

Article

Compression-Unimpaired Joint Encryption and Authentication for PIZ-Compressed OpenEXR HDR Images

Jing-Ya Yan and Tzung-Her Chen * 

Department of Computer Science and Information Engineering, National Chiayi University, Chiayi City 600, Taiwan; s1020422@mail.ncyu.edu.tw

* Correspondence: thchen@mail.ncyu.edu.tw

Abstract

High dynamic range (HDR) images require protection of both confidentiality and integrity, but conventional LDR-oriented security methods are difficult to apply directly because HDR images differ substantially in file structure, numerical representation, and compression mechanisms. This paper focuses on secure protection of PIZ-compressed OpenEXR HDR images, where preserving compression syntax and decoder compatibility is essential. In our previous work, we proposed a commutative joint encryption and authentication scheme for uncompressed OpenEXR HDR images (This paper is an extended version of our previously published work in *Multimedia Tools and Applications*, entitled “Commutative Encryption and Authentication for OpenEXR High Dynamic Range Images” by Chen and Yan. Compared with the previous work, this extended version specifically addresses the challenge of integrating PIZ compression encoding into the joint encryption and authentication process for OpenEXR HDR images, which was not considered in our earlier study). However, that scheme did not address OpenEXR images encoded with PIZ compression. To overcome this limitation, this paper presents a compression-unimpaired joint encryption and authentication scheme that operates directly on Huffman-coded values in the chunk data of PIZ-compressed OpenEXR images. Specifically, we partition the histogram of values output according to the corresponding Huffman code into two segments: an encrypted part for confidentiality and an embedded part for authentication. To the best of our knowledge, this study is the first attempt to investigate commutative encryption and authentication directly in the PIZ-compressed OpenEXR HDR image domain. Experimental results on seven OpenEXR HDR images show that the decrypted and authenticated images preserve high visual quality, with PSNR values exceeding 75 dB in most cases. The protected files remain decodable by OpenEXR-compatible software, and the file size is preserved because the proposed operations replace valid Huffman codes with codes of identical length. These results indicate that the proposed scheme can jointly support confidentiality, authentication, format compliance, and compression preservation.



Academic Editor: Cheonshik Kim

Received: 7 June 2026

Revised: 2 July 2026

Accepted: 7 July 2026

Published: 9 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: high dynamic range image; encryption; authentication; compression; OpenEXR; PIZ

1. Introduction

High Dynamic Range (HDR) imaging has become increasingly important in professional visual computing workflows because it preserves luminance and color information that cannot be adequately represented by conventional Low Dynamic Range (LDR) formats. Recent advancements in this field encompass various distinct domains, including deep learning-based single-image HDR reconstruction [1,2], quality assessment metrics

for tone mapping [3], advanced fusion techniques using neuromorphic hardware [4], and specialized frameworks for HDR image compression [5]. As HDR images are increasingly transmitted, archived, and processed in compressed formats, security mechanisms must protect image content without damaging file syntax or compression structures. HDR images have demonstrated vast application potential across diverse domains such as cinematic production, digital heritage preservation, astronomical imaging, holography, and medical diagnostics. In contrast to conventional LDR images, HDR images offer a broader range of pixel values, enabling more realistic representations of real-world scenes. However, ensuring the security of these images is critical, particularly with respect to confidentiality and integrity, which can be achieved through image encryption and authentication techniques, respectively. Unfortunately, conventional methods designed for LDR image encryption and authentication cannot be directly applied to HDR images. This limitation arises from the substantial differences in data formats and memory requirements between HDR and LDR images. For instance, HDR images are commonly represented using higher-precision numerical formats than conventional 8-bit LDR images. In the OpenEXR format considered in this study, each color component is typically represented using a 16-bit half floating-point value, which provides higher precision and dynamic range while requiring more storage than 8-bit LDR representations.

Over the past decade, research on securing HDR images has been categorized into four main areas: data hiding [6–11], robust image watermarking [12,13], image authentication [14], and image encryption [15,16]. Table 1, originally presented in [17], summarizes the current state of security research related to the Radiance RGBE [18], LogLuv TIFF [19], and OpenEXR [20] file formats.

Table 1. Comparisons of the related works on securing HDR images.

	Radiance RGBE	LogLuv TIFF	OpenEXR
Data hiding	Cheng & Wang [6], Yu et al. [7], Chang et al. [8],	Li et al. [9]	Lin et al. [10], Lan et al. [11]
Watermarking	Autrusseau & Goudia [12],	Luo et al. [13]	
Authentication	Tsai et al. [14]		
Encryption		Lin et al. [15]	Chen & Chang [16]
Authentication & Encryption			Chen & Yan [17]

Since its introduction in 2000 by Industrial Light & Magic (ILM), OpenEXR has emerged as the industry-standard format for HDR imaging, widely adopted across visual effects, animation, and digital content creation workflows. Its popularity has been further bolstered by ILM's release of open-source libraries, facilitating broad integration into commercial and professional software, including Adobe Photoshop, Autodesk 3ds Max, and high-end scanner systems such as SilverFast.

Unlike legacy HDR formats such as Radiance RGBE and LogLuv TIFF—which offer limited or no native support for compression and lack full compatibility with modern production pipelines—OpenEXR distinguishes itself by incorporating robust, flexible, and format-compliant compression techniques that make it ideal for large-scale HDR image storage and transmission without sacrificing visual fidelity or structural integrity. Specifically, OpenEXR supports multiple compression methods, including ZIP (both per-scanline and per-block) and PIZ. While ZIP compression, based on traditional deflate

algorithms, performs well on smooth images with low entropy, it is often inadequate for content featuring high-frequency details. PIZ, on the other hand, is a wavelet-based lossless compression algorithm tailored for HDR imagery. It applies a reversible wavelet transform followed by Huffman encoding, making it particularly effective for compressing grainy or noisy images, such as those with film grain, fine textures, or natural noise—scenarios where ZIP or RLE tend to underperform.

One of PIZ's major strengths is its robustness and predictability in compression performance, regardless of image complexity. This makes it exceptionally well-suited for production environments where maintaining the integrity and consistency of image data is essential, such as in visual effects compositing, animation rendering, and digital mastering. As a result, PIZ compression has become the preferred choice for preserving the full dynamic range and detail of HDR images in professional workflows, ensuring high visual precision while meeting practical requirements for efficient data handling.

In particular, OpenEXR is purpose-built to faithfully represent extreme luminance variations—from deep shadows to intense highlights—while preserving image quality during both processing and transmission. These combined capabilities—advanced compression support, industry-wide toolchain compatibility, and superior handling of high-frequency image content—have solidified OpenEXR's position as the dominant HDR image format in both academic research and commercial practice.

Given OpenEXR's dominance in HDR imaging and its widespread deployment in compression-dependent production environments, ensuring the security of OpenEXR images—without disrupting their compression structure—has become a critical research imperative. Traditional approaches to image security often neglect compression constraints, leading to solutions that either disrupt file decodability or result in format corruption. To specifically address these challenges, Chen and Yan [17] previously proposed a novel joint encryption and authentication scheme explicitly tailored for the OpenEXR file format. Their method capitalized on the commutative property between encryption and authentication, enabling authentication data to be embedded into already encrypted images—or vice versa—without interfering with the final decryption result. This ensured that the protected OpenEXR image remained decodable by standard software (such as OpenEXR-compatible CODECs), which would merely interpret it as noise-like content rather than a corrupted file.

Building on our previous work [17], which first introduced a commutative joint encryption and authentication scheme for uncompressed OpenEXR HDR images, this paper presents a significantly extended and enhanced approach that explicitly incorporates compression encoding. Specifically, we propose a novel compression-preserving joint encryption and authentication scheme tailored for OpenEXR images encoded with PIZ compression. The proposed method maintains the integrity of the original compression structure by operating directly on the Huffman-coded chunk data within the file layout. The approach involves partitioning the histogram of Huffman-coded values into two distinct segments: one designated for encryption to ensure data confidentiality and the other for embedding authentication information.

To the best of our knowledge, this work constitutes the first study to explore commutative encryption and authentication mechanisms in the context of compressed HDR imagery. By ensuring compatibility with the PIZ compression algorithm—known for its lossless wavelet-based design optimized for high-frequency image content—our scheme offers a secure, compression-aware, and format-compliant solution, preserving both the compression ratio and the visual fidelity of HDR images.

This tailor-made scheme offers several notable advantages specifically aligned with the requirements of compressed HDR imaging. It supports confidentiality against unauthorized access and provides an authentication mechanism for detecting modifications

that affect the embedded authentication bits. Crucially, the design is fully compliant with the OpenEXR file format and PIZ compression, allowing protected files to be decoded without triggering format rejection or compression errors, even though the output may appear noise-like. The proposed method introduces only imperceptible distortion into the authenticated image, thereby preserving high visual quality. Moreover, the scheme operates directly in the compressed domain—on Huffman-coded values—making it inherently compression-preserving and minimizing bitstream alteration.

The discussion of tamper localization is limited to the theoretical behavior of the proposed embedding and verification process, because the present study focuses on compression-preserving design and theoretical authentication analysis rather than quantitative tamper-localization benchmarking. Importantly, it retains the commutative property between encryption and authentication, thereby offering greater flexibility and practicality in pipeline workflows where encryption and verification may occur in variable order. Finally, the scheme preserves the compressed bitstream size by replacing valid Huffman codes with codes of identical length, thereby avoiding additional transmission overhead.

The remainder of this paper is organized as follows. Section 2 presents preliminaries on HDR image formats and OpenEXR PIZ compression encoding. Section 3 describes the proposed compression-unimpaired encryption and authentication scheme. Section 4 reports the experimental results. Section 5 discusses security analysis and limitations. Section 6 concludes the paper and outlines future research directions.

2. Preliminaries

2.1. HDR Image Format

In terms of file layout formats, the most straightforward option is the Radiance RGBE format, which includes a header that records the magic number and the RLE tag to indicate whether the image is compressed with RLE. The LogLuv TIFF format, based on the TIFF standard, defines several tags. In comparison, the OpenEXR file format offers greater flexibility because users can create arbitrary tags, and the format supports tiled images.

The half floating-point number used to represent one of the three color components is compatible with IEEE 754 and is constructed with one sign bit, five exponent bits, and 10 mantissa bits.

The 16-bit sequence can be converted into a floating-point number by Equation (1).

$$F = \begin{cases} 0, & \text{if } E = 0 \text{ and } M = 0 \\ (-1)^S \times 2^{E-15} \times \left(1 + \frac{M}{1024}\right), & \text{if } 1 \leq E \leq 30 \\ (-1)^S \times 2^{-14} \times \left(\frac{M}{1024}\right), & \text{if } E = 0 \text{ and } M > 0 \\ \pm\infty, & \text{if } E = 31 \text{ and } M = 0 \\ \text{NaNs}, & \text{if } E = 31 \text{ and } M > 0 \end{cases} \quad (1)$$

where the floating-point number F is expressed as a combination of three components: the sign bit S , exponent bits E , and mantissa bits M , each represented in decimal form.

