have been proposed that use different features extractable from the detected image. In some cases it is important to determine if the set of images differ and if so then to what extent. Digital Rights Management (DRM) has gained prominence and importance due to widespread availability of tools to modify images. Sometimes the need arises to claim the rightful ownership of an image. The blockchain through the use of hash codes can be used to determine tampering similar to its success in many industries where privacy and security of the documents is important.**

In this paper, we embed the image hash code in the JPEG image header. The proposed system simplifies the hashing and encryption process by processing only the image data without which a reconstruction is not possible. The results show that the image data can thus be shared without the risk of any undetected tampering or misuse.

Keywords—Digital Rights Management, JPEG, tamper-proofing, encryption, image similarity, blockchain

I. INTRODUCTION

The images and videos being communicated and shared over the Internet have proliferated due to the operational ease and widespread availability of the image capture devices. Almost every mobile device has a camera, and the power that visual information plays has resulted in it being the target of manipulation through editing software to create tampered image and video versions [1], [2]. There is thus a need for simplified detection techniques that can be used by the owner of the image and video content to avoid or stop any tampering after it is communicated and available in the public domain.

The authentication of the images received against the communicated image is a major problem [3]. This assumes added importance for images and videos in the healthcare and military domains. There are many image similarity-based techniques that can often determine if the image data has been manipulated. Images are represented internally by a matrix of numbers and hence the similarity measures are based on calculating a distance measured between the two images. The use of machine learning techniques for the image anomaly detection are presented in [1].

Detecting the image similarity is also important in case of steganography where the intention is to hide any kind of data, such as textual or image, within an image so that the changes are visually imperceptible [4]. In steganography, it should be very difficult to detect or determine the hidden secret [4], [5].

The image tamper protection could be for the purposes of copyright, or DRM. In the case of copyright, the image data is available in plain and accessible for anyone that would like to use it subject to the copyright conditions. The images in such cases might have a hidden or visible watermark inserted with an aim to make it as difficult as possible to be removed [5]. In the case of DRM, the actual data is only made available to the authorized users and others to prevent unauthorized copying and duplication. This can be ensured through encryption so an unauthorized user will not be able to interpret the data unless they purchase the rights to view that copyrighted content.

Blockchain technology provides tamper proofing of the information and achieves this by using hash codes and cryptography. Blockchain is a decentralized storage and the blocks in a chain are interlinked through the hash codes facilitating prevention of any tampering to the blockchain data [6]. The data may reside within the blockchain, usual and practical for small data, or may be stored off-chain in case of larger files such as image content [2].

The hash function has the property that it will change drastically even for a small changes in the information over which it is calculated. This is used in digital signatures to detect changes and also used for file downloads to detect that no change has occurred during the download process. For medical systems, it is important to protect the data for security and privacy [7]. The data protection was achieved by storing the medical data in the Interplanetary File System (IPFS) and its hash in the blockchain [7]. An object in the IPFS has a size limit of 256 kB.

In this paper, we focus on using the hash function to detect if the image data has been manipulated. This detection could be for the purpose of determining change during communications, for copyright purposes, or to retrieve an image from image database etc. The focus is on providing details of embedding image hash to the image header for ensuring data preservation for tamper-detection and tamper-proofing.

This paper also covers some image similarity measures to illustrate their usage for determining similarity or a distance measure. The application of encrypting the entire image or a subset are also described.

Rest of the paper is organized as follows: Section II describes the related work. Section III covers the proposed techniques for similarity detection and tamper proofing. The results and discussion are provided in Section IV and the conclusion in Section V.

II. Related Work

The use of blockchain can provide a public decentralized storage for the image signature for later authentication [2]. A scheme was proposed to embed the transaction ID within the JPEG file header in COM (comment) marker, by first converting the image data to YC_bC_R and then coding the 8x8 blocks using the Discrete Cosine Transform (DCT) [2]. An image signature was derived by keeping the bits from AC and DC coefficients to calculate a signature of size MN/8, which was then protected with Advanced Encryption Standard (AES-128) [2]. The image signature was then stored in a blockchain, BigchainDB and the signature was able to locate the tampered location [2].

A system Imagechain was proposed to hash the whole image file data so that it could not be altered [5]. The system could work for any image type and the image data itself was not stored within the blockchain. The images were linked as a chain with the hash of next image stored within an image [5].

Video tampering is often performed by decompressing the compressed video and then recompressing it [6]. The authors in [6] devised a method for video integrity verification based on blockchain to store the segmented video data with Secure Hash Algorithm (SHA-256) [6].

A method for forgery prevention of vehicular black box image data showed use of consensus algorithm to decrease the IPFS and download delay time by 63% [8]. The video was uploaded to IPFS, and the video hash of 64 bytes was stored on the blockchain [8].

