future internet

MDPI

Article

An Image Authentication Scheme Using Merkle Tree Mechanisms

Yi-Cheng Chen, Yueh-Peng Chou and Yung-Chen Chou *

Department of Computer Science and Information Engineering (CSIE), Asia University, No. 500, Liufeng Road, Wufeng District, Taichung City 41354, Taiwan

* Correspondence: yungchen@gmail.com; Tel.: +886-4-23323456

Received: 14 May 2019; Accepted: 1 July 2019; Published: 6 July 2019

Abstract: Research on digital image processing has become quite popular and rapid in recent years, and scholars have proposed various image verification mechanisms. Similarly , blockchain technology has also become very popular in recent years. This paper proposes a new image verification mechanism based on the Merkle tree technique in the blockchain. The Merkle tree root in the blockchain mechanism provides a reliable environment for storage of image features. In image verification, the verification of each image can be performed by the Merkle tree mechanism to obtain the hash value of the Merkle tree node on the path. In addition, the method combines the Inter-Planetary File System (IPFS) to improve the availability of images. The main purpose of this paper is to achieve the goal of image integrity verification. The proposed method can not only verify the integrity of the image but also restore the tampered area in the case of image tampering. Since the proposed method employs the blockchain mechanism, the image verification mechanism does not need third party resources . The verification method is performed by each node in the blockchain network. The experimental results demonstrate that the proposed method successfully achieved the goal of image authentication and tampered area restoration.

Keywords: image authentication; Merkle tree; hash function; blockchain; distributed storage

1. Introduction

According to the advanced development of information technologies, the internet has become an indispensable part of data delivery in our daily life. The creation of content such as digital images is getting easier than before. It can be produced by a digital camera, scanner, computer software, etc. The image might be stolen, have tampered content, or even be misappropriated by an unexpected user. Thus, how to organize, store, and deliver the images becomes an important issue because the internet is not a secure environment [1].

Digital image content integrity has been an important research topic of digital image management in recent years. Digital images are easy to copy and transmit anywhere over public computer networks, and can be spread across the world very fast. Also, it could easily be tampered with and transformed by the unexpected user. Thus, the integrity of a digital image is determined by checking whether an image has been tampered with or not. Generally speaking, the integrity is not only checking the difference from the original image but also to point out the area of the tampered part, in the case of an image that has been tampered with. For example, the correctness of license plate information on a vehicle violation photo is quite important. That means the vehicle violation image cannot be tampered with [2–4].

Normally, the tampered area detecting can be done by comparing the difference between original image and tampered image. This is not an efficient way to authenticate the integrity of an image. Many researchers use an information hiding technique to embed the features of a digital image into

the image to achieve the goal of image integrity protection. When image integrity checking is required, the verification information can be extracted from the image, then applied to their image authentication procedure to check the tampered area on the image.

Bitcoin has become a famous cryptocurrency in recent years, the developers combined open source software and cryptography, and it operates on a P2P network and decentralized databases. The system allows the nodes of the entire P2P network to be used to reach a network agreement according to the seed file, ensuring that the currency is secure and trustworthy in its issuance, management, and circulation [5–7]. Actually, the blockchain technique can be applied to many applications in the real world such as IoT, FinTech, virtual currency, etc.

We try to design an image management system which is composed by Merkle tree technology in the blockchain, Inter-Planetary File System (IPFS) decentralized storage system, and image verification scheme [8]. IPFS is a P2P distributed storage system that attempts to connect all computing devices to the same file system for large amounts of data storage [8]. However, IPFS did not provide the function of image integrity verification. We were inspired by the blockchain technique and IPFS to create a blockchain based image integrity verification scheme.

In this paper, Merkle tree technology is adopted to generate the image verification information, and IPFS is used to manage the verification data and to form the proposed image verification scheme. In order to solve the drawback of centralized image management, the proposed method utilizes the IPFS protocol to handle image authentication data management with decentralized properties. Here, the proposed method not only provides the image authentication ability but can also restore the tampered area. According to our design, it is not necessary to have a trusted third party to manage the image verification data. The verification method is performed by each node in the blockchain network.

The remaining sections of this paper are organized as follows. Section 2 shows some background knowledge related to the blockchain, Merkle tree, and image authentication. Section 3 shows the detail of the proposed image authentication steps. The simulation results are illustrated in Section 4. Finally, we make some conclusions in Section 5.

2. Background

In this section, we will introduce some background knowledge of our proposed method. Section 2.1 introduces the concept of blockchain. Section 2.2 describes the Hash function. The main idea of our proposed method, Merkle Tree, is detailed in Section 2.3. Then, the Image Authentication and Decentralized Storage are described in Sections 2.4 and 2.5, respectively.

2.1. Blockchain Technology

Blockchain [9,10] technology is composed of multiple technologies, namely cryptography, mathematics, consensus algorithms, and economic models. It is a secure, shared and distributed ledger (database) that records all transactional data as what are called blocks. The blockchain use P2P networks and consensus mechanisms to solve the problem of distributed data synchronization, and so it is not necessary to have a centralized trusted authority. Bitcoin is one of most famous applications using the blockchain technique [5,11,12].

The blockchain data structure is defined as an ordered back-linked record of blocks of transactions. It can be in a database or saved as a file. Each block can be recognized by a SHA256 cryptographic hash algorithm on the header of the block. Generally, the block is composed of two parts—the main data and the header. The main data contains a list of transactions, while the header includes a hash of the previous and current block, Merkle Root, timestamp, nonce, and other information. Figure 1 shows the structure of blockchain. A blockchain has the following six features:

1. **Decentralized**: The basic structure of blockchain, the network is decentralized, meaning it has no need to rely on any server or node. The data can be recorded, stored, and updated by a group of nodes.

2. **Transparency**: When data is transmitted on the blockchain, records on each node are open and transparent—this is the reason that blockchain can be trusted.
3. **Open Source**: The records of blockchain systems are publicly verifiable for any user, and the user can also use the blockchain system to develop any application.
4. **Autonomy**: Based on the consensus mechanism, each node in the blockchain can transmit or update data to each other in a secure situation. This idea is from a single entity to the entire system so that no one can interfere with it.
5. **Immutable**: Any records will always be kept and stored and will not be altered unless the remaining nodes have a record where greater than 51% of the record will be changed.
6. **Anonymity**: The blockchain technology solves the problem of trust on the node-to-node, so the data transmission or the transaction can be hidden, and only when the trader's blockchain address is known will it be exposed.

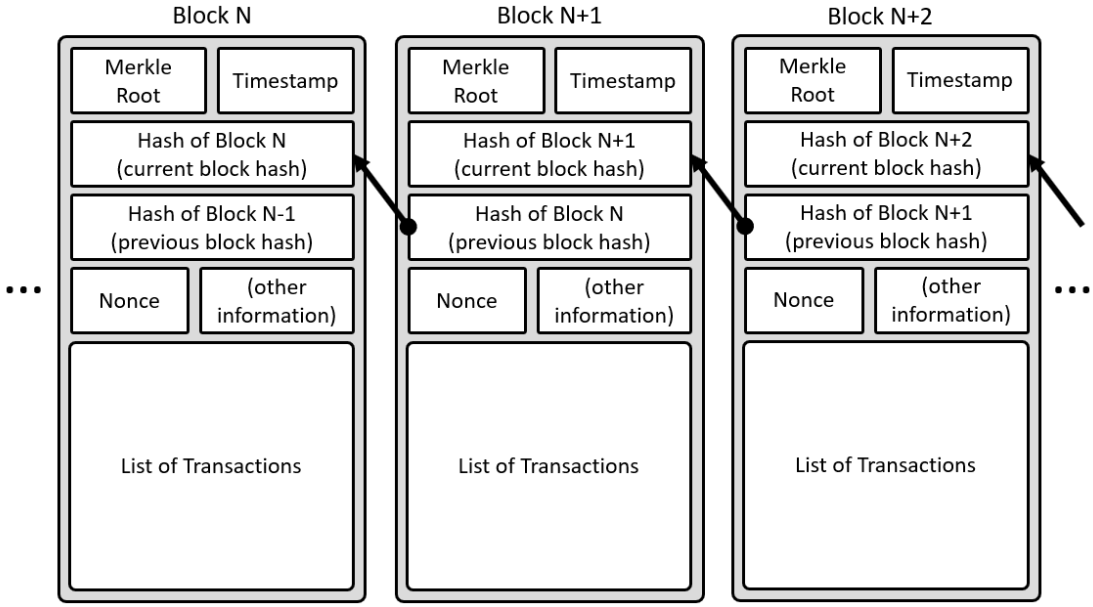


Figure 1. Structure of the blockchain.