As an illustration, consider a 16-bit sequence, 0000101100101100, which can be transformed into a floating-point representation of 0.000218868. This process involves three steps. First, the 16-bit sequence is partitioned into three sub-sequences with lengths of 1, 5, and 10 bits, respectively. Next, each sub-sequence is converted from binary to decimal notation. Finally, the resulting decimal values are combined to obtain the desired floating-point representation. The specific details of this conversion procedure are as follows:

$$S = 0$$

$$E = 00010$$

$$\begin{aligned}
 M &= 1100101100 \\
 F &= (-1)^S \times 2^{E-15} \times \left(1 + \frac{M}{1024}\right) \\
 &= (-1)^0 \times 2^{2-15} \times \left(1 + \frac{812}{1024}\right) \\
 &= 1 \times 0.00012207 \times 1.79297 \\
 &= 0.000218868
 \end{aligned}$$

Table 2 illustrates three prevalent HDR formats and provides a comparison of their fundamental attributes, as reported by [21]. Based on the findings in Table 2, users can select the HDR format that best aligns with the requirements of their application. For instance, if a user requires a more extensive dynamic range, the Radiance RGBE format may be preferable; if a smaller file size is needed, the LogLuv24 format could be more appropriate; and if a lower relative error is desired, the OpenEXR format may be the optimal choice.

Table 2. Comparison of three common HDR image formats.

Format	Bits/Pixel	Dynamic Range ($\log_{10} \frac{cd}{m^2}$)	Relative Error (%)	Compression Encoding
Radiance RGBE	32	76 orders	1.0	Run length encoding (RLE)
LogLuv24	24	4.8 orders	1.1	None
LogLuv32	32	38 orders	0.3	RLE
OpenEXR	48	10.7 orders	0.1	PIZ, ZIP, ZIPS, RLE, etc.

The OpenEXR image format is widely adopted for HDR image storage due to its versatile features. It offers support for various lossy and lossless compression encoding methods, multiple data types such as 32-bit unsigned integer, 32-bit single floating-point number, and 16-bit half floating-point number, and it provides free open-source libraries implemented in C++ by Industrial Light & Magic (ILM) with a free compiler environment. Additionally, the file structure of OpenEXR is highly flexible, allowing users to assemble any number of color channels with different data types into a single image. This is achieved by utilizing either pre-defined attributes or custom user-defined attributes, as illustrated in the Header of Figure 1. Lastly, OpenEXR also supports tiled images, which facilitate the random access of rectangular sub-regions of an image.

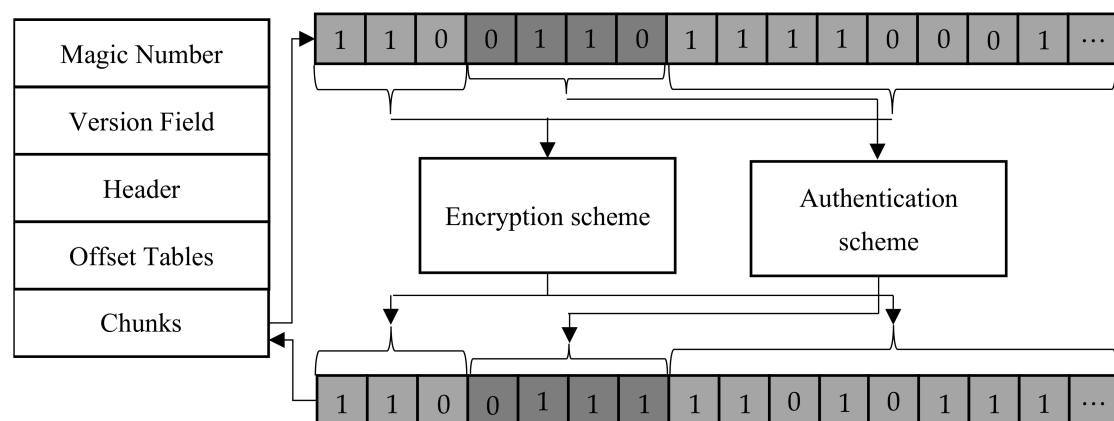


Figure 1. Schematic representation of file structure and encrypted and embedded segments of the bitstream in a graphical format.

For instance, consider a 16-bit sequence, 0000111100101100, which can be transformed into a floating-point value of 0.000437737. The specific steps involved in converting the 16-bit sequence into the corresponding floating-point number are as follows:

$$0000111100101100 \rightarrow 0,00011,1100101100$$

$$S = 0$$

$$E = 00011$$

$$M = 1100101100$$

$$\begin{aligned} F &= (-1)^S \times 2^{E-15} \times \left(1 + \frac{M}{1024}\right) \\ &= (-1)^0 \times 2^{3-15} \times \left(1 + \frac{812}{1024}\right) \\ &= 1 \times 0.000244141 \times 1.79297 \\ &= 0.000437737 \end{aligned}$$

Based on the above-mentioned survey, the following observations can be made:

- (1) The current literature still lacks a compression-aware joint encryption and authentication scheme for HDR image formats, particularly for OpenEXR images using PIZ compression encoding.
- (2) Research efforts focusing on LogLuv TIFF and OpenEXR formats are limited.
- (3) Existing schemes are not suitable for compression encoding in any HDR format. When encryption is performed prior to compression, the correlation between adjacent pixels is destroyed, leading to a reduction in the compression rate. Therefore, the consideration of compression encoding is crucial for an effective encryption scheme.

2.2. The OpenEXR Image Format with PIZ Compression Encoding

Typically, images are compressed prior to transmission over the Internet to reduce bandwidth usage, time delay, and other transmission expenses. The OpenEXR image format offers several compression methods, including both lossless and lossy compression methods. However, different compression methods have varying tradeoffs between speed and compression ratio. While the lossless compression method can retain the image data before and after compression, the lossy compression method only approximates the image data, resulting in pixel values that may be slightly altered from the original. This paper focuses on PIZ, a lossless compression method supported by OpenEXR and commonly used in HDR image workflows. PIZ combines reversible wavelet transformation and Huffman encoding, making it suitable for studying compression-domain encryption and authentication. The reported compression performance of PIZ may vary according to image content, texture complexity, and implementation settings.

The PIZ compression encoding technique is integrated with a sequence of steps (refer to Figure 2), all of which involve reversible computations, ensuring lossless compression. For clarity of explanation, a 4×4 image block is used as a simplified example to illustrate the reversible steps of PIZ compression. In actual OpenEXR PIZ processing, the compressed data are organized in chunks, and the Huffman table and encoded data are handled according to the chunk structure defined by the OpenEXR file layout. Therefore, the 4×4 block example should be regarded as an illustrative case rather than the actual processing unit of full-resolution images. The image block is assembled using the pixel values and three color channels, each comprising 16-bit half floating-point numbers, as tabulated in Table 3. The left column of Table 3 specifies the three color channels, whereas the right column lists the pixel values. There are 16 pixel values in each color channel, thereby totaling $4 \times 4 = 16$.

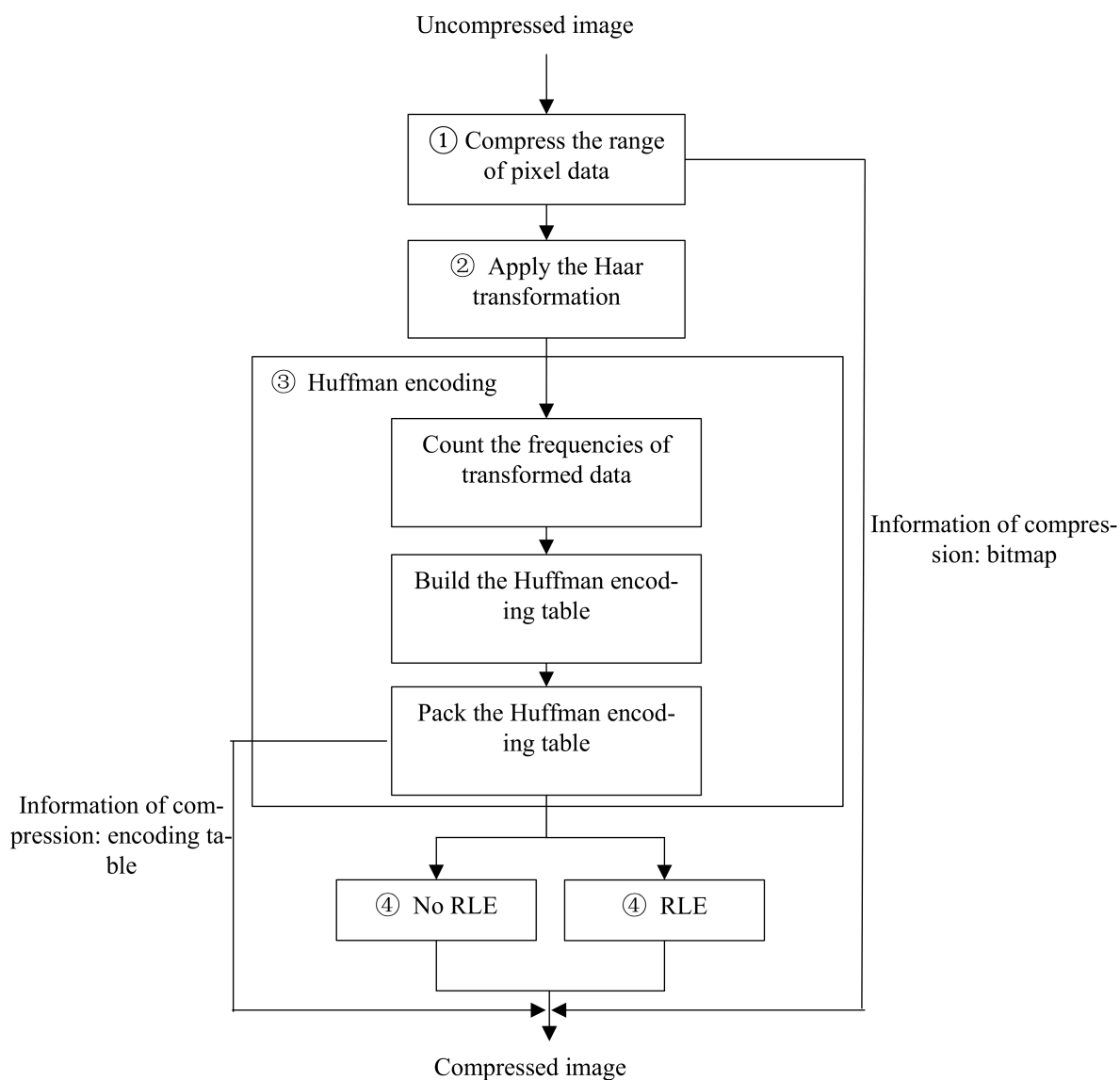


Figure 2. The flowchart of PIZ compression encoding method.

Table 3. The pixel values of each color channel in the 4×4 image block.