Blockchain and watermark were used for preventing image misuse and preserving DRM [9]. DC coefficients were used for embedding the watermark whereas the DRM information (comprising date, location, owner information) was embedded [9]. A consortium blockchain was used to store the artwork and the DRM information [9].

Hash codes were used to protect the image data integrity and to detect any data tampering [10]. The proposed scheme used image projections and distributed source codes to keep the hash size small [10]. The intended recipient could use the hash values between the received and the original image to detect Mean Square Error (MSE) distortion [10].

An idea of forming a blockchain of all the blocks in the image as a linear chain was proposed that showed effective resistance against vector quantization (VQ) [11]. This was performed to hide the watermark within the image with better localization [11]. The image alignment for block-wise image hash codes and tampering detection was proposed in [3].

III. Proposed System

We used the Python language for the image similarity, hash, and encryption implementations.

A. Image Data and Header

Joint Photographic Expert Group (JPEG) is the most commonly used image coding standard. Although it provides for lossless compression, to save on storage and communications bandwidth, commonly lossy compression is used. A JPEG image file contains a JPEG header followed by the image data. The image header contains important information such as image width, height, size etc., and there are some markers which are optional to be used [12]. The availability of optional markers makes it possible for the programmers to embed a calculated image value, such as an image hash to be embedded within the image header without affecting the image encryption or decryption process.

JPEG standard [12] compresses the raw pixel data in 8X8 blocks, and the compressed representation contains, the DC coefficient, followed by AC coefficient. DC coefficient for each 8x8 block is a single number but can be used to recreate a coarse representation of the image whereas the AC coefficients are required for generating finer image details [13].

We propose to embed the image data hash within the image header and also highlight the significance of using only the DC coefficients to calculate the image hash, and to embed the resulting hash within the JPEG header. This can help to identify the tampering location using a simplified process and is similar to the scheme by Dobre et al. [2].

We used two images (481x321 pixels) from the Berkeley dataset [14] as shown in Fig. 1. The choice of the images from a public dataset was to enable the reproducibility of the results, however, the proposed system for embedding the hash codes is independent of the image resolution.

B. Image Similarity Measures

The simplest comparison to differentiate two similar images capturing the same view, for example in time lapse, could be to compare their file size. An image with more details would have a larger size compared to a same size image with less finer details. However, obviously two entirely dissimilar images might have the same or similar size in bytes. Although the techniques such as structural similarity and others will have a zero difference between an image and its exact copy, it is possible that two images which are different have the same similarity index [15], [16].

Such calculations based on the image are also commonly used to determine that an image has not changed during a download or transmission, for example, [17] makes available the checksums for each of the image to be downloaded. These and similar metrics are not perfect; therefore, we propose the use of a hash function.

Although there are many similarity measures but we have chosen only two, that is, Structural Similarity (SSIM) and Mean Squared Error (MSE) [18]. MSE and SSIM are widely used as a distance or similarity measure in compression, segmentation, and matching tasks [16]. We use these two similarity measures to illustrate that with these and other such similarity measures, it is hard to determine their relationship to the nature and extent of changes to an image.

C. Hash

An image hash is a compact representation of an image content [3]. The image hash can be used to detect any tampering with the image data [10].

A hash function will generate a different hash value even if a single bit within the image data is changed. It also has an additional advantage of generating a different hash value for the same image, if it was rotated, inverted, or transformed in any other way.

SHA-256 generates an output bit length of 256 bits and belongs to the family of SHA-2 hash algorithms [19]. SHA-256 algorithm has the main steps such as a main loop, padding, and word expansion [19].

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

Fig. 1. Sample images (a) Original Pyramid image (b) Original Zebra image (c) Pyramid image with 16 pixels set to zero, (129, 184) to (132, 187) (d) Zebra image with 16 pixels set to zero, (280, 176) to (283, 179) (e) Pyramid image after a horizontal flip (f) Pyramid image after a horizontal flip (g) Pyramid image with a triangle added (h) Zebra image tampered by copy-paste of the isolated zebra from the original image.

TABLE I. RESULTS FOR DIFFERENT IMAGE SIMILARITY MEASURES

Similarity Measure	Pyramid Image				Zebra Image			
	Original	16-pixels Set as 0	Horizontal Flipped Image	With a Block Added	Original	16-pixels Set as 0	Horizontal Flipped Image	With a Zebra Added
MSE	0.0	2.75	1420.23	853.52	0.0	72.95	2524.78	19.25
SSIM	1.0	0.99	0.388	0.97	1.0	0.85	0.22	0.98