2.2. Hash Function

A hash function can be used for any size of data and corresponds to a fixed-size hash output. It is a one-way function where the returned values are called hash values, hash codes, digests, or simply hashes (Figure 2). Hash functions are often used in computer software for efficiently searching data. Also, the hash function can effectively avoid duplicate data being written into a file or database. Hash functions are also commonly used in an encryption algorithm. The hash function in a cryptosystem, it can quickly generate the hash value of an input and check the correctness corresponding to the data in the cryptosystem. Also, the one-way hash function has no inverse function to convert the hash value back to the original data [13].

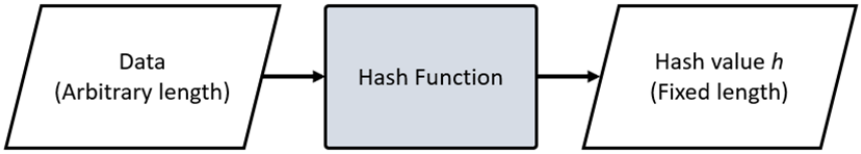


Figure 2. Hash Function.

2.3. Merkle Tree

The Merkle tree was presented by Ralph Merkle in 1979 [14]. It is a useful tree structure for several application areas, especially in cryptography. Merkle trees have been an essential key to data verification throughout the history of computers. Their structure helps to verify the consistency of data content. Its architecture helps to speed up security authentication in big data applications. It is a complete binary tree, and each node is to hash the value from its child node. Merkle tree structure is shown in Figure 3 [15,16].

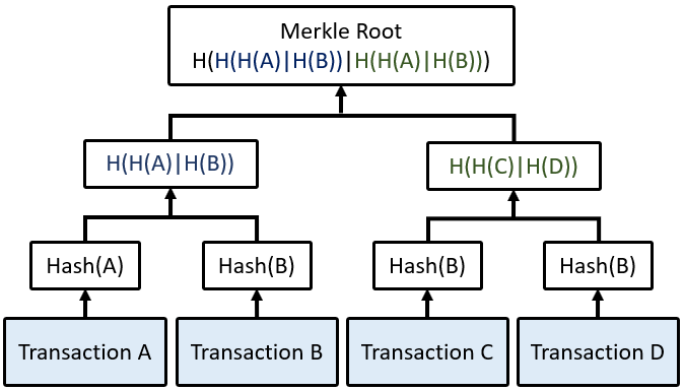


Figure 3. Merkle tree structure.

Merkle trees are also a fundamental part of blockchain technology. For a block, the Merkle root comes from a hashing transaction and pairing two transactions to hash and generate the upper level tree node. By doing so, it will get one hash to store that is deterministic based on the hashes of all the underlying transactions. This single hash is called the Merkle root (Figure 4).

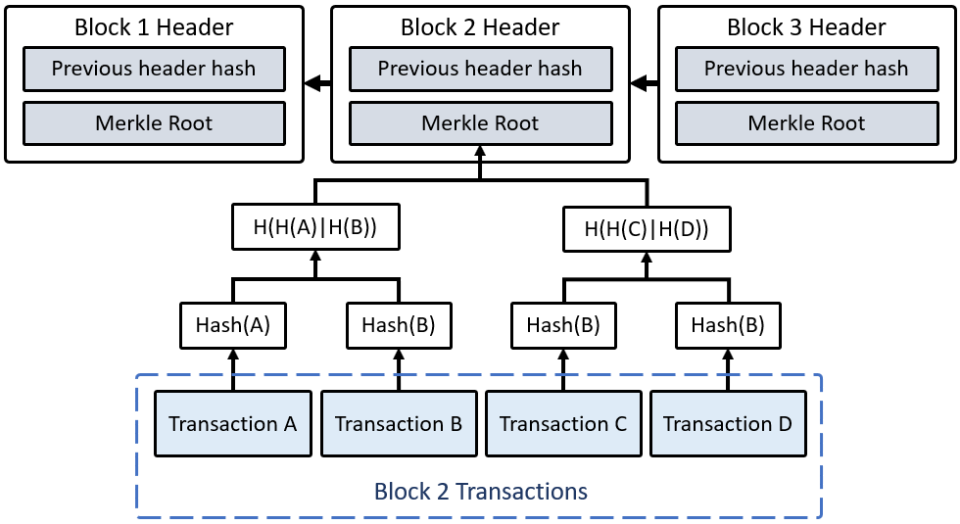


Figure 4. The architecture of Merkle tree in the blockchain.

In the blockchain, each block has a Merkle root stored in the block header. Merkle tree allows every node on the network to verify individual transaction without having to download and validate the entire block. If a copy of the block in the blockchain networks has the same Merkle root to another, then the transactions in that block are the same. Even a bit of incorrect data would lead to vastly different Merkle roots because of the properties of the hash. Therefore, it is not necessary to verify the amount of required information (Figure 5).

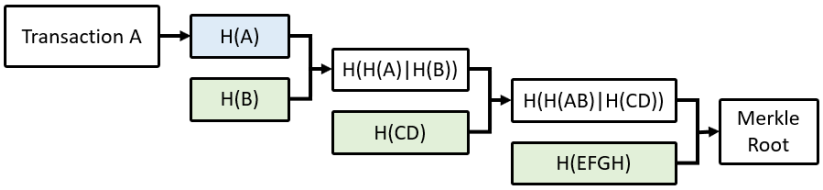


Figure 5. The verification of Merkle tree.

2.4. Image Authentication

In order to securely transmit the digital image over the internet, image watermarking and the techniques of image encryption and authentication are the most important research issues in recent years. Therefore , when the receiver receives the image, whether the image is tampered with or not, it would be verified. So, the validity of the information source is extremely important. For protecting the integrity of images, several methods have been proposed [17]. These methods include traditional cryptography, fragile and semi-fragile watermarks, and digital signatures based on image content [1,18]. Cox et al. [3] collected several applications of watermarking, like broadcast monitoring, owner identification, proof of ownership, authentication, transactional watermarking, copy control, and covert communication.

D. Bhowmik et al. [19] proposed a novel watermark-based multimedia blockchain framework. The watermark contains two parts of information: First, a cryptographic hash containing the history records (blockchain transactions); second, the image hash of the original media content. Later, D. Bhowmik et al. [20] proposed a blockchain framework for JPEG images that has its own block content. The image related information is hidden in the header of a JPEG image, and the image is stored in a multimedia server. The information for verifying copyright were stored in the blockchain framework.