Color Channel	Pixel Values of the 4×4 Image Block			
R	0.0489807	0.0569153	0.0595398	0.0606384
	0.0484314	0.0578918	0.0644531	0.0640259
	0.0497131	0.0575867	0.065979	0.06427
	0.0505676	0.0577698	0.0666504	0.0656128
G	0.0274658	0.0293884	0.0263062	0.0270996
	0.0264282	0.029541	0.0281982	0.0292358
	0.0259705	0.0281067	0.0282288	0.0280762
	0.0261841	0.0291748	0.0292053	0.0277405
B	0.00806427	0.00986481	0.00855255	0.00815582
	0.00830841	0.00988007	0.00865936	0.00830841
	0.00782013	0.00875092	0.00875092	0.00850677
	0.00758743	0.00817108	0.00827789	0.00855255

Step 1: In this stage, a bitmap is utilized to establish the mapping relationship between the original pixel values and the compressed pixel values. In an uncompressed image, a 16-bit half floating-point number is represented by a 16-bit bitstream. However, during the compression process, the bitstream is calculated as unsigned integers, resulting in a value range of 0 to 65,535. Typically, not all values within this range are utilized in a single image, creating gaps between the values. To bridge this gap, the values are mapped to consecutive unsigned integers. For instance, the values shown in Table 3 are mapped to other values (as depicted in Table 4). The right column of Table 4 represents the compressed values, whereas the left column lists the three color channels. With 45 distinct values in total, the compressed values range from 1 to 45. The bitmap provides additional compression-related information and forms an integral component of the compressed image.

Table 4. The compressed values of each color channel in the 4×4 image block.

Color Channel	Compressed Values of the 4×4 Image Block			
R	31	34	38	39
	30	37	42	40
	32	35	44	41
	33	36	45	43
G	19	28	16	18
	17	29	23	27
	14	22	24	21
	15	25	26	20
B	3	12	9	4
	7	13	10	7
	2	11	11	8
	1	5	6	9

Step 2: In this stage, we employ the two-dimensional Haar transform in both horizontal and vertical directions to map the compressed data onto the wavelet domain. The Haar transform maps two input values, namely ‘ a ’ and ‘ b ’, into two corresponding wavelet values ‘ l ’ and ‘ h ’, as depicted in Equations (2) and (3) for the inverse transformation. As an example, the values presented in Table 4 are transformed into wavelet coefficients, as shown in Table 5. The left and right columns of Table 5 correspond to the three color channels and the respective wavelet coefficients, respectively. In the case of negative values, their complements are used to represent them. For instance, -1 is represented by 65,535 as its complement. By comparing Tables 4 and 5, it can be observed that the number of values is reduced from 45 to 17, which is beneficial for image compression. Table 6 provides a detailed account of the wavelet transformation for the values in the red color channel. Since the image size is 4×4 , the level of the Haar transform is set to 2.

$$\begin{cases} l = \left\lfloor \frac{a+b}{2} \right\rfloor \\ h = a - b \end{cases} \quad (2)$$

$$\begin{cases} a = l + \left\lceil \frac{h}{2} \right\rceil \\ b = a - l \end{cases} \quad (3)$$

Table 5. The wavelet values of each color channel in the 4×4 image block.

Color Channel	Transformed Values of the 4×4 Image Block			
R	36	65,531	65,527	0
	65,535	4	65,533	65,533
	65,533	65,533	3	2
	65,535	0	65,534	1
G	21	65,525	65,535	65,533
	0	3	65,528	2
	2	65,527	5	4
	65,534	2	65,535	65,533
B	6	65,528	65,534	4
	65,533	65,533	65,534	2
	1	65,529	5	0
	3	65,531	2	6

Table 6. Details of the transformations with the values in the red color channel.

Transformation Descriptions	Values of the 4×4 Image Block			
Original compressed values	31	34	38	39
	30	37	42	40
	32	35	44	41
	33	36	45	43
Level 1 of horizontal direction	$\left\lfloor \frac{31+34}{2} \right\rfloor = 32$	$31 - 34 = -3$	$\left\lfloor \frac{38+39}{2} \right\rfloor = 38$	$38 - 39 = -1$
	$\left\lfloor \frac{30+37}{2} \right\rfloor = 33$	$30 - 37 = -7$	$\left\lfloor \frac{42+40}{2} \right\rfloor = 41$	$42 - 40 = 2$
	$\left\lfloor \frac{32+35}{2} \right\rfloor = 33$	$32 - 35 = -3$	$\left\lfloor \frac{44+41}{2} \right\rfloor = 42$	$44 - 41 = 3$
	$\left\lfloor \frac{33+36}{2} \right\rfloor = 34$	$33 - 36 = -3$	$\left\lfloor \frac{45+43}{2} \right\rfloor = 44$	$45 - 43 = 2$
Level 1 of vertical direction	$\left\lfloor \frac{32+33}{2} \right\rfloor = 32$	$\left\lfloor \frac{-3+-7}{2} \right\rfloor = -5$	$\left\lfloor \frac{38+41}{2} \right\rfloor = 39$	$\left\lfloor \frac{-1+2}{2} \right\rfloor = 0$
	$32 - 33 = -1$	$-3 - -7 = 4$	$38 - 41 = -3$	$-1 - 2 = -3$
	$\left\lfloor \frac{33+34}{2} \right\rfloor = 33$	$\left\lfloor \frac{-3+-3}{2} \right\rfloor = -3$	$\left\lfloor \frac{42+44}{2} \right\rfloor = 43$	$\left\lfloor \frac{3+2}{2} \right\rfloor = 2$
	$33 - 34 = -1$	$-3 - -3 = 0$	$42 - 44 = -2$	$3 - 2 = 1$
Level 2 of horizontal direction	$\left\lfloor \frac{32+39}{2} \right\rfloor = 35$	-5	$32 - 39 = -7$	0
	-1	4	-3	-3
	$\left\lfloor \frac{33+43}{2} \right\rfloor = 38$	-3	$33 - 43 = -10$	2
	-1	0	-2	1
Level 2 of vertical direction	$\left\lfloor \frac{35+38}{2} \right\rfloor = 36$	-5	$\left\lfloor \frac{-7+-10}{2} \right\rfloor = -9$	0
	-1	4	-3	-3
	$35 - 38 = -3$	-3	$-7 - -10 = 3$	2
	-1	0	-2	1

Step 3: In this stage, we utilize Huffman encoding to produce the Huffman table. Initially, we tally the frequency of wavelet values, as illustrated in Table 7. Subsequently, we generate the Huffman table based on the frequency distribution, including an additional value of 65,536 (refer to Table 8). There are 18 unique values, each assigned a corresponding Huffman code. The Huffman code assigned to the supplementary value is adopted as the flag code for Run-Length Encoding (RLE) in the following step. Eventually, we compress the Huffman table utilizing another encoding method to generate additional information for compression and integrate it into the compressed image.

Table 7. The frequency of wavelet values in the 4×4 image block.

Wavelet Values	0	1	2	3	4	5	6	21	36	65,525	65,527	65,528	65,529	65,531	65,533	65,534	65,535
Frequency	4	2	6	3	3	2	2	1	1	1	2	2	1	2	8	4	4

Table 8. The generated Huffman table of the 4×4 image block.

Wavelet Values	Code Length	Huffman Code
0	4	0011
1	5	00001
2	3	110
3	4	0100
4	4	0101
5	4	0110
6	4	0111
21	5	00010
36	5	00011
65,525	6	000000
65,527	4	1000
65,528	4	1001
65,529	6	000001
65,531	5	00100
65,533	3	111
65,534	4	1010
65,535	4	1011
65,536	5	00101

Step 4: Each wavelet coefficient is encoded into a bitstream using the corresponding Huffman code in the pre-defined Huffman table. In addition, if the run-length encoding (RLE) technique can be used to represent a sequence of identical coefficients, a shorter bitstream is generated using the RLE rules. Otherwise, the wavelet coefficients are sequentially encoded using the corresponding Huffman codes. The encoding process is described in detail in Equation (4), where $sCode$ denotes the wavelet coefficient, $runCode$ is the flag code indicating whether RLE is used, and $runCount$ is the number of consecutive identical coefficients represented by RLE. The function $L(\cdot)$ returns the length of the corresponding

Huffman code, and the function $C(\cdot)$ maps the input to the corresponding Huffman code. The symbol $||$ denotes the concatenation operation.

$$\begin{cases} \text{RLE} : L(sCode)||C(sCode)||L(runCode)||C(runCode)||runCount \\ \text{otherwise} : L(sCode)||C(sCode) \end{cases} \quad (4)$$

After the PIZ compression encoding undergoes the four-step process, the resulting pixel values are then output by the Huffman code. To ensure that the output code matches the corresponding Huffman code in the Huffman table and that the inverse wavelet values align with the corresponding bitmap of the image, the encryption and authentication scheme must be meticulously designed. Otherwise, if the decoding process produces an incorrect code, the image software may trigger an alert and prevent the image from being properly displayed.

3. Proposed Scheme

In the pursuit of commutative image encryption and authentication, our primary objective is to segregate the encrypted and embedded portions of the image. The encryption algorithm must safeguard the critical part of the image content, while the authentication mechanism must utilize the less significant portions to prevent the occurrence of severe distortions. Nevertheless, to preserve the OpenEXR file format with PIZ compression encoding, the proposed scheme modifies only the values represented by the corresponding Huffman codes in the chunk data. The histogram of these Huffman-coded values is divided into two parts: (1) an encrypted section, represented by the light gray area, and (2) an embedded section, depicted by the dark gray area. The encryption algorithm encrypts the former portion, while the authentication algorithm embeds data into the latter.

3.1. The Encryption and Embedding Phase

The proposed scheme involves the encryption of image data and the embedding of watermarks into the image after performing Step 4 of PIZ compression encoding. During output processing, each transformed value is mapped to its corresponding Huffman code, as demonstrated in Table 9. Table 8 illustrates the Huffman table of the 4×4 image block utilized in this scheme. Table 9 consists of 48 original output codes for the 4×4 image block, whereas Table 8 contains 18 types of wavelet values along with their corresponding Huffman codes.

In the proposed method, two Huffman-coded values with the same code length are selected as the embedding pair, while the remaining values are assigned to the encryption section. The use of two values enables binary watermark embedding, where one value represents bit 0, and the other represents bit 1. The two values must have identical code length so that replacement between them does not change the compressed bitstream length or impair PIZ decoding. Therefore, the values are not selected arbitrarily in the unrestricted sense; they must satisfy the constraints of equal code length, valid Huffman representation, and sufficient occurrence frequency for the intended watermark length.

For encryption, the encryption key K_E is used to initialize a pseudorandom sequence generator that produces a keystream for Huffman-code mapping. To ensure security, the generator should be instantiated as a cryptographically secure PRNG in practical implementations. Recent studies have explored chaos-based pseudo-random sequence generation, S-box design, and memristor-based chaotic systems to enhance the security and cryptanalytic resistance of image and video encryption schemes [22,23]. These studies provide useful insights into secure sequence generation. However, their technical settings mainly focus on spatial-domain or transform-domain encryption and do not consider PIZ-compressed OpenEXR Huffman-coded structures. Therefore, in this study, the PRNG

is used within a compression-preserving Huffman-code mapping framework rather than as a pixel-domain or transform-domain encryption mechanism.

Table 9. The original output code of each color channel in the 4×4 image block.

Color Channel	Original Output Code of the 4×4 Image Block			
R	00011	00100	1000	0011
	1011	0101	111	111
	111	111	0100	110
	1011	0011	1010	00001
G	00010	000000	1011	111
	0011	0100	1001	110
	110	1000	0110	0101
	1010	110	1011	111
B	0111	1001	1010	0101
	111	111	1010	110
	00001	000001	0110	0011
	0100	00100	110	0111

For each Huffman code length l , the valid codes with the same length are collected into a set S_l . If the original code has an index i in S_l , the encrypted code is mapped to the index

$$i' = (i + r_j) \bmod |S_l|$$

where r_j is the pseudorandom value generated for the j -th coded symbol. This mapping preserves code length and therefore maintains the compressed bitstream size.