TABLE II. HASH CODES FOR THE ORIGINAL ZEBRA IMAGE AND THE ZEBRA IMAGE WITH 1 BIT CHANGED

Type	Image	Hash Codes
MD5	Original	3d831ee8ff9cacbec573225cf495057e
	1-bit flip	a9a28b866742562adf9eef0a624b337e
SHA-1	Original	9faf8685800333e78378c8c5c54efd5fb22dccd0
	1-bit flip	49589284e10cd4fb0d753d5ea19da46e5cc49313
SHA-256	Original	e2e4b4f56ed5132ff336de37d01c1c1ab2ae0cd8df4c0f8ae5e63a6c1ef7b66a
	1-bit flip	071f6819f5f2e0f51cf77821e964c1fc3574e32c758995197cc253d766014eb0

D. *Encrypting the Image Data*

The image data itself may or may not be required to be secured depending on the type of the application use-case. With image and video data being shared, the use of an embedded hash code of the image data within the image header is all that is required. This is useful in cases requiring to prove that it has not been tampered and that it is the original image, which can easily be ascertained.

There are applications which require the actual image data to be encrypted and made accessible only to the authorized users with a shared key or password.

In such cases, where the image data itself is required to be encrypted, we propose use of AES-256 which is a strong cryptographic algorithm. This can be applied over the entire image or a subset of it, such as, to only encrypt the DC coefficients from the image data. This process will involve the use of cryptographic keys which can be shared with the authorized users only.

IV. RESULTS

A. *Image Similarity Measures*

The results for SSIM and MSE are shown in Table I. In case of SSIM, it can be seen that the comparison of an image with itself (a perfect match) yields a value of 1.0, whereas with an increase in the image differences, the value approaches 0.0. Similarly, different values are obtained for MSE for the various changes to the images as shown in Table I.

B. *Hash*

The different hash codes were obtained (Table I) for the original Zebra.jpg, and for a modified Zebra.jpg (by flipping one bit in the second last byte from the end of the image file from hexadecimal A6 to A7). It can be seen that just a single bit flip causes a large change in the hash. Any other bit location from the image data can be chosen to be flipped, which will have similar effect on the hash code.

Other than the sizes of the resulting hash codes, each of the hash type changes for a smallest change in the image data. However, we recommend use of SHA-256 as the same code cannot be generated from two different data values, whereas the same degree of protection is not possible with the other two types shown in Table II.

The use of APP0 and COM markers for embedding the 256 bits hash code (from Table II) within the image header, is illustrated in Fig. 2. Although embedding the SHA-256 hash code can use either of these markers but the method for both of the APP0 and COM marker is shown in Fig. 2.

The start of APP0 marker (shown in red) is indicated as FFE0 and must have a size of 16 bytes including, 4A 46 00 00 (JFxx) at the end [20], hence three markers are required. The COM marker (shown in green) starts with FF FE followed by the SHA-256 hash code, and the four hex numbers 00 00 are required. The other markers appearing in Fig. 2 are shown in blue.

However, a more robust scheme would be to encode the image hash code (Fig. 2) with AES-256 encryption. The result of encrypting the hash code using PyCrypto library is shown in Fig. 3.

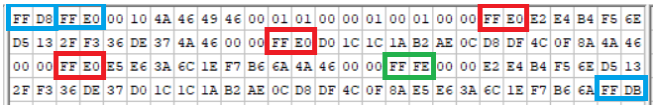

Fig. 2. The use of markers APP0 and COM to embed hash codes.

b'25mlPMP+zP55y5Ui5DZ9ToLnC10qtDvT8fucSkpSaLjP
VSH549ihsJO+lOZ8SO1PwqHM6Kc9OokrxWJbRsonfrY/
QI0pQ9/eIKy3gnFmcGKwXtvr5xXLVPYF0FUcz1q0'E2E4
B4F56ED5132FF336DE37D01C1C1AB2AE0CD8DF4C0F
8AE5E63A6C1EF7B66A

Fig. 3. Output of AES-256 encryption.

C. Encrypting the Image Data

We have described the use of a hash for providing the information within the image, however, for applications that require to hide the actual image data, AES-256 encryption can be used. The image data being large can be stored in an IPFS storage and the blockchain can be used to store the image hash. The hash code may be calculated only for the DC coefficients of a JPEG image rather than for whole of the image data.

D. Comparison

We have described the use of a hash for providing the information within the image, however, applications requiring to hide the calculated image hash code can also use a key or password to encrypt the image hash code before embedding it in the image.

The work by Dobre et al. [2] requires few complex operations to calculate the image signature and embedding it in the image. In comparison the proposed scheme of using only the DC components from an image could be achieved through data parsing which is a simplified operation.

If there is a temporal or spatial relationship between images, such as, time lapse images and video sequences, then the proposed scheme can be extended for finding the extent of the change. The proposed scheme can also be extended to videos by regarding these as sequences of images.

Compared to the other proposed schemes in the literature that store hash external to the image or video [21], storing the hash within the image is a scalable and maintainable solution.