Storing image verification information on the blockchain is a good strategy, but the image is still stored in the centralized manner, or kept by the owner, which will affect the availability of image management. Thus, we proposed a method to solve this problem.

2.5. Decentralized Storage: IPFS

IPFS stands for Interplanetary File System. It is an open-source, peer-to-peer (P2P) distributed hypermedia protocol that aims to function as a ubiquitous file system for all computing devices. IPFS is similar to the World Wide Web but looks more like a single BitTorrent swarm that exchanges objects within a single Git repository. In other words, IPFS provides a high throughput, content-addressable (block) storage system and content-related hyperlinks. It combines a decentralized hash table, data exchange, and a self-certified namespace, which also forms a generalized Merkle architecture. IPFS has no single point of failure and nodes do not need to trust each other, distributed content delivery can also save bandwidth consumption [8,21].

3. Proposed Method

This section is composed of three parts. The first part is focused on the Merkle tree generation followed by the image verification step in the second part. Finally, the last part shows the key stage of image recovery in detail.

3.1. Merkle Tree Generation

The first part in the generation of the Merkle tree process is as follows, the image is sliced into several blocks, then encrypted and uploaded to the IPFS system. Before generating the Merkle tree, we have to do some image processing. First, we take k MSBs (Most Significant Bit) of each pixel from the original image I_{org} as the new image $I^{'}$. Next, the image is sliced $I^{'}$ into non-overlapping blocks B_i, where the block size is $N \times N$ pixels. After that, each block is encrypted B_i and uploaded to the IPFS system. As the IPFS is a decentralized file system that anyone who knows the file address can

browse to to access the files, for file protecting purposes, any encrypting system can be adopted to our method, such as XOR Cipher, RSA, AES, etc. The process is shown in Figure 6.

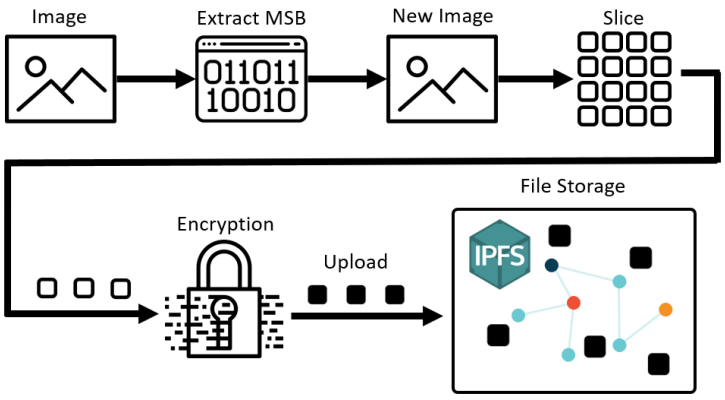


Figure 6. Process of image slicing and encryption.

When the file is uploaded to IPFS, the system will apply a hash function to generate the hash value of the image file (also known as the fingerprint to identify the file). Then, the fingerprint is used as transactions in the blockchain TX_i (see Figure 7), and each TX_i as the leaves, to build up the Merkle tree M (see Section 2.3). Meanwhile, the Merkle Root is stored into the blockchain (see Figure 8). Here, using the fingerprint as the transaction method has two advantages. First, the cipher image block is placed in a decentralized system to maintain its availability and remove the redundant files in the whole network. Second, the fingerprint is stored in the blockchain, IPFS can use the fingerprint of the file to locate its address, so that image recovery can also be performed. The key steps of the proposed Merkle tree generation method are summarized as Procedure 1.

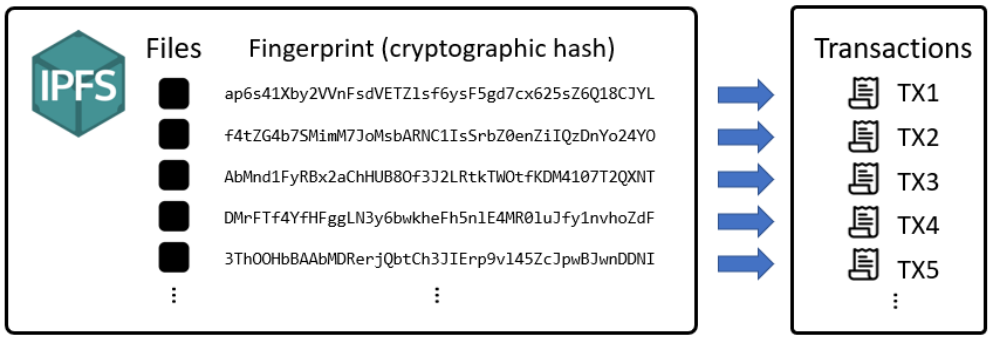


Figure 7. Inter-Planetary File System (IPFS) fingerprint as the transactions in the blockchain (leaves of Merkle tree).

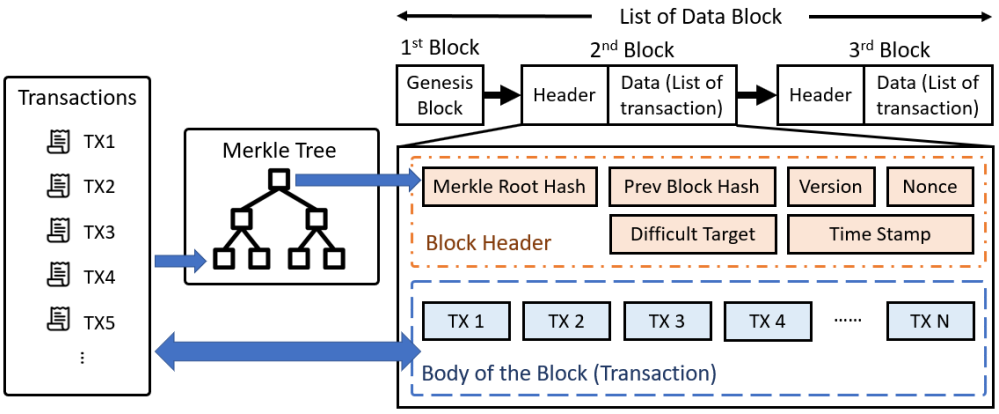


Figure 8. The process of storing data in the blockchain.

Procedure 1 Merkle Tree Generation

Input: Original Image I_{org}
Output: Merkle tree M

1: Take k MSBs from each pixel in original image I_{org} as a New Image $I^{'}$.
2: Slice Image $I^{'}$ into non-overlapping block B_i, where the block size is $N \times N$ pixels.
3: Encrypt B_i by ${B_i}^e = E_K(B_i)$.
4: Upload ${B_i}^e$ to IPFS and take the unique hash (fingerprint) as TX_i, generated by IPFS.
5: Generate Merkle tree M by taking TX_i as the leaf node of Merkle tree.
6: Store TX_i and the Merkle Root of M in the blockchain.

3.2. Image Verification

To verify the integrity is an important function of our proposed method. The proposed image verification procedure can not only detect whether the image was tampered with or not, but can also point out the tampered area on the image. Let the test image be denoted as I_{unv} (the image that needs to be verified), and the test result is a set and denoted as $R = \{r_i \mid i = 1, 2, 3, ..., n\}$, where r_i is the test result corresponding to block B_i and $r_i = \{0, 1\}$, and n is the number of total blocks of I_{unv}.

First, generate Merkle tree M_{unv} for I_{unv} by using the Merkle tree generation method (refer to Section 3.1). Then, get Merkle tree M_{org} corresponding to the original image from the block in the blockchain. Finally, generate comparing results r_i which use M_{org} to compare with M_{unv}. In case a leaf node is different, then set the value of r_i to 1, otherwise set it to 0. The process is shown in Figure 9. The key steps of the proposed image verification method are summarized as Procedure 2.