For authentication, we utilize SHA-256 to generate a bitstream that serves as the watermark, which is subsequently embedded in the images. The encryption and embedding stages are elaborated in detail below:

Step 1: Two qualified values from the Huffman table are selected for watermark embedding purposes, subject to the constraints of identical code length, a valid Huffman representation, and sufficient occurrence frequency. In the illustrative example shown in Table 8, values 5 and 6 are selected because they have the same code length of 4 and can therefore be interchanged without changing the compressed bitstream length. In a general image, the embedding pair may be selected from other Huffman-coded values that satisfy the same constraints. If the selected pair does not provide sufficient embedding capacity, another qualified pair, or multiple qualified pairs, may be selected according to the required authentication payload.

Step 2: The Huffman tree is considered a tree set with 0 and 1 in different orders. For each level $i, i = 1, 2, \dots, n$, the two branches may swap according to the bit sequence. If the height of the perfect binary tree is n , there are $2^n - 1, n \neq 1$ trees in the tree set. In this step, we utilize the encryption key K_E as the seed of the PRNG to generate random numbers and map each wavelet value, except for the two values of 5 and 6, which are chosen in **Step 1**. As an illustration, consider the initial wavelet coefficient of the red channel, which has a value of 36 and a code length of 5 bits. We gather all Huffman codes of equal length into a set, denoted as {00001, 00010, 00011, 00100, 00101}. The index of the value 36 in this set is 2. Assuming that the first randomly generated number is 2, we add 2 to the index of value 36, resulting in a new index of $(2 + 2) \bmod 5 = 4$, corresponding to a new Huffman code of 00101.

As another example, consider the second wavelet coefficient of the red channel, which has a value of 65,531 and a code length of 5 bits. Again, we select the same set of Huffman codes and determine the index of value 65,531, which is 3. Assuming that the second randomly generated number is 4, we add 4 to the index of value 65,531, giving a new index of $(3 + 4) \bmod 5 = 2$, corresponding to a new Huffman code of 00011.

Step 3: A watermark consisting of a bitstream generated using the SHA-256 cryptographic hash function is proposed. The authentication key K_A is used as the input for SHA-256, and the resulting output is embedded into selected values in Step 1 through value modification. If the embedded bit is 0, the corresponding value is replaced with 5, and the output code is set to 0110. Conversely, if the embedded bit is 1, the corresponding value is replaced with 6, and the output code is set to 0111.

Let v_0 and v_1 denote the two selected Huffman-coded values used to represent embedded bits 0 and 1, respectively. The embedding capacity C of a chunk can be expressed as

$$C = N(v_0) + N(v_1)$$

where $N(v_0)$ and $N(v_1)$ are the numbers of occurrences of the two selected values in the Huffman-coded output. In the 4×4 illustrative example, the output code contains two instances of 0110 and one instance of 0111, resulting in an embedding capacity of three bits. For full images, the total embedding capacity is obtained by summing the available capacity over all processed chunks.

3.2. The Decryption Phase

The decryption process reverses the encryption mapping while keeping the embedded authentication values unchanged.

Step 1: Use the encryption key K_E and the same PRNG initialization rule to regenerate the pseudorandom sequence.

Step 2: For each received Huffman code, determine whether it belongs to the embedding pair. If the code corresponds to v_0 or v_1 , it is skipped during decryption because it carries authentication information.

Step 3: For each encrypted code outside the embedding pair, identify the set S_l containing all valid Huffman codes with the same code length l .

Step 4: If the encrypted code has an index i' in S_l , recover the original index by

$$i = (i' - r_j) \bmod |S_l|$$

Step 5: Replace the encrypted code with the Huffman code corresponding to the index i . Since all replacements are performed within the same code-length set, the compressed bitstream length remains unchanged.

To illustrate, the first output code of the red channel is 00101, with a code length of 5. We select all Huffman codes with the same code length and create a set $\{00001, 00010, 00011, 00100, 00101\}$, where the index of the output code 00101 is 4. The first generated random number is 2, and we subtract it from index 4 to obtain a new index of $(4-2) \bmod 5 = 2$, corresponding to the decrypted code of 00011. For another example, the second output code is 00011, with a length of 5. Therefore, the same tree set is selected, and the index of the output code 00011 is 2. If the second generated random number is 4, we subtract 4 from the index, resulting in a new index of $-2 \bmod 5 = 3$, corresponding to the decrypted code of 00100.

3.3. The Verification Phase

The verification phase comprises two scenarios: (1) verifying the encrypted image prior to decryption, and (2) verifying the decrypted image subsequent to decryption. The commutative property holds because encryption and authentication operate on disjoint subsets of Huffman-coded values. Let E denote the encryption operation applied to the encryption subset S_E , and let A denote the authentication embedding operation applied to the embedding subset S_A . Since

$$S_E \cap S_A = \emptyset \quad (5)$$

the encryption operation does not modify the embedded authentication values, and the authentication operation does not modify the encrypted values. Therefore, for any valid Huffman-coded sequence X ,

$$E(A(X)) = A(E(X)) \quad (6)$$

This commutativity is valid under the following conditions: (1) the encryption and embedding subsets are disjoint; (2) all replacements are made only among valid Huffman codes of identical length; and (3) the selected embedding pair provides sufficient capacity for the authentication payload.

Step 1: The verifier employs the SHA-256 algorithm with an authentication key K_A to generate watermarks for the received image as part of the authentication process.

Step 2: The verification process involves scanning all output codes and identifying the specific codes that match with the two pre-selected values, namely, 0110 and 0111.

Step 3: The watermark is extracted by the verifier based on the selected codes. When the code is 0110, the corresponding extracted bit value is 0. Conversely, if the code is 0111, the corresponding extracted bit value is 1.

Step 4: The verifier validates the identity of the extracted watermark against the watermark generated in **Step 1**. Successful validation indicates that no modification has been detected in the embedded authentication sequence. Conversely, failure to validate the extracted watermark against the generated watermark indicates that the image may have been modified or tampered with.

4. Experimental Results

This section presents the experimental findings of the proposed cryptographic scheme. The primary considerations encompass: (1) the confidentiality of the encrypted and authenticated image, (2) the integrity of both the encrypted and authenticated image and the decrypted and authenticated image, and (3) the imperceptible distortion of the decrypted and authenticated image. The experiments were conducted on the Visual Studio platform, utilizing the ILM's open-source library [24]. To ensure the applicability of the results, seven test images in OpenEXR format (*.exr) [24], as listed in Table 10, were employed throughout the analysis. All HDR images in this section underwent tone-mapping.

Table 10. PSNR of images.

HDR Images	PSNR
Bonita	87.8043
Cannon	85.8876
Desk	88.7328
GoldenGate	91.1624
MtTamWest	77.7918
Kapaa	82.4417
Ocean	93.1851

4.1. Simulations of Commutative Encryption and Authentication

The performance of the proposed scheme is evaluated in terms of: (1) transforming plain images into noise-like cipher images, (2) visual distortion, and (3) format compliance.

Figure 3 showcases the experimental results of the images labeled “Bonita” in Figure 3a and “Cannon” in Figure 4a. The encrypted and authenticated images in Figures 3b and 4b appear noise-like in the tone-mapped display and do not reveal recognizable visual content of the original images. Only authorized participants can recover the original images or, more precisely, the decrypted and authenticated images as shown in Figures 3c and 4c. Hence, the issue of confidentiality can be satisfied using the proposed encryption scheme, while further security analyses will be presented later. Analysis of Figures 3 and 4 reveals that the encrypted and embedded image manifests as a concealed representation, whereas the decrypted and embedded image closely approximates the original image with minimal distortions.

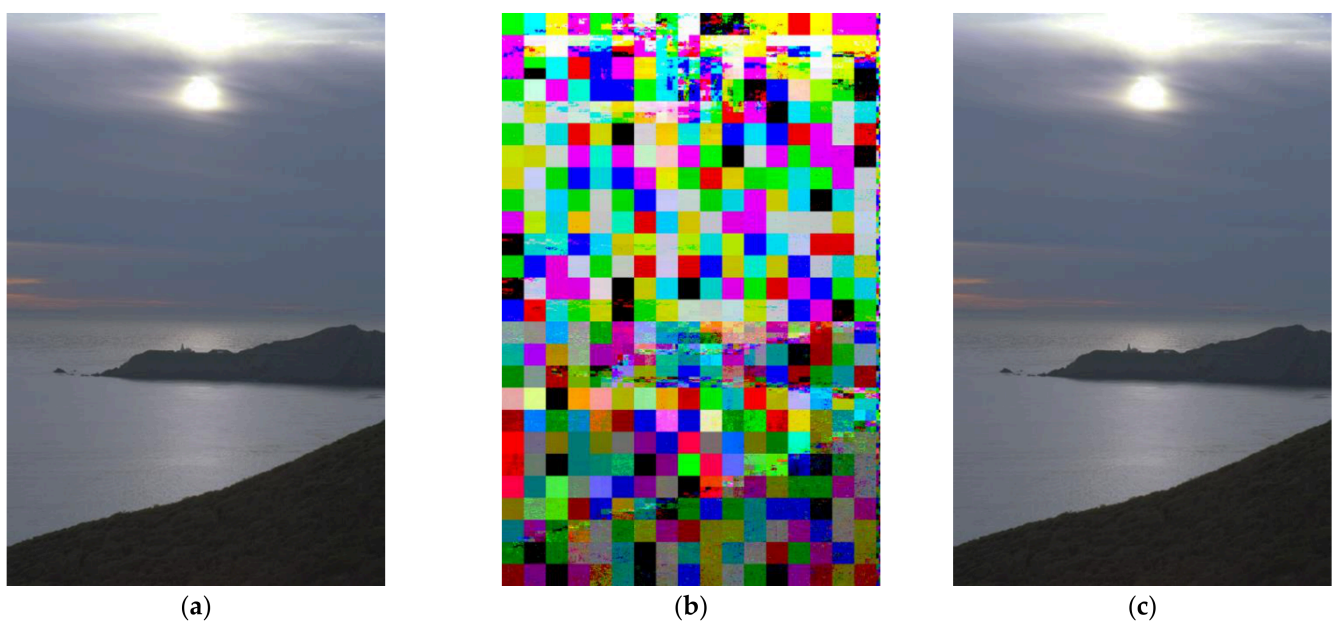


Figure 3. Experimental results of image “Bonita”: (a) original image, (b) encrypted and embedded image, (c) decrypted and embedded image.

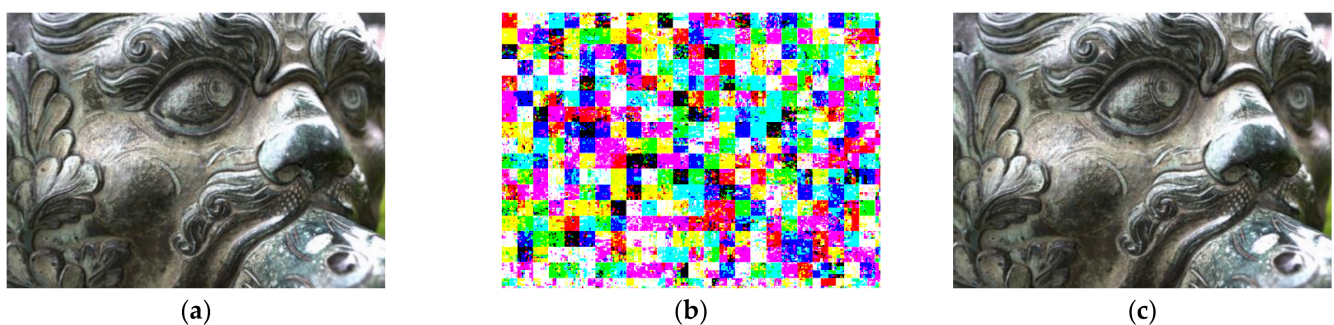


Figure 4. Experimental results of image “Cannon”: (a) original image, (b) encrypted and embedded image, (c) decrypted and embedded image.