V. Conclusion

In this paper, we propose a method for calculating the image hash and storing it within the image header. The hash provides a strong tamper protection as even a single bit change to the image is detectable. The proposed method is extensible to finding hash only for the important image data, such as DC coefficients for a JPEG image, and storing it within the image header. This would not only determine if the image has been modified during transmission, but can also infer the location of any unauthorized changes. The proposed method can make it easier to share the image data and at the same time ensure that it cannot be misused by tampering.

As part of future work, we will use cryptography to encrypt the image data and manage the security and sharing using blockchain as a Digital Rights Management mechanism.

References

[1] K. A. P. da Costa, J. P. Papa, L. A. Passos, D. Colombo, J. D. Ser, K. Muhammad and V. H. C. de Albuquerque, “A critical literature survey and prospects on tampering and anomaly detection in image data,” *Applied Soft Computing Journal*, vol. 97, 106727, 2020.

[2] R. A. Dobre, R. O. Preda, C. C. Oprea and I. Pirnog, “Authentication of JPEG Images on the Blockchain,” *2018 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*, 2018, pp. 211-215, doi: 10.1109/ICCAIRO.2018.00042.

[3] S. Battiato, G. M. Farinella, E. Messina and G. Puglisi, “Robust Image Alignment for Tampering Detection,” *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, August 2012.

[4] M. A. Alsarayreh, M. A. Alia and K. Abu Maria, “A novel image steganographic system based on exact matching algorithm and key-dependent data,” *Journal of Theoretical and Applied Information Technology*, 2017.

[5] K. Koptyra and M. R. Ogiela, “Imagechain—Application of Blockchain Technology for Images,” *Sensors*, vol. 21, 2021. https://dx.doi.org/10.3390/s21010082.

[6] S. Ghimire, J. Y. Choi and B. Lee, “Using Blockchain for Improved Video Integrity Verification,” *IEEE Transactions on Multimedia*, vol. 22, no. 1, Jan 2020.

[7] J. Sun, L. Ren, S. Wang and X. Yao, “A blockchain-based framework for electronic medical records sharing with fine-grained access control,” *PLoS ONE*, vol. 15, no. 10, e0239946. https://doi.org/10.1371/journal.pone.0239946

[8] D. Na and S. Park, "Lightweight blockchain to solve forgery and privacy issues of vehicle image data," *2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2021, pp. 37-40, doi: 10.23919/APNOMS52696.2021.9562586.

[9] M. Zhaofeng, H. Weihua and G. Hongmin, A new blockchain-based trusted DRM scheme for built-in content protection, *EURASIP Journal on Image and Video Processing*, vol. 91, , https://doi.org/10.1186/s13640-018-0327-1

[10] M. Tagliasacchi, G. Valenzise and S. Tubaro, “Hash-Based Identification of Sparse Image Tampering,” *IEEE Transactions on Image Processing*, vol. 18, no. 11, Nov 2009.

[11] W. Fang, Y. Wang and X. Wang, “Image Tampering Location and Restoration Watermarking Based on Blockchain Technology,” M. Atiquzzaman et al. (Eds.): BDCPS 2019, AISC 1117, pp. 420–428, 2020. https://doi.org/10.1007/978-981-15-2568-1_58

[12] Information Technology – Digital Compression and Coding of Continuous-tone Still Images–Requirements and Guidelines, CCITT T.81 The International (09/92) Telegraph and Telephone Consultative Committee. https://www.w3.org/Graphics/JPEG/itu-t81.pdf

[13] S. Nazir, O. Alzubi, M. Kaleem and H. Hamdoun, “Image subset communication for resource-constrained applications in wireless sensor networks,” *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 28, no. 5, 2020.

[14] The Berkeley Segmentation Dataset and Benchmark: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

[15] J. Davies, “Implementing SSL/TLS using cryptography and PKI,” John Wiley and Sons, 2011.

[16] G. Palubinskas, “Image similarity/distance measures: what is really behind MSE and SSIM?,” *International Journal of Image and Data Fusion*, vol. 8, no. 1, pp.32-53, 2017.

[17] The USC-SIPI Image Database: https://sipi.usc.edu/database/

[18] Scikit-image: Image processing in Python, https://scikit-image.org/docs/stable/api/skimage.metrics.html

[19] S. H. Rohini, G. Nischal and R. B. Shettar, “Power-Efficient Approach to Optimize SHA-256 Bit Using Reversible Logic,” *Computing and Network Sustainability*, 2019, pp. 275-283, Springer, Singapore.

[20] E. Hamilton, JPEG File Interchange Format, version 1.02, Sep 1992 https://www.w3.org/Graphics/JPEG/jfif3.pdf.

[21] R. Kamal, E. E. -D. Hemdan and N. El-Fishway, “Video Integrity Verification based on Blockchain,” *2021 International Conference on Electronic Engineering (ICEEM)*, 2021, pp. 1-5, doi: 10.1109/ICEEM52022.2021.9480653.