Procedure 2 Image Verification

Input: Unverified Image I_{unv}
Output: Set of the tampered result $R_i = (r_1, r_2, ..., r_n)$,

$$r_i \begin{cases} 0, & \textit{right block} \\ 1, & \textit{tampered block} . \end{cases}$$

1: Generate Merkle tree M_{unv} of I_{unv} by using the Merkle tree generation method (refer to Section 3.1).
2: Retrieve M_{org} corresponding to original image form the blockchain.
3: Check the size of M_{org} and M_{unv} Merkle trees, if both of them have the same size then go to Step 4, else, set each r_i to 1 then stop procedure.
4: Compare M_{org} and M_{unv} recursively from root to the tree left. Check left side child hashes followed by right side child hashes. If the node matches, then set the value of r_i to 0. If the node does not match, and it is the bottom node of the Merkle tree, then set the value of r_i corresponding to the node to 1, else, compare the child nodes.
5: Go to Step 4, until all nodes have been checked.

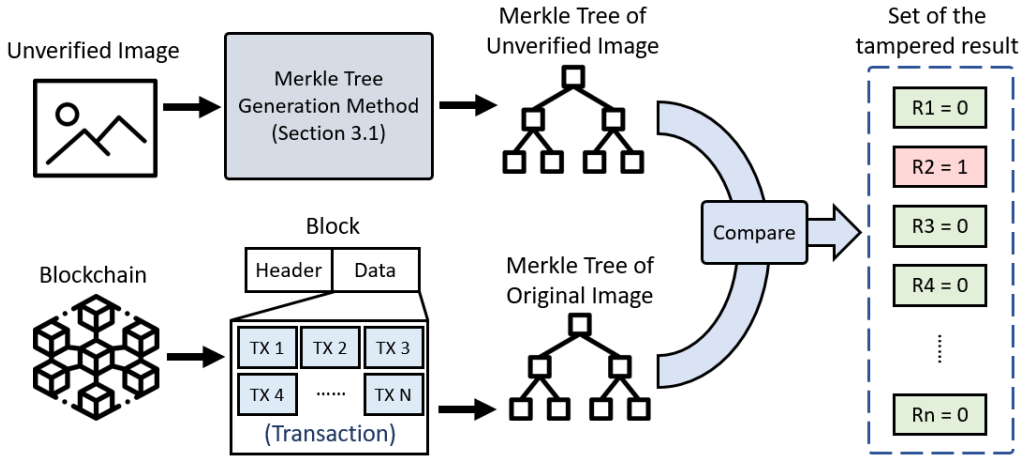


Figure 9. Process of image verification.

3.3. Image Recovery

In the case where image has been tampered with, the tampered area can be pointed out by using the Image Verification method in Section 3.2 (see Figure 10). After the tampered area has been detected, the transactions data is retrieved from the blockchain (see Figure 11). Then, the cipher image blocks are retrieved from the IPFS and the file is decrypted to patch back to the tamped area on the image. After downloading the data from IPFS, decrypt it to image blocks (refer Section 3.1). Finally, the decrypted image block is used to restore the decrypted area on the tampered area (refer Figure 12). The key steps of the proposed image recovery method are summarized as Procedure 3.

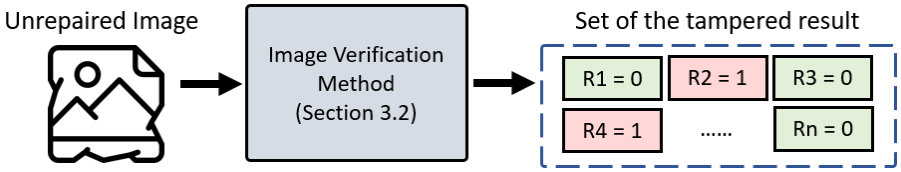


Figure 10. Process of getting the tampered result of tampered image.

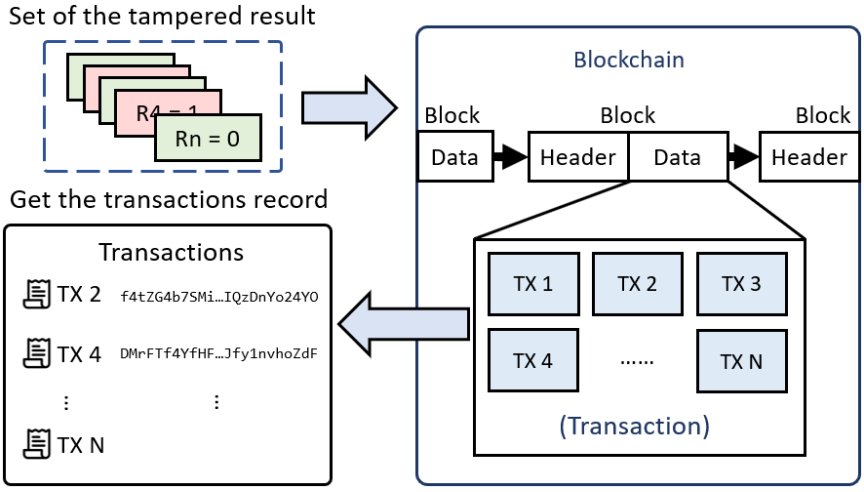


Figure 11. Process of getting the transactions record.

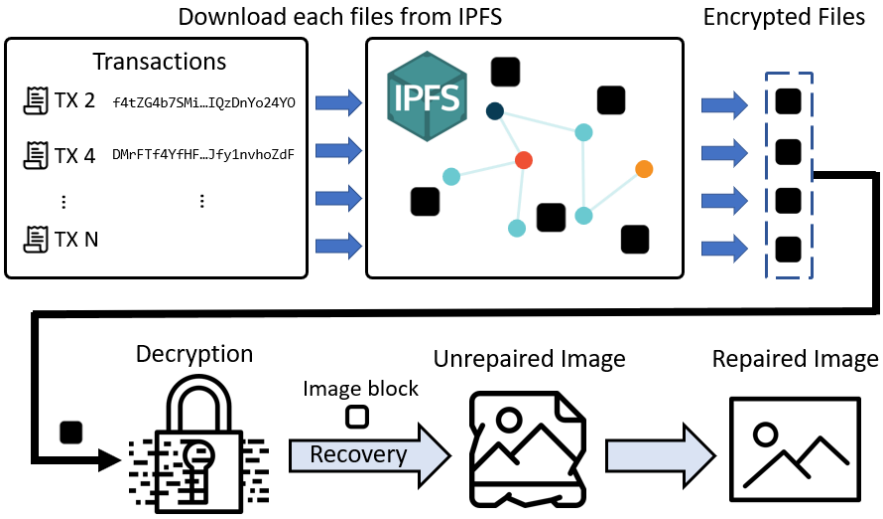


Figure 12. Process of image recovery.

Procedure 3 Image Recovery

Input: Unrecovered Image I_{unr}
Output: Repaired Image I_{unr}

1: Detect tampered area on Iunr by applying Image verification method (refer to Section 3.2) and get the result set $R_i = (r_1, r_2, ..., r_n)$,.

2: If $(r_i == 1)$, then get TX_i from the block of blockchain in which the original image stored, and go to Step 3, otherwise go to Step 5.