Moreover, Figures 3b and 4b clearly demonstrate adherence to formatting standards, wherein the encrypted and authenticated images undergo successful decoding, manifesting as noise-like representations. In the devised framework, authentication bits are intricately integrated into the images, providing a mechanism to discern any potential tampering within the image data. The embedded authentication bits provide a theoretical basis for

detecting modifications that affect the selected Huffman-coded authentication positions. The decrypted and authenticated images depicted in Figures 3c and 4c closely resemble the original images with minimal distortions. The assessment of visual quality is expounded upon in the subsequent discourse.

4.2. Visual Quality of the Tone-Mapped Version of the Decrypted and Authenticated Images

The PSNR evaluation is a widely used method to assess the effectiveness of image authentication schemes and is defined as

$$PSNR = 10 \times \log_{10} \frac{255^2}{MSE} \cdot (\text{dB})$$

and

$$MSE = \frac{1}{3 \times M \times M} \sum_{k \in \{R, G, B\}} \sum_{i=1}^M \sum_{j=1}^M (I_{ij}^k - \tilde{I}_{ij}^k)^2,$$

where I_{ij}^k and $\tilde{I}_{ij}^k$ are the original image and the decrypted and authenticated version. The quality of an image improves with an increase in PSNR.

Since PSNR was originally designed for LDR images, the HDR images were first converted into tone-mapped LDR versions before PSNR calculation. The PSNR values reported in Table 10 should therefore be interpreted as visual-quality indicators under the adopted tone-mapping setting rather than as absolute HDR-domain distortion measures. The same tone-mapping setting was applied consistently to the original images and the decrypted/authenticated images to ensure a fair comparison within this study.

Generally, distinguishing between an authenticated image and its original counterpart becomes challenging for the human visual system when the PSNR exceeds the range of 30 dB [25] to 40 dB [26]. Higher PSNR values signify reduced distortion, and given that the majority of PSNRs exceeded 75 dB, the findings underscore a minimal level of distortion attributable to the embedding process.

5. Security Analysis

This section provides a security analysis of the proposed joint encryption and authentication scheme. The primary security considerations include the confidentiality of the encrypted and authenticated image, integrity of both encrypted-authenticated and decrypted-authenticated images, and imperceptible distortion in decrypted and authenticated images. This analysis examines cryptographic attacks aimed at compromising confidentiality (including ciphertext-only attacks, known-plaintext attacks, and chosen-plaintext attacks), with further evaluation of image encryption security through histogram analysis and correlation analysis. Additionally, malicious attacks targeting integrity are discussed.

5.1. Confidentiality Analysis

The proposed scheme achieves content protection by modifying Huffman coding values in the chunks section of the OpenEXR file structure. Specifically, the histogram corresponding to Huffman coding output values is partitioned into two components: an encryption component for confidentiality and an embedding component for authentication. The encryption algorithm is specifically designed to protect critical portions of the image content.

5.1.1. Histogram Analysis

Histogram analysis is a commonly employed technique to evaluate the efficacy of image encryption schemes. In this study, we present the histogram analysis of the “Bonita” original image, as depicted in Figure 5. Specifically, (a–c) represent the sign, exponent,

and mantissa bits of the red channel, while (d–f) denote the corresponding bits of the green channel, and (g–i) refer to the bits of the blue channel. Figure 6, on the other hand, illustrates the histogram analysis of the encrypted and embedded image “Bonita”. The sign, exponent, and mantissa bits of the red, green, and blue channels are represented by (a–c), (d–f), and (g–i), respectively.

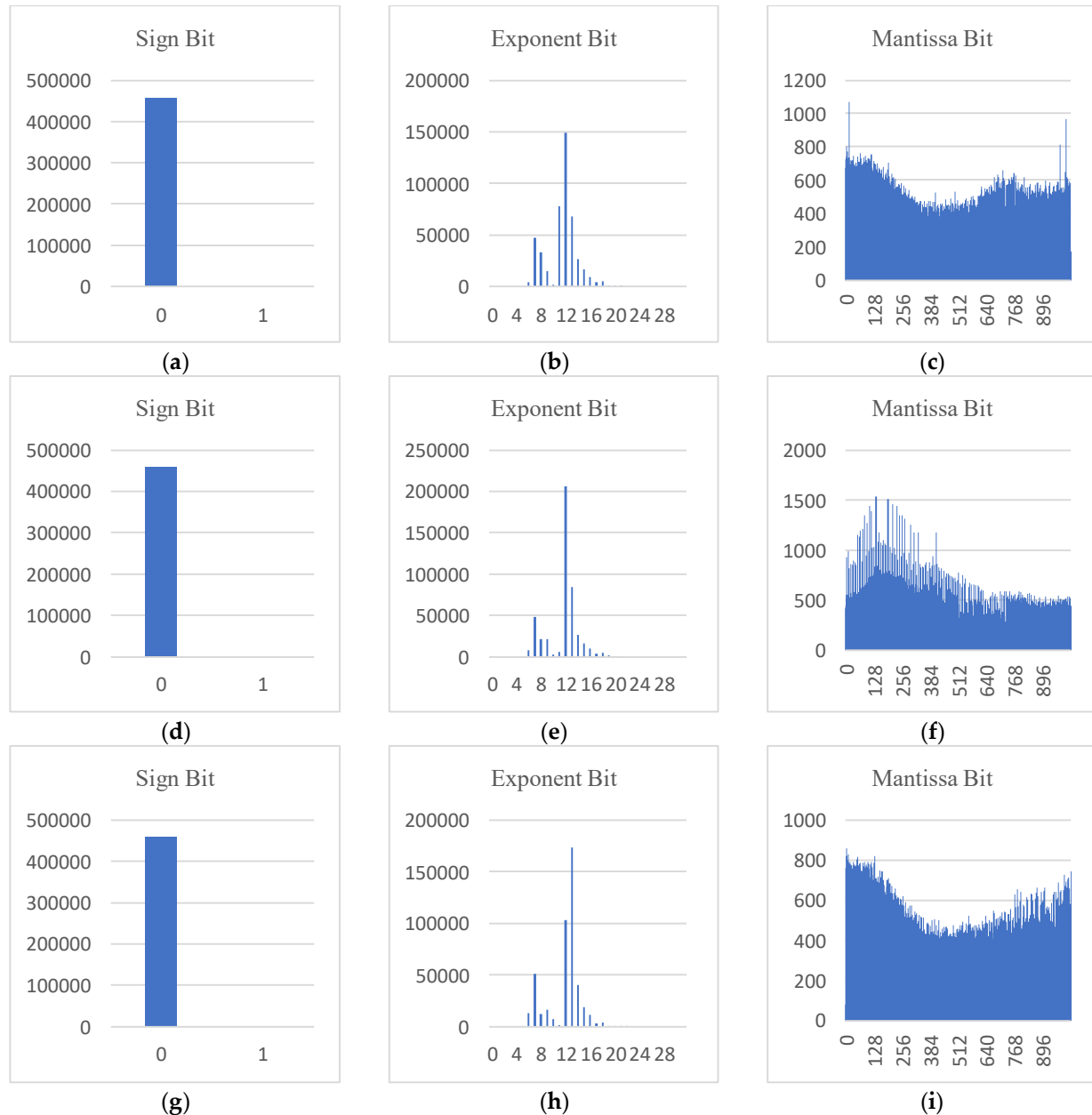


Figure 5. Histogram analysis of the original image “Bonita”: (a–c) are the sign, exponent, and mantissa bits of the red channel; (d–f) are the sign, exponent, and mantissa bits of the green channel; and (g–i) are the sign, exponent, and mantissa bits of the blue channel.

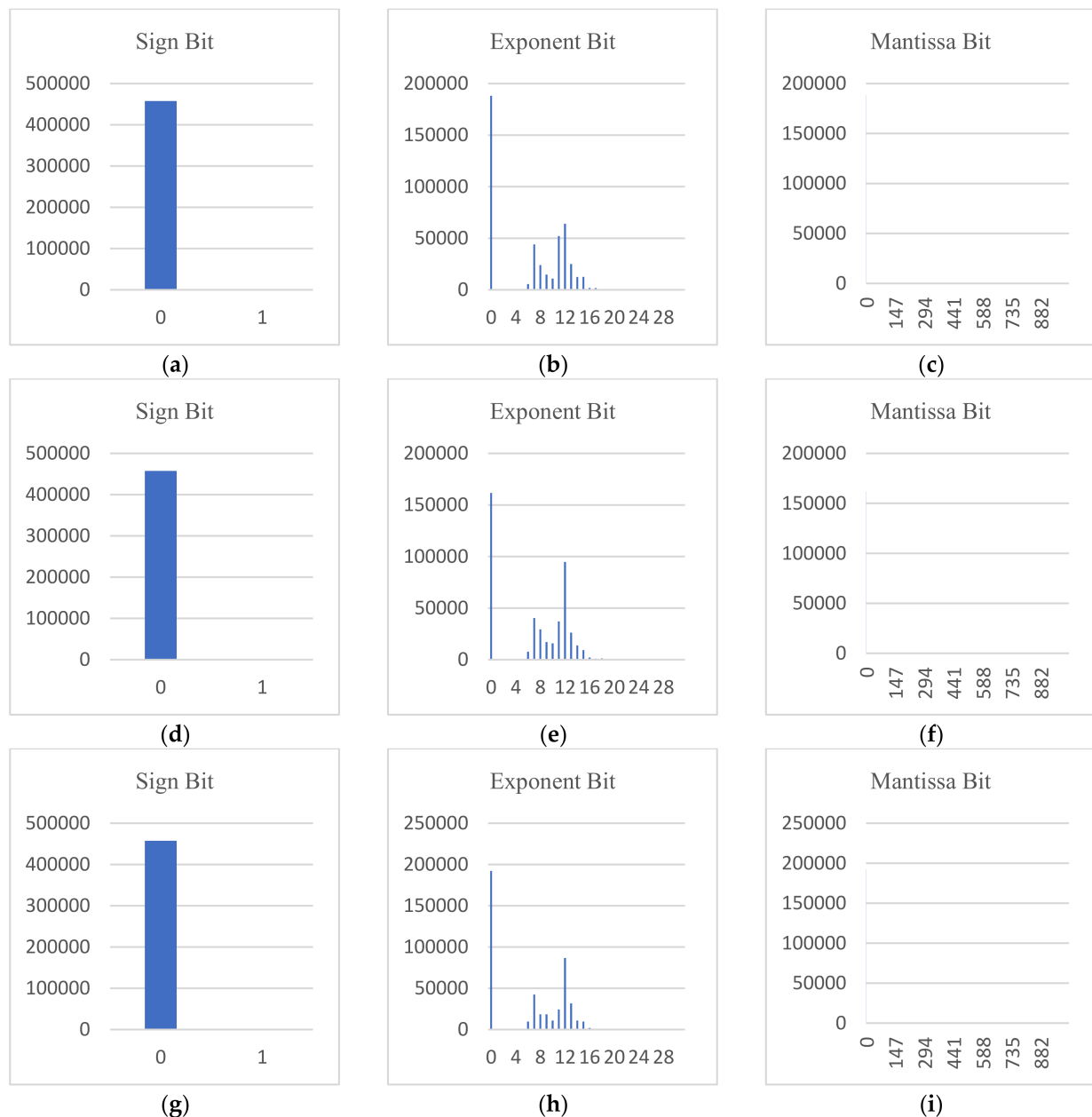


Figure 6. Histogram analysis of the encrypted and embedded image "Bonita": (a–c) are the sign, exponent, and mantissa bits of the red channel; (d–f) are the sign, exponent, and mantissa bits of the green channel; and (g–i) are the sign, exponent, and mantissa bits of the blue channel.

Figure 7 illustrates the histogram analysis of the original image "GoldenGate": (a–c) correspond to the sign, exponent, and mantissa bits of the red channel; (d–f) correspond to the sign, exponent, and mantissa bits of the green channel; and (g–i) correspond to the sign, exponent, and mantissa bits of the blue channel. Figure 8 depicts the histogram analysis of the encrypted and embedded image "GoldenGate": (a–c) correspond to the sign, exponent, and mantissa bits of the red channel; (d–f) correspond to the sign, exponent, and mantissa bits of the green channel; and (g–i) correspond to the sign, exponent, and mantissa bits of the blue channel. By comparing Figures 5–8, it can be observed that the histogram of the encrypted and embedded image has assumed a different distribution. The histograms of each color are similar, and the frequency of value 0 is higher than that of other values. The histogram analysis results indicate that the distinctive features of the image content have been removed.

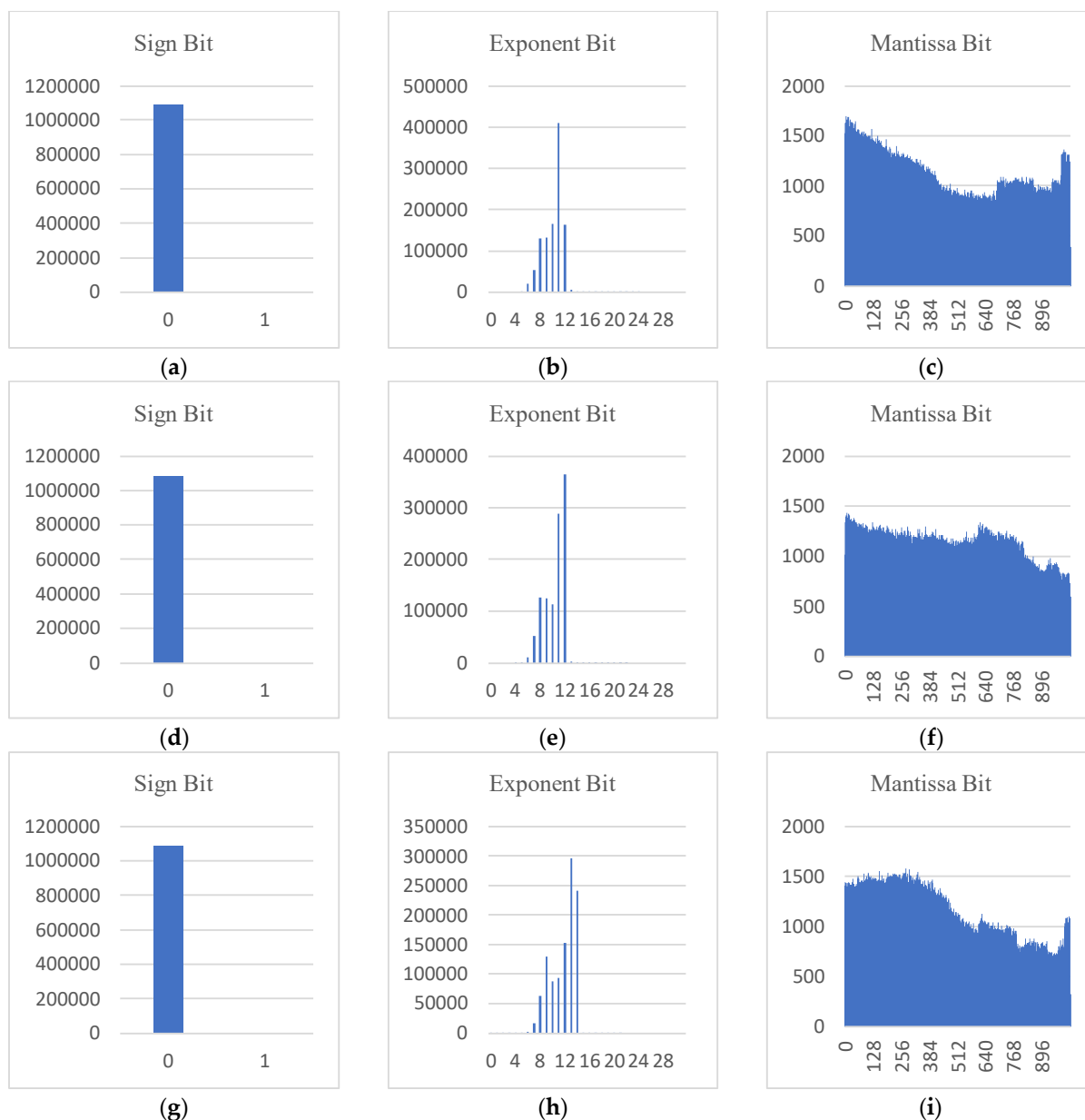


Figure 7. Histogram analysis of the original image "GoldenGate": (a–c) are the sign, exponent, and mantissa bits of the red channel; (d–f) are the sign, exponent, and mantissa bits of the green channel; and (g–i) are the sign, exponent, and mantissa bits of the blue channel.

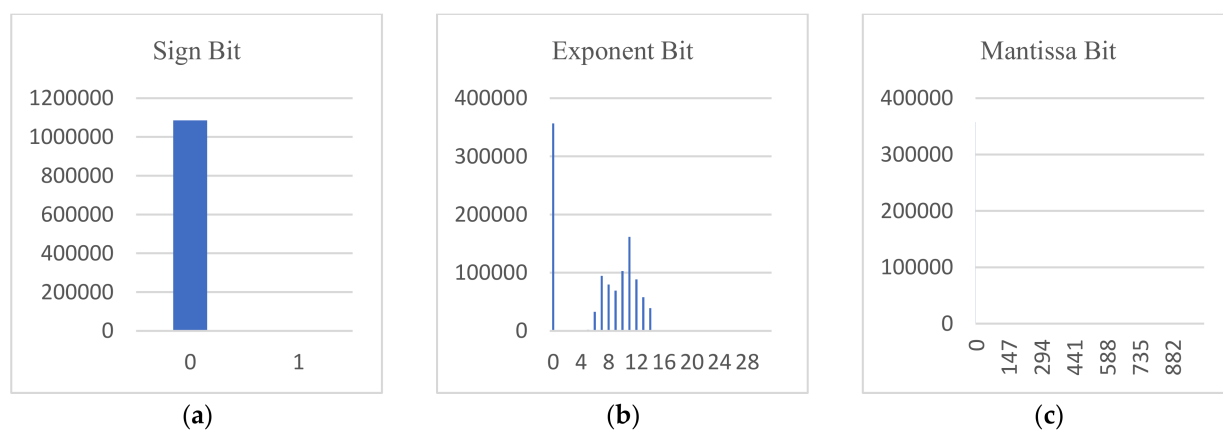


Figure 8. Cont.

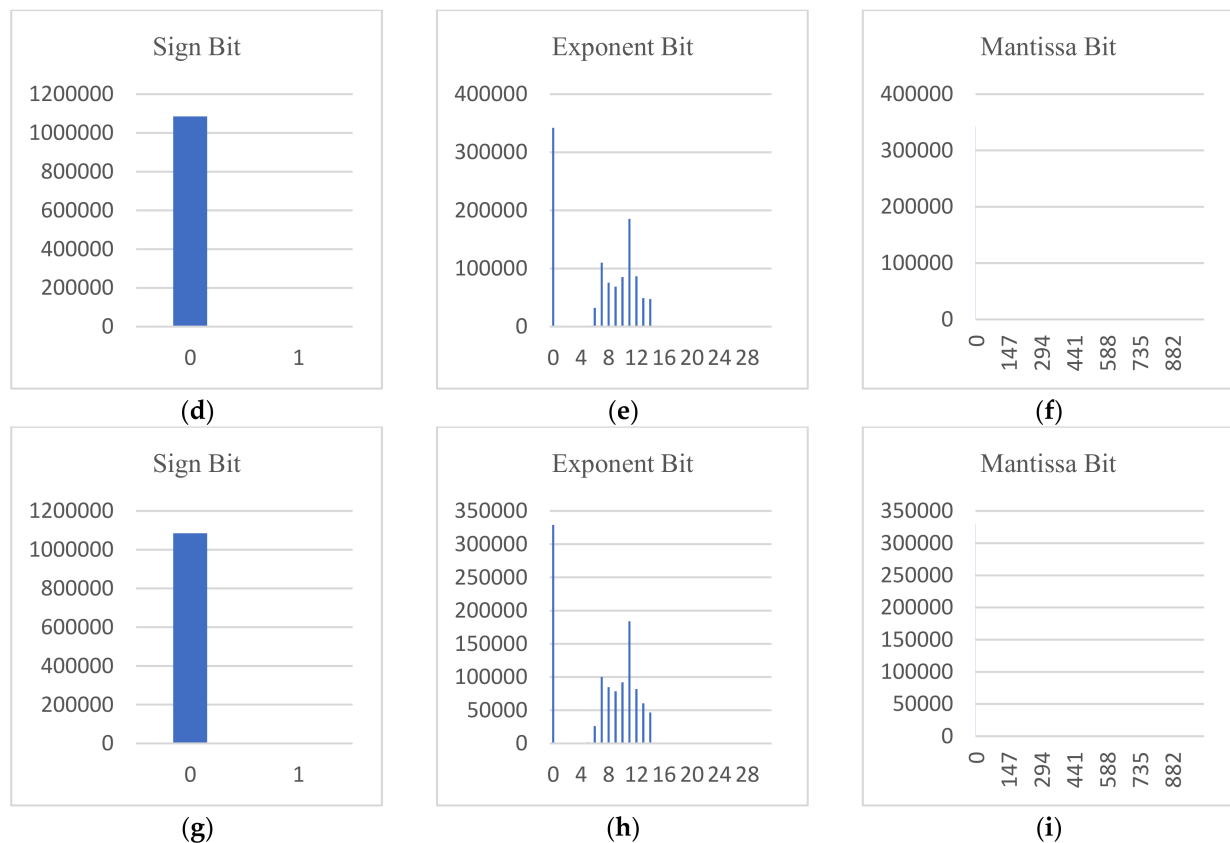


Figure 8. Histogram analysis of the encrypted and embedded image “GoldenGate”: (a–c) are the sign, exponent, and mantissa bits of the red channel; (d–f) are the sign, exponent, and mantissa bits of the green channel; and (g–i) are the sign, exponent, and mantissa bits of the blue channel.