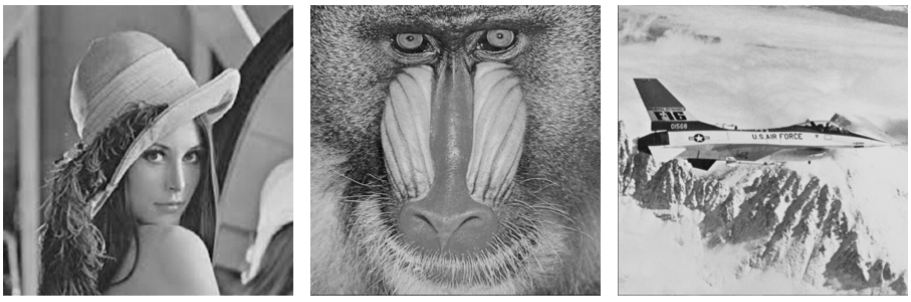
3: Download the file corresponding to TX_i from IPFS, and decrypt the file to $I_i^{'}$.

4: Restore $I_i^{'}$ back to the area that was tampered with in the I_{unr} and generate as Restored Image $I_{unr}^{'}$.

5: If all of the block has been checked then stop procedure, otherwise go to Step 2.

4. Simulation Results

In order to evaluate the performance of the proposed method, we implemented it in Ubuntu 18.04.02 OS on a quad-core 3.4 GHz CPU with 4GB RAM. Six commonly used images were used in our simulation, namely, Lena, Baboon, Jet, Boat, Pepper, and Sailboat. All of the images were 8-bit grayscale images, and the size of each image was 512 × 512 pixels, as shown in Figure 13.



(a) Lena (b) Baboon (c) Jet

(d) Boat (e) Pepper (f) Sailboat

Figure 13. Test images.

The blockchain was implemented using python 3.6.8, the program refers to the article on the blockchain structure [5,7,11,12]. As shown in Figure 14, the genesis block (also called the first block) was generated in the blockchain. The value of the previous hash has sixty-four '0'. The image block size is $N \times N$ (e.g., $N = 16$) pixels in the Merkle tree generation method (Refer to Section 3.1), and RSA encryption is applied to play the encryption mechanism in our method.

```
index : 0
timestamp : 1554197644.928186
previousHash : 0000000000000000000000000000000000000000000000000000000000000000
data : First Block
nonce : 106636
hash : 0000e2219a6e44b82a2a059f2eb922ccc7d3f3780c4c60352723ff1642aa6281
merkleroot : None
```

Figure 14. Running program of the blockchain.

In our simulation, the image can be divided into 2048 block images. The RSA key pair was generated by applying python Cryptographic module pycrypto 2.6.1. After that, encrypted files were uploaded to IPFS. Figure 15 shows a part of the fingerprint after upload $B_i^{'}$ to IPFS. Each fingerprint of $B_i^{'}$ is added to the transaction and the Merkle Root is stored into a block in the blockchain, the result is shown as Figure 16.

```
added Qmf8Cox7VAzhBWEqjbEBmZBhpFaZ8oCGSL83FUmMTmjHMp lean_1024_1.enc
added QmYURpNSicqtkWycxYbMbBqExK3RYE9VhKqFkRvtnzsQpX lean_1024_2.enc
added QmdH7e36NpGGtcn2jNGtaDhuscTPehjQHgyviV9Rdf4mrJ lean_1024_3.enc
```

Figure 15. Upload of encrypted files.

```
index : 1
timestamp : 1553929404.4188406
previousHash : 00004014719817bfef815fe22c7698adfa2b02262e13d76221fa67d59113a319
data : MSB_lena_16
nonce : 563
hash : 0000a7f09c625f864a7984a30af80b76e63046f8cbb21772262125f3d3791426
merkleroot : 44ebe9c0ba74ab36d1c4ff40b6c0ed57b7eb13a3149b7a488ee3675b1b4c63e7
----------Transactions----------
owner : tommy
imageHash : 5363CA467448DA8970B9D8CAACDC75BB9FE75757D1C8245660BC55EA44CE6A26
fingerprint : Qmf8Cox7VAzhBWEqjbEBmZBhpFaZ8oCGSL83FUmMTmjHMp
totalPieces : 16
piece : 1
owner : tommy
imageHash : 239933F7D897236B2EDEA4B96337B0DACF0273DD9CA7AD67F31A4D815CD8DC9A
fingerprint : QmYURpNSicqtkWycxYbMbBqExK3RYE9VhKqFkRvtnzsQpX
totalPieces : 16
piece : 2
owner : tommy
imageHash : 6C7DEF198B9C79867A5092033C046A4009277AA47F51FAAD1C2C74B706939964
fingerprint : QmdH7e36NpGGtcn2jNGtaDhuscTPehjQHgyviV9Rdf4mrJ
totalPieces : 16
piece : 3
owner : tommy
imageHash : 48E73FB01ACEF42468B44AEF7EA73521528174A443BCBB731E8557A36B919FD0
fingerprint : QmYWSyn7QHZfdSPtiMvrQPmry5zU61Zzz4wjFYYb1TuTeg
```

Figure 16. Stored transaction information to a block.

For testing the effectiveness of tamper area detection in our method, we tamper the test image (e.g., add a flower into Lena image), shown in Figure 17. The testing uses different slicing block sizes ($N = \{128, 64, 32, 16\}$). Figure 18 shows the result of the detected tamper area on the tampered Lena image. Figure 18a is the result when $N = 32$, while Figure 18b is the result when $N = 16$. Figure 19 shows the detection accuracy results in different slicing block size. As we see, the accuracy is improved when the slicing size is smaller.

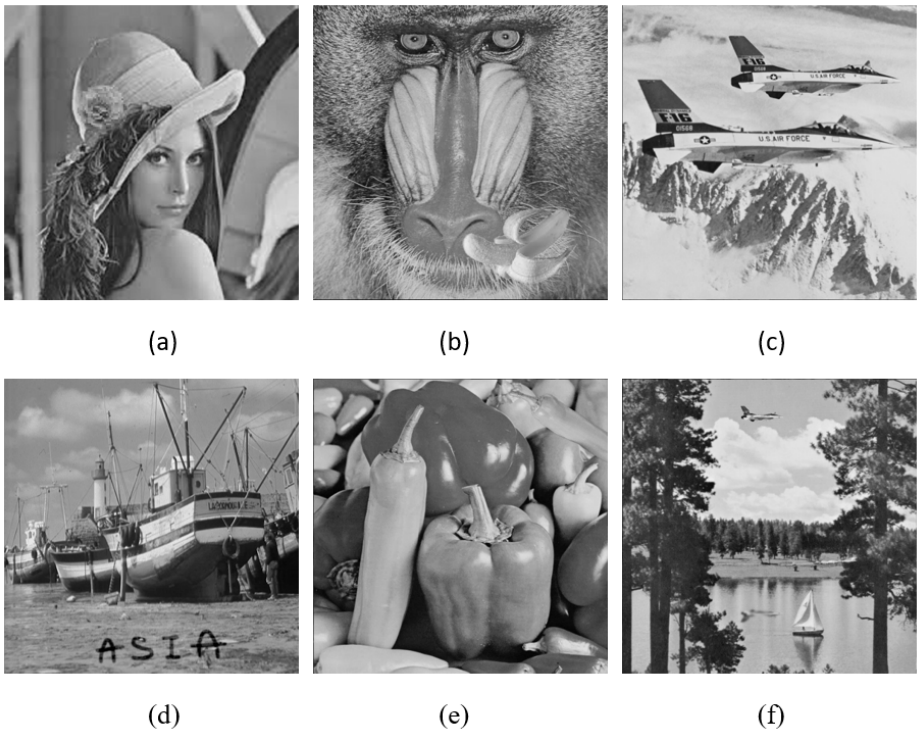


Figure 17. Tampered images. (**a**) Lena, (**b**) Baboon, (**c**) Jet, (**d**) Boat, (**e**) Pepper, (**f**), Sailboat.

```
check error >>>>>>>>>>>>>>>>  69
check error >>>>>>>>>>>>>>>>  70
check error >>>>>>>>>>>>>>>>  71
check error >>>>>>>>>>>>>>>>  84
check error >>>>>>>>>>>>>>>>  85
check error >>>>>>>>>>>>>>>>  86
check error >>>>>>>>>>>>>>>>  87
check error >>>>>>>>>>>>>>>>  101
check error >>>>>>>>>>>>>>>>  102
check error >>>>>>>>>>>>>>>>  103
check error >>>>>>>>>>>>>>>>  118
[69, 70, 71, 84, 85, 86, 87, 101, 102, 103, 118]
```

(a)

```
check error >>>>>>>>>>>>>>>>  397
check error >>>>>>>>>>>>>>>>  398
check error >>>>>>>>>>>>>>>>  425
check error >>>>>>>>>>>>>>>>  426
check error >>>>>>>>>>>>>>>>  427
check error >>>>>>>>>>>>>>>>  428
check error >>>>>>>>>>>>>>>>  429
check error >>>>>>>>>>>>>>>>  459
check error >>>>>>>>>>>>>>>>  460
[266, 267, 268, 269, 297, 298, 299, 300, 301,
 328, 329, 330, 331, 332, 333, 334, 360, 361,
 362, 363, 364, 365, 366, 393, 394, 395, 396,
 397, 398, 425, 426, 427, 428, 429, 459, 460]
```

(b)

Figure 18. The result of tampered region detection on tampered Lena image (**a**) $N = 32$ (**b**) $N = 16$.

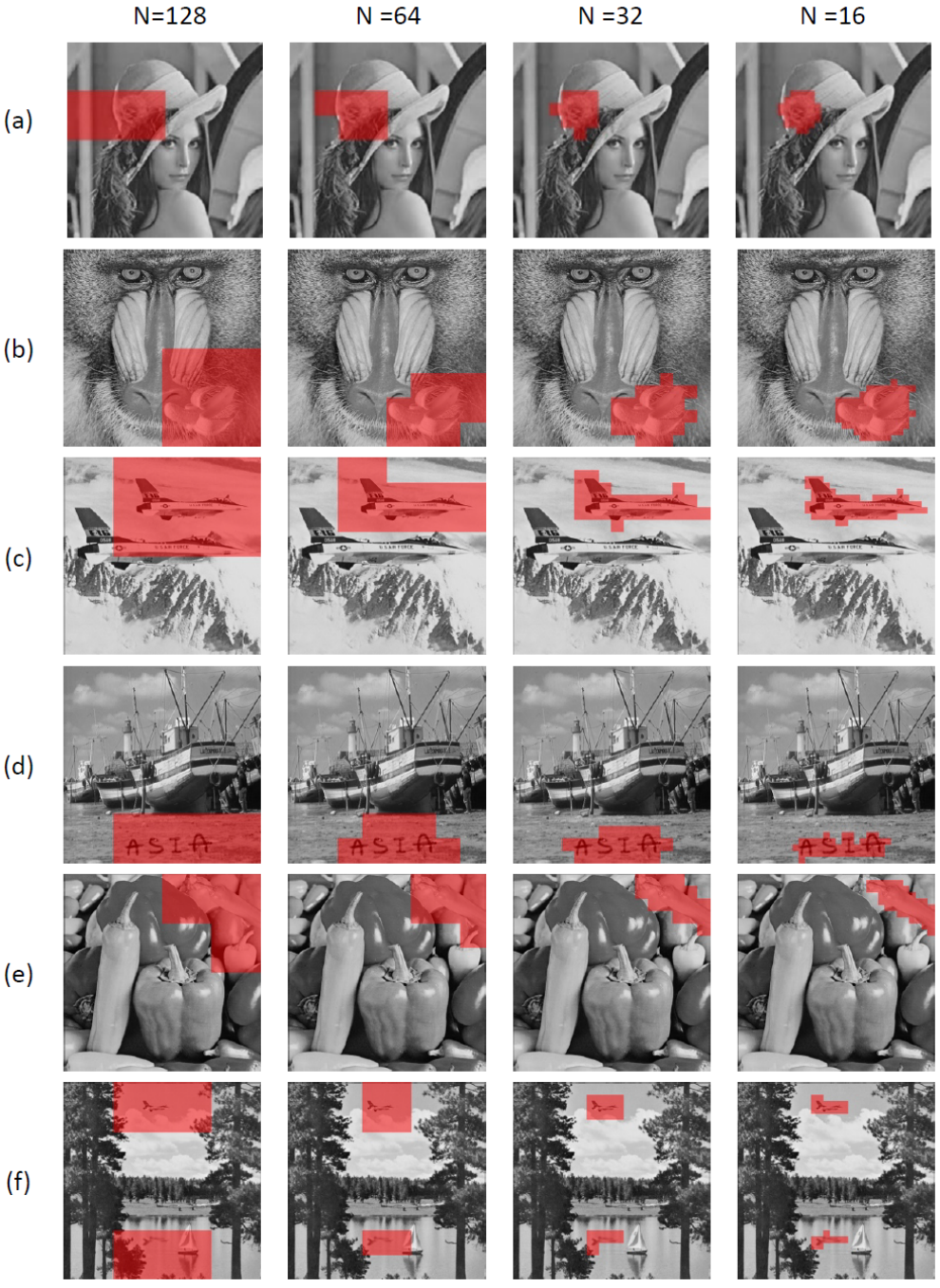


Figure 19. The result of detected tamper area with different slicing size: (**a**) Lena, (**b**) Baboon, (**c**) Jet, (**d**) Boat, (**e**) Pepper, (**f**), Sailboat.

The size of slicing block setting is highly related to fineness of tampered area. When $N = 128$, the detected area is very rough, and covers a lot of areas that have not been changed. When $N < 128$, the detected area is getting more excellent, meaning when the setting of N is smaller, we will get a better detection result. On the other hand, the size of Merkle tree is also highly correlated with the image block, which means a small N will lead to a large Merkle tree compared to large N. Figure 20 shows that when the amount of a sliced block reaches the maximum node of bottom layers of Merkle tree, the size of Merkle tree will grow in multiples. In this experiment, an N value of 32 is the most appropriate.

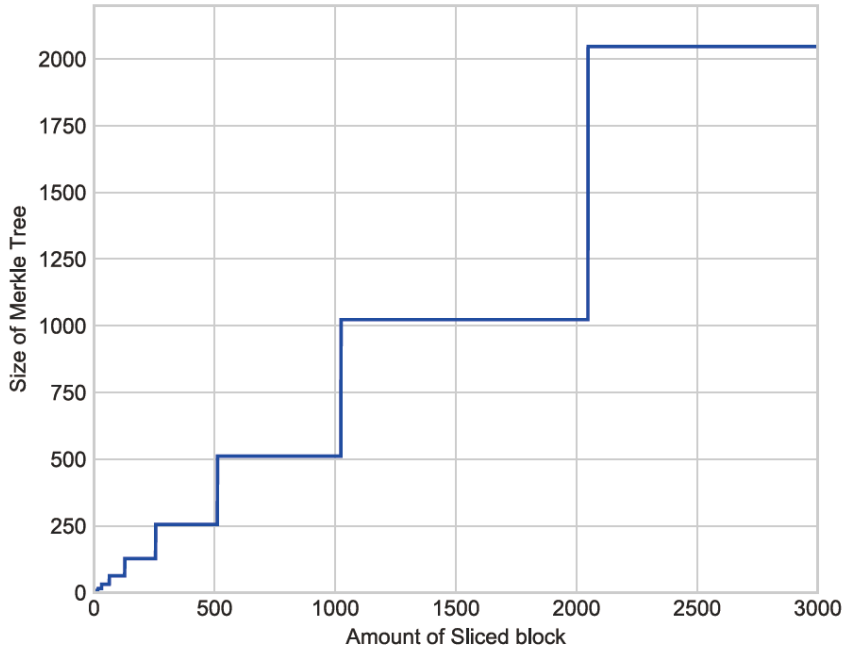


Figure 20. The size of Merkle tree in different amounts of sliced block.