5.1.2. Correlation Analysis

Correlation analysis is a widely used technique in evaluating the performance of both image encryption and authentication schemes. The following equations are adapted to calculate the correlation coefficient. Suppose x and y are values of two adjacent pixels. The means of the x and y sets are first calculated, and the covariance between the two sets is then computed as follows:

$$\text{cov}(x, y) = \frac{\sum_{i=1}^n (x_i - \bar{x}_i)(y_i - \bar{y}_i)}{n - 1}, \text{ where } \bar{x}_i = \frac{\sum_{i=1}^n x_i}{n} \text{ and } \bar{y}_i = \frac{\sum_{i=1}^n y_i}{n}.$$

The sample standard deviation of the x set and the y set and, finally, the correlation $\text{corr}(x, y)$ are calculated by

$$\text{corr}(x, y) = \frac{\text{cov}(x, y)}{s_x s_y}, \text{ where } s_x = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x}_i)^2}{n - 1}} \text{ and } s_y = \sqrt{\frac{\sum_{i=1}^n (y_i - \bar{y}_i)^2}{n - 1}}.$$

This technique assesses the degree of correlation between adjacent pixels in an image. Typically, prior to encryption, adjacent pixels exhibit high correlation due to the inherent nature of image content. Conversely, after encryption, adjacent pixels typically display reduced correlation as the encryption scheme destroys the original correlation structure within the image.

Table 11 presents the correlations of the original images, decrypted and embedded images, and encrypted and embedded images. Two observations can be made from these results. First, the correlations of the decrypted and embedded images are almost identical to those of the original images. This indicates that the authentication process introduces

only minimal distortion and does not significantly alter the structural characteristics of the recovered images. Second, the correlation results should be interpreted with caution because the proposed method operates in the compressed Huffman-coded domain rather than directly performing spatial-domain pixel permutation. For most test images, the encrypted and embedded images show reduced correlations compared with the original images. However, for GoldenGate, the encrypted image exhibits higher correlation coefficients than the original image in all three channels. This suggests that correlation reduction is not guaranteed for every image when encryption is constrained to valid equal-length Huffman-code substitution. Therefore, correlation analysis alone is insufficient for evaluating the confidentiality of the proposed compression-preserving scheme and should be considered together with visual inspection, histogram analysis, format compliance, and cryptanalytic discussion. As shown in Figures 3–8, the encrypted and embedded images appear noise-like and no longer reveal recognizable visual content. These results indicate that the proposed encryption process effectively obscures image content while preserving the decodability of the OpenEXR file format.

Table 11. Correlations of the images.

HDR Images	Color Channels	Correlations of Original Images	Correlations of Decrypted and Embedded Images	Correlations of Encrypted and Embedded Images
Bonita	Red	0.994204	0.994204	0.923951
	Green	0.994308	0.994308	0.952773
	Blue	0.9903	0.9903	0.944776
Cannon	Red	0.953931	0.953931	0.832219
	Green	0.952203	0.952203	0.851516
	Blue	0.95967	0.95967	0.822685
Desk	Red	0.956225	0.956225	0.713613
	Green	0.95354	0.95354	0.523129
	Blue	0.953471	0.953472	0.594866
GoldenGate	Red	0.516995	0.516995	0.801798
	Green	0.425434	0.425434	0.654734
	Blue	0.765963	0.765963	0.963931
MtTamWest	Red	0.990965	0.990965	0.96373
	Green	0.992769	0.992769	0.964589
	Blue	0.994037	0.994037	0.967152
Kapaa	Red	0.994929	0.994929	0.919169
	Green	0.994776	0.994772	0.949628
	Blue	0.995245	0.995242	0.954155
Ocean	Red	0.946433	0.946433	0.342596
	Green	0.927255	0.927255	0.321657
	Blue	0.919599	0.919599	0.218623

5.2. Integrity Analysis

The proposed scheme is designed to detect modifications that alter the embedded authentication bits or the Huffman-coded values associated with authentication. Therefore, the integrity protection discussed in this section is based on the theoretical sensitivity of

the embedded SHA-256-based authentication bits. Because the present study focuses on the design and theoretical analysis of the compression-unimpaired joint encryption and authentication mechanism, quantitative claims regarding detection rate, false alarm rate, or localization accuracy are not made. Image authentication serves as a secondary line of defense for image security, with the objective of maintaining the integrity of image content. Embedded authentication bits provide a mechanism for identifying any potential tampering within the image data.

5.2.1. Tamper Detection

In practical scenarios, attackers lacking encryption keys cannot interpret or modify the encrypted HDR image meaningfully. Any tampering, particularly to Huffman-coded blocks, alters unreadable data, affecting embedded authentication bits. While decoders may treat such changes as noise, such modifications may alter the embedded authentication sequence and cause verification failure. The sensitivity of the verification process is therefore discussed from a theoretical perspective based on the behavior of the embedded SHA-256-based authentication bits. A systematic evaluation of detection rate, false alarm rate, and localization accuracy under different tampering operations will be investigated in future work.

5.2.2. Resistance to Specific Attacks

The scheme is expected to be sensitive to unauthorized modifications that affect the selected Huffman-coded authentication positions. Typical manipulations, such as content replacement or local modification, may alter the embedded authentication sequence and lead to verification failure. The present study focuses on the design and theoretical analysis of the compression-unimpaired joint encryption and authentication mechanism. A systematic quantitative evaluation of different tampering types is left for future work. Due to the use of the one-way hash function SHA-256, even if an attacker obtains authentication bits and corresponding ciphertext/plaintext portions and other information, it remains computationally infeasible to deduce the key K_A from the one-way hash function. This makes the forgery of authenticated images computationally infeasible.

5.3. Cryptanalysis Attack Analysis

We consider three attack scenarios—ciphertext-only, known-plaintext, and chosen-plaintext—and derive corresponding key management rules:

- **Ciphertext-Only Attacks:** An adversary with only the encrypted image relies on brute force to recover K_E . Hence, K_E must have sufficient bit length to render exhaustive search infeasible.
- **Known-Plaintext and Chosen-Plaintext Attacks:** When attackers obtain or select plaintext–ciphertext pairs, reuse of K_E would allow key inference. Therefore, K_E is generated fresh per session as a one-time seed for the PRNG (OTP style), ensuring that compromise of one session key does not affect others.
- **Hash-Based Forgery Attacks:** The authentication key K_A is used as secret input to SHA-256 for a MAC. Thanks to SHA-256's one-way and collision-resistant properties, recovering K_A or finding collisions is computationally infeasible. While longer K_A further protects against brute-force-style guessing, its primary security derives from the hash function's strength.

Let $|K_E|$ and $|K_A|$ denote the bit lengths of the encryption key and authentication key, respectively. If the two keys are generated independently, the theoretical key space is

$$2^{|K_E|} \times 2^{|K_A|} = 2^{|K_E|+|K_A|}$$

For example, if both K_E and K_A are selected with at least 128-bit security strength, the combined key space is at least 2^{256} . The confidentiality of the encrypted component depends on the unpredictability of the PRNG keystream initialized by K_E , while the integrity protection depends on the one-way and collision-resistant properties of SHA-256 with secret input K_A . Therefore, practical security requires independent key generation, non-reuse of K_E across sessions, and a sufficient key length for both keys.

In summary, confidentiality is supported by using a sufficiently long and freshly generated encryption seed for the PRNG, while integrity relies on the cryptographic robustness of SHA-256 with the authentication key. Together, these mechanisms provide resistance against standard cryptanalytic and forgery attacks.

5.4. Complexity Analysis

Let N denote the number of Huffman-coded symbols in the processed compressed chunks, M_l denote the number of valid Huffman codes with code length l , and C denote the number of selected embedding positions. The encryption phase scans the Huffman-coded symbols once and performs an index mapping within the same code-length set. If the code-length sets are indexed in advance, each mapping requires constant-time lookup; therefore, the time complexity of encryption is $O(N)$. The decryption phase performs the inverse mapping using the same pseudorandom sequence and also has time complexity $O(N)$. The authentication embedding phase scans the selected embedding positions and modifies the corresponding codes, resulting in $O(C)$ time, where $C \leq N$. The verification phase similarly scans the coded sequence to extract authentication bits and compare them with the SHA-256-generated watermark, resulting in $O(N)$ time in the worst case.

The space complexity is mainly determined by the code-length index tables and the chunk buffer. Since the proposed method processes the compressed data chunk by chunk, the required memory is proportional to the size of one processed chunk rather than the full image. Therefore, the proposed scheme is suitable for streaming-style processing and can be adapted to high-throughput transmission scenarios, provided that secure key management and PRNG initialization are properly implemented.

5.5. Further Discussions

Table 12 presents a qualitative feature-level comparison between existing HDR image security methods and the proposed scheme. It should be noted that, to the best of our knowledge, no existing method addresses the same setting as this study: commutative encryption and authentication directly performed in the PIZ-compressed OpenEXR Huffman-coded domain. Therefore, quantitative comparisons in terms of PSNR, runtime, compression ratio, and tamper detection rate would not constitute a fair like-for-like comparison unless the competing methods were redesigned and reimplemented for the same compressed-domain OpenEXR setting. For this reason, Table 12 focuses on functional compatibility, including file format, confidentiality, integrity, commutativity, compression-domain operation, and compression-unimpaired processing.

Table 12. Comparison between the related work and the proposed scheme.

	Tsai et al. [14]	Lin et al. [15]	Chen & Chang [16]	Chen & Yan [17]	The Proposed
File format	Radiance RGBE	LogLuv TIFF	OpenEXR	OpenEXR	OpenEXR
Format compliant	Yes	Yes	Yes	Yes	Yes
Confidentiality	N/A	Yes	Yes	Yes	Yes
Integrity	Yes	N/A	N/A	Yes	Yes
Commutativity	N/A	N/A	N/A	Yes	Yes
Compression domain	N/A	N/A	N/A	N/A	Yes
Compression-unimpaired	N/A	N/A	N/A	N/A	Yes

The method by Tsai et al. [14] utilizes the Radiance RGBE format, while Lin et al. [15] employ LogLuv TIFF. More recent works by Chen & Chang [16], and Chen & Yan [17], as well as the proposed method, adopt the OpenEXR format, which is widely recognized for its robustness and suitability for HDR content. All referenced approaches, including the proposed one, are format-compliant, ensuring compatibility with standard HDR workflows.

With respect to security features, the proposed method provides a complementary advantage over previous HDR image security schemes by jointly considering confidentiality, integrity, commutativity, and compression-domain operation. Since the comparison in Table 12 is qualitative and based on supported functions rather than newly conducted benchmark experiments, the results should be interpreted as a feature-level comparison rather than a quantitative performance ranking. While earlier methods provide limited or partial support for these properties—e.g., Tsai et al. [14] focus on integrity, and Lin et al. [15] focus on confidentiality—the proposed scheme is designed to jointly support confidentiality, integrity, commutativity, compression-domain processing, and compression-unimpaired operation under the PIZ-compressed OpenEXR setting. This ensures that encryption does not interfere with compression efficiency, which is crucial for the storage and transmission of HDR imagery.