Visual quality is an important factor for evaluating the performance of the image restoration method. We adopt the PSNR (peak signal-to-noise ratio) value as the visual quality measurement in our simulation. PSNR approximates human perception of reconstruction quality, as shown in Equations (1) and (2) . Given a noise-free $m \times n$ monochrome image I and its noisy approximation K, mean squared error MSE is defined as:

$$MSE = \frac{1}{m \times n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 . \quad (1)$$

The PSNR (in dB) is defined as:

$$PSNR = 10 \times log \left(\frac{255^2}{MSE} \right) . \quad (2)$$

Due to the characteristics of the hashing function, as long as there is a slight change or even a bit, the hash values will be completely different. In order to make this method more robust, we extract the first few bits of the Most Significant Bit (MSB) of the image. In this experiment, we take two approaching parameters to our proposed method, MSB and block slice size N, and made several different adjustments. The MSB values are 3, 4, 5, and 6 bits; and block slice sizes are 128, 64, 32, and 16 pixels, making up to 16 different measurements, Tables 1 and 2 shown the experiment results of the different testing image.

Refer to Figure 21 , the MSB setting is highly related to visual quality. The PSNR value is not much different when the MSB value is 3 bits and 4 bits. However, when the MSB value is 5 bits or more, the PSNR value is significant improved. That means the higher the MSB value setting, the better the PSNR value, and more complete the recovery. However, the MSB values will affect the accuracy of detecting the tampered area of the image. If the MSB value is set too high, its recognition will not be robust. In this experiment, an MSB value of 5 bits is the most appropriate.

Table 1. The mean squared error (MSE) value on different images with different measurements.

Image	bits	16 × 16	32 × 32	64 × 64	128 × 128
lena	3 bits	3.29	3.85	6.88	10.52
	4 bits	2.66	3.12	5.70	8.99
	5 bits	0.62	0.84	1.33	2.12
	6 bits	0.13	0.23	0.34	0.45
baboon	3 bits	8.30	10.37	14.84	23.63
	4 bits	6.82	8.47	12.14	19.48
	5 bits	1.57	1.94	2.77	4.41
	6 bits	0.31	0.39	0.55	0.88
airplane	3 bits	7.12	9.80	20.25	35.92
	4 bits	6.19	8.41	16.55	28.52
	5 bits	1.36	1.89	3.77	6.45
	6 bits	0.27	0.38	0.77	1.31
boat	3 bits	4.57	7.95	12.11	18.19
	4 bits	3.90	6.68	10.17	15.25
	5 bits	0.85	1.47	2.25	3.38
	6 bits	0.17	0.29	0.44	0.66
peppers	3 bits	41.62	39.83	38.03	17.00
	4 bits	42.50	40.75	38.89	14.11
	5 bits	49.05	47.29	45.33	3.27
	6 bits	55.94	54.23	52.32	0.65
sailboat	3 bits	4.48	6.75	10.22	34.23
	4 bits	3.65	5.46	8.39	35.33
	5 bits	0.81	1.21	1.90	41.70
	6 bits	0.17	0.25	0.38	48.65

Table 2. The peak signal-to-noise ratio (PSNR) value on different image with different measurements.

Image	bits	16 × 16	32 × 32	64 × 64	128 × 128
lena	3 bits	42.96	42.27	39.75	37.90
	4 bits	43.88	43.19	40.57	38.59
	5 bits	50.22	49.44	46.89	44.86
	6 bits	57.15	54.59	52.87	51.57
baboon	3 bits	38.93	37.97	36.41	34.39
	4 bits	39.79	38.85	37.28	35.23
	5 bits	46.18	45.26	43.71	41.68
	6 bits	53.15	52.24	50.70	48.69
airplane	3 bits	39.60	38.21	35.06	32.57
	4 bits	40.21	38.88	35.94	33.57
	5 bits	46.79	45.36	42.39	40.03
	6 bits	53.77	52.32	49.29	46.95
boat	3 bits	41.52	39.12	37.29	35.53
	4 bits	42.21	39.88	38.05	36.29
	5 bits	48.82	46.45	44.61	42.84
	6 bits	55.87	53.53	51.67	49.90
peppers	3 bits	41.62	39.83	38.03	35.82
	4 bits	42.50	40.75	38.89	36.63
	5 bits	49.05	47.29	45.33	42.98
	6 bits	55.94	54.23	52.32	49.98
sailboat	3 bits	44.23	41.96	38.14	34.23
	4 bits	45.59	43.20	39.46	35.33
	5 bits	52.20	49.80	45.91	41.70
	6 bits	59.11	56.68	52.92	48.65

The false positive rate of the tampering area evaluations is not suitable for the proposed method—because of the features of IPFS, the same image file has only one corresponding fingerprint, even when adjusting the brightness or contrast of the image.

Our proposed method can not only work on grayscale images but also on any type and any size of digital image. See Figure 22a, where the testing image size is 768 × 512 pixels 24-bit color images, and the tampered image is shown in Figure 22b. Figure 22c shows the result of using our proposed method on an RGB image.

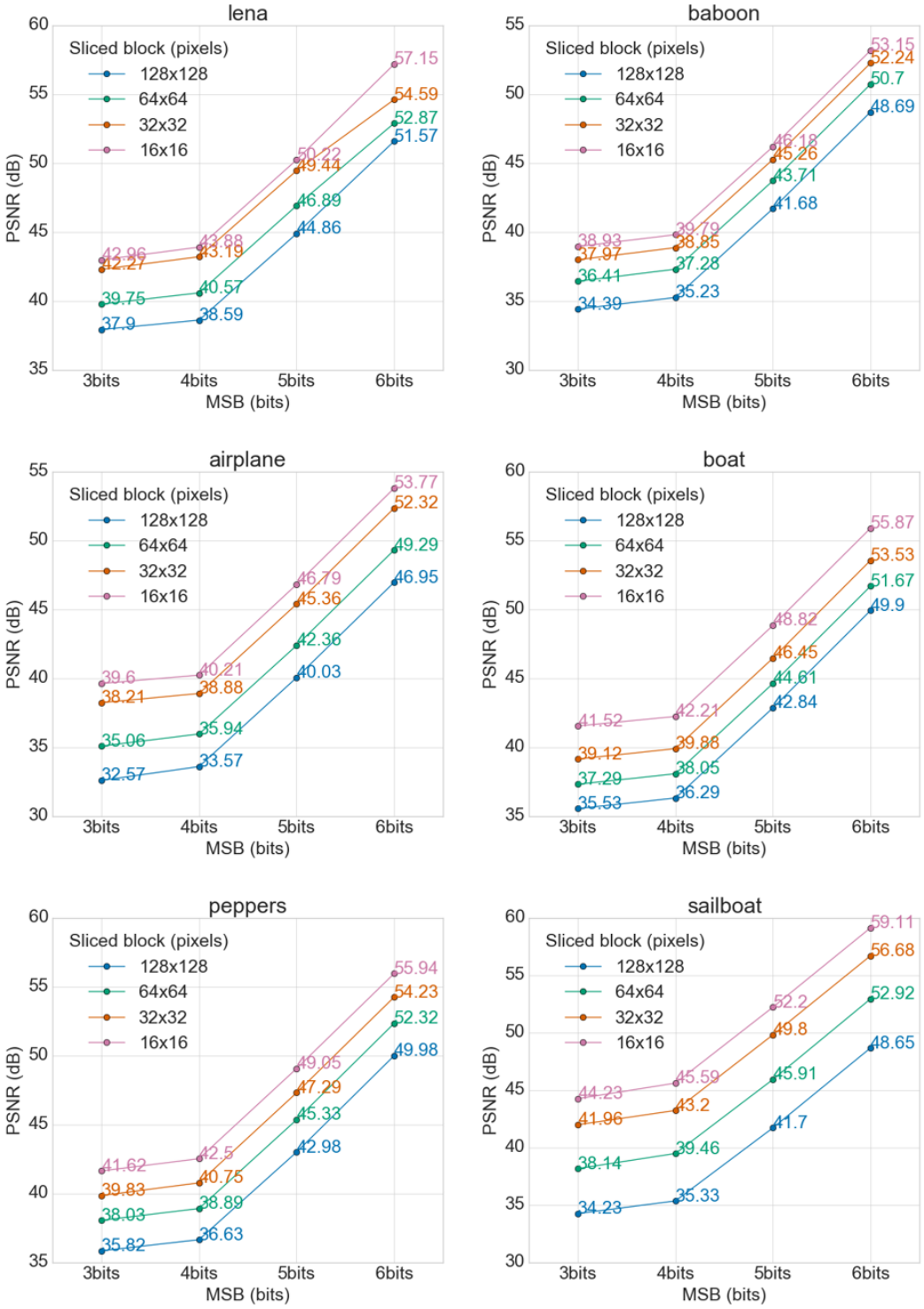


Figure 21. PSNR value comparison.

Figure 22. The simulation result of testing on an RGB image. (**a**) Test image. (**b**) Tampered image. (**c**) Tampered area detection result.

5. Conclusions

The slice block size N depends on each case, the smaller the size, the more accurate the detection, but the larger the generating of the Merkle tree will be, and vice versa. In this paper, we integrate Merkle tree technology, IPFS decentralized storage system, and an image verification process to achieve the goal of image verification. The proposed method shows the ability to perform integrity verification and recovery of images. From the simulation results, we have achieved a verification and restoration process of the image tampering area. We suggest that $N = 32$ and $MSB = 5$ is suitable in our proposed method. In terms of file types, we only mention the digital image in this paper. For future works, this may not only be applied to the digital image file but also expanded to video or any other multimedia (e.g., a decentralized document verification system). Through this system, governments or organizations can protect their documents, avoid tampering or falsifying messages, and ensure the credibility of documents. In addition to blockchain framework research, it is also worth studying for blockchain smart contracts.

Author Contributions: Conceptualization, Y.-C. Chen and Y.-C. Chou; methodology, Y.-C. Chen; validation, Y.-C. Chen and Y.-P. Chou; formal analysis, Y.-C. Chen and Y.-P. Chou; investigation, Y.-C. Chen and Y.-P. Chou; resources, Y.-C. Chou; data curation, Y.-C. Chen, Y.-P. Chou, and Y.-C. Chou; writing—original draft preparation, Y.-C. Chen; writing—review and editing, Y.-C. Chen and Y.-C. Chou; visualization, Y.-C. Chen and Y.-P. Chou.

Funding: This research received no external funding.

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

References

1. Hwang, M.S.; Lin, I.C. *Introduction to Information and Network Security (in Chinese)*; Mcgraw Hill: Taipei, Taiwan, 2011.
2. Kumar, C.; Singh, A.K.; Kumar, P. A recent survey on image watermarking techniques and its application in e-governance. *Multimed. Tools Appl.* **2018**, *77*, 3597–3622. [CrossRef]
3. Cox, I.J.; Miller, M.L.; Bloom, J.A. Watermarking applications and their properties. In Proceedings of the International Conference on Information Technology: Coding and Computing (Cat. No. PR00540), Las Vegas, NV, USA, 27–29 March 2000; pp. 6–10.
4. Joshi, M.A. *Digital Image Processing: An Algorithmic Approach*; PHI Learning Pvt. Ltd.: Delhi, India, 2018.
5. Lin, I.C.; Liao, T.C. A Survey of Blockchain Security Issues and Challenges. *IJ Netw. Secur.* **2017**, *19*, 653–659.
6. Swan, M. *Blockchain: Blueprint for a New Economy*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2015.
7. Hughes, L.; Dwivedi, Y.K.; Misra, S.K.; Rana, N.P.; Raghavan, V.; Akella, V. Blockchain research, practice and policy: Applications, benefits, limitations, emerging research themes and research agenda. *Int. J. Inf. Manag.* **2019**, *49*, 114–129. [CrossRef]
8. Benet, J. Ipfs-content addressed, versioned, p2p file system. *arXiv* **2014**, arXiv:1407.3561.
9. Mougayar, W. *The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology*; John Wiley & Sons: Hoboken, NJ, USA, 2016.
10. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [CrossRef]

11. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: http://bitcoin.org/bitcoin.pdf (accessed on 5th July 2019).
12. Sultan, K.; Ruhi, U.; Lakhani, R. Conceptualizing Blockchains: Characteristics & Applications. In Proceedings of the 11th IADIS International Conference Information Systems, Lisbon, Portugal, 14–16 April 2018; pp. 49–57.
13. Carter, J.L.; Wegman, M.N. Universal classes of hash functions. *J. Comput. Syst. Sci.* **1979**, *18*, 143–154. [CrossRef]
14. Merkle, R. Secrecy, Authentication, and Public Key Systems. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 1979.
15. Becker, G. Merkle signature schemes, merkle trees and their cryptanalysis. Ruhr-University Bochum, Tech. Rep. 2008. Available online: https://www.emsec.ruhr-uni-bochum.de/media/crypto/attachments/files/2011/04/becker_1.pdf (accessed on 5th July 2019).
16. Merkle, R.C. Protocols for public key cryptosystems. In Proceedings of the 1980 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 14–16 April 1980; p. 122.
17. Haouzia, A.; Noumeir, R. Methods for image authentication: A survey. *Multimed. Tools Appl.* **2008**, *39*, 1–46. [CrossRef]
18. Chang, C.C.; Chen, T.S.; Huang, K.F. *Digital Image Processing Techniques*; Flag Information Co. Ltd.: Taipei, Taiwan, 2003. (In Chinese)
19. Bhowmik, D.; Feng, T. The multimedia blockchain: A distributed and tamper-proof media transaction framework. In Proceedings of the 2017 22nd International Conference on Digital Signal Processing (DSP), London, UK, 23–25 August 2017; pp. 1–5.
20. Bhowmik, D.; Natu, A.; Ishikawa, T.; Feng, T.; Abhayaratne, C. The Jpeg-Blockchain Framework For Glam Services. In Proceedings of the 2018 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), San Diego, CA, USA, 23–27 July 2018; pp. 1–6.
21. Badr, B.; Horrocks, R.; Wu, X.B. *Blockchain By Example: A Developer'S Guide to Creating Decentralized Applications Using Bitcoin, Ethereum, and Hyperledger*; Packt Publishing Ltd.: Birmingham, UK, 2018.