Existing HDR image security schemes do not operate under the same problem setting as the proposed method, namely commutative encryption and authentication directly in the PIZ-compressed OpenEXR Huffman-coded domain. Therefore, quantitative comparisons of PSNR, runtime, compression ratio, and tamper detection rate on the same benchmark would not constitute a fair like-for-like comparison.

In summary, the proposed method adheres to format standards and jointly supports confidentiality, authentication, commutativity, and compression-unimpaired operation, making it suitable for secure HDR image applications in compressed OpenEXR-based workflows.

- (1) **Robust Confidentiality and Cryptanalysis Resistance:** Content protection is theoretically guaranteed by partitioning the histogram of the Huffman coding values. The PRNG-driven mapping operates exclusively on the designated encryption component and aims to obscure the semantic structures of the original image while preserving the compressed bitstream syntax. As discussed in the correlation analysis, correlation reduction is not guaranteed for every test image because the proposed method operates under equal-length Huffman-code substitution constraints. By employing an OTP-style dynamic seed update per session, the scheme elevates security depth, rendering the system computationally secure against ciphertext-only, known-plaintext, and chosen-plaintext attacks.

- (2) **Format Compliance via Syntactic Preservation:** Unlike traditional encryption methods that frequently corrupt file headers or bitstream syntax, the proposed approach maintains absolute compatibility with standard HDR decoders. Because the modifications are confined to substituting valid Huffman codes with alternative codes of identical lengths within the chunks section, the structural integrity of the OpenEXR format remains intact. Standard CODECs interpret the encrypted bitstream as valid, high-frequency data, displaying a noise-like representation instead of triggering “corrupted file” errors.
- (3) **Theoretical Sensitivity to Authentication-Bit Modification:** The integrity architecture establishes a robust secondary line of defense. In a realistic attack scenario, an adversary lacking the encryption key cannot execute targeted or meaningful modifications in the decrypted domain. Blind tampering with ciphertext blocks may alter the Huffman-coded values associated with authentication. If the embedded SHA-256-based authentication sequence is changed, the verification process will fail. Since this study does not provide quantitative tamper detection experiments, the tamper sensitivity of the proposed scheme is discussed theoretically.
- (4) **Fidelity Preservation:** The scheme minimizes visual distortion by restricting watermark embedding to less significant, pre-selected segments of the Huffman histogram. Because the embedded data introduces only minor value modifications, the tone-mapped results show high PSNR values in the tested images, indicating limited visual distortion under the adopted tone-mapping setting.
- (5) **Operational Commutativity:** By completely isolating the encryption and embedding segments within the partitioned histogram, the scheme achieves full mathematical commutativity. This independence allows pipeline operators to perform watermark verification either before or after image decryption without strict sequence constraints, offering enhanced flexibility for distributed computing workflows.
- (6) **Compression-Unimpaired Operations:** A major limitation of conventional image encryption is the degradation of compression ratios due to the destruction of spatial redundancy. The proposed scheme overcomes this by operating directly within the compressed domain. Because the proposed substitutions are restricted to valid Huffman codes of identical length, the bitstream length of the modified Huffman-coded data is preserved. Under this constraint, the proposed method does not introduce additional payload length into the compressed chunk data. This supports compression-unimpaired operation in the sense of preserving the compressed bitstream size during encryption and authentication.

6. Conclusions and Future Work

In this paper, we propose commutative encryption and authentication schemes for HDR images in compressed format. To the best of our knowledge, this is the first study to investigate commutative joint encryption and authentication directly in the PIZ-compressed OpenEXR Huffman-coded domain while preserving compression syntax. The commutative property of the proposed image encryption and authentication schemes allows users to verify received images both before and after decryption, making these schemes more practical for real-world applications. The proposed scheme is specifically designed for the OpenEXR image format with PIZ compression encoding. A key advantage of our approach is that the file size of the compressed image remains unchanged after encryption and watermark embedding. Several limitations and future research directions remain. First, this study focuses on PIZ-compressed OpenEXR images; extending the scheme to other OpenEXR compression modes such as ZIP, ZIPS, and RLE remains future work. Second, this study emphasizes compression-preserving design and theoretical security

analysis; more extensive quantitative evaluations, including encryption/decryption time, authentication time, tamper detection rate, false alarm rate, and localization accuracy, will be conducted in future work. Third, while this paper focuses on flat images with 16-bit half floating-point channels, OpenEXR also supports deep images and other data types, such as 32-bit unsigned integers and 32-bit floating-point channels. Extending the proposed scheme to these settings is another important direction.

Author Contributions: All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by J.-Y.Y. and T.-H.C. The first draft of the manuscript was written by J.-Y.Y., and all authors commented on previous versions of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Science and Technology Council of Taiwan under grant NSTC 110-2221-E-415-006-MY2 and 114-2221-E-415-013.

Data Availability Statement: The datasets generated and/or analyzed during the current study are available from the corresponding author upon reasonable request. The source code used for HDR image processing and compression is based on the publicly available OpenEXR library developed by Industrial Light & Magic (ILM), which can be accessed at <http://www.openexr.com>.

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

References

- Peng, K.-W.; Chiang, J.-C.; Chen, S.-G.; Lin, Y.-S. HDR-CNF: Single-image high dynamic range imaging based on conditional normalizing flows. *Multimed. Tools Appl.* **2025**, *84*, 29491–29511.
- Hong, W.; Zhang, H.; Ma, J. MERF: A practical HDR-Like image generator via mutual-guided learning between multi-exposure registration and Fusion. *IEEE Trans. Image Process.* **2024**, *33*, 2361–2376. [[CrossRef](#)] [[PubMed](#)]
- Jiang, Q.; Li, X.; Wang, X.; Wang, Z.; Zhai, G. Dataset and Metric for Quality Assessment of HDR Tone Mapping: Detail Visibility, Color Naturalness, and Overall Quality. *IEEE Trans. Multimed.* **2025**, *27*, 4058–4071. [[CrossRef](#)]
- Han, J.; Yang, Y.; Duan, P.; Zhou, C.; Ma, L.; Xu, C.; Huang, T.; Sato, I.; Shi, B. Hybrid high dynamic range imaging fusing neuromorphic and conventional images. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 8553–8565. [[CrossRef](#)] [[PubMed](#)]
- Shen, X.; Pan, L.; Ni, Z.; He, Y.; Yang, W.; Wang, S.; Kwong, S. Breaking Boundaries: Unifying Imaging and Compression for HDR Image Compression. *IEEE Trans. Image Process.* **2025**, *34*, 510–521. [[CrossRef](#)]
- Cheng, Y.-M.; Wang, C.-M. A novel approach to steganography in high-dynamic-range images. *IEEE Multimed.* **2009**, *16*, 70–80. [[CrossRef](#)]
- Yu, C.-M.; Wu, K.-C.; Wang, C.-M. A distortion-free data hiding scheme for high dynamic range images. *Displays* **2011**, *32*, 225–236. [[CrossRef](#)]
- Chang, C.-C.; Nguyen, T.-S.; Lin, C.-C. Distortion-free data embedding scheme for high dynamic range images. *J. Electron. Sci. Technol.* **2013**, *11*, 20–26.
- Li, M.-T.; Huang, N.-C.; Wang, C.-M. A data hiding scheme for high dynamic range images. *Int. J. Innov. Comput. Inf. Control* **2011**, *7*, 2021–2035.
- Lin, Y.-T.; Wang, C.-M.; Chen, W.-S.; Lin, F.-P.; Lin, W. A novel data hiding algorithm for high dynamic range images. *IEEE Trans. Multimed.* **2016**, *19*, 196–211. [[CrossRef](#)]
- Lan, C.-F.; Wang, C.-M.; Lin, W. XtoE: A novel constructive and camouflaged adaptive data hiding and image encryption scheme for high dynamic range images. *Appl. Sci.* **2022**, *12*, 12856. [[CrossRef](#)]
- Autrusseau, F.; Goudia, D. Non linear hybrid watermarking for high dynamic range images. In *2013 IEEE International Conference on Image Processing*; IEEE: Piscataway, NJ, USA, 2013; pp. 4527–4531.
- Luo, T.; Jiang, G.; Yu, M.; Xu, H.; Gao, W. Robust high dynamic range color image watermarking method based on feature map extraction. *Signal Process.* **2019**, *155*, 83–95. [[CrossRef](#)]
- Tsai, Y.-Y.; Liu, H.-L.; Ying, C.-Y. Applying homogeneity index modification to high-capacity high-dynamic-range image authentication with distortion tolerance. *Multimed. Tools Appl.* **2022**, *81*, 24957–24976. [[CrossRef](#)]
- Lin, K.-S.; Chen, T.-H.; Lin, C.-H.; Chang, S.-S. A tailor-made encryption scheme for high-dynamic range images. In *Genetic and Evolutionary Computing: Proceedings of the Seventh International Conference on Genetic and Evolutionary Computing, ICGEC 2013, Prague, Czech Republic, 25–27 August 2013*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 183–192.
- Chen, T.-H.; Chang, S.-S. Image encryption on HDR images for OpenEXR Format. *Int. J. Eng. Sci.* **2014**, *4*, 19–23.

17. Chen, T.-H.; Yan, J.-Y. Commutative encryption and authentication for OpenEXR high dynamic range images. *Multimed. Tools Appl.* **2021**, *80*, 27807–27828. [CrossRef]
18. Ward, G. Real pixels. In *Graphics Gems II*; Morgan Kaufmann: San Francisco, CA, USA, 1991; Volume 2, pp. 80–83.
19. Larson, G.W. Logluv encoding for full-gamut, high-dynamic range images. *J. Graph. Tools* **1998**, *3*, 15–31. [CrossRef]
20. Kainz, F.; Bogart, R.; Stanczyk, P.; Hillman, P. Technical introduction to OpenEXR. *Ind. Light Magic* **2009**, *21*. Available online: <https://exse.eyewated.com/fls/ad897289e5f3134a.pdf> (accessed on 6 June 2026).
21. Debevec, P.; Reinhard, E.; Heidrich, W.; Ward, G.; Myszkowski, K.; Pattanaik, S. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*; Elsevier Science: Amsterdam, The Netherlands, 2010.
22. Lin, Y.; Wei, Y.; Chen, D.; Li, Y.; Erkan, U.; Toktas, A.; Gao, S.; Zhang, Y. Cryptanalysis and improvement of a video cryptosystem via chaos and S-box. *ACM Trans. Multimed. Comput. Commun. Appl.* **2026**, *22*, 1–28. [CrossRef]
23. Zhang, B.; Zhou, N.-R.; Zhou, S.; Wang, M.-M. Novel 4D chaotic system based on memristor and its application in secure color image encryption. *Digit. Signal Process.* **2025**, *170*, 105791. [CrossRef]
24. OpenEXR. OpenEXR Downloads. Industrial Light & Magic. Available online: <https://www.openexr.com/> (accessed on 6 June 2026).
25. Chen, T.-S.; Chang, C.-C.; Hwang, M.-S. A virtual image cryptosystem based upon vector quantization. *IEEE Trans. Image Process.* **1998**, *7*, 1485–1488. [CrossRef] [PubMed]
26. Lin, P.-Y.; Lee, J.-S.; Chang, C.-C. Protecting the content integrity of digital imagery with fidelity preservation. *ACM Trans. Multimed. Comput. Commun. Appl. (TOMM)* **2011**, *7*, 1–20. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.