IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 18, 2018, accepted August 13, 2018, date of current version September 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2866153

A Watermark-Based Scheme for Authenticating JPEG Image Integrity

OH-JIN KWON, SEUNGCHEOL CHOI, AND BEOMYEOL LEE
Department of Electrical Engineering, Sejong University, Seoul 05006, South Korea

Corresponding author: Seungcheol Choi (choisc@sju.ac.kr)

This work was supported by the Standard Technology Development and Spread Program of KATS/KEIT through the Developing Standard Platform and Library for JPEG privacy-security under Grant 10069128.

ABSTRACT JPEG is the most common image format in smartphones, computers, or digital cameras. Because numerous JPEG images are easily shared and distributed, there are privacy and security concerns for these images. Hence, the JPEG committee has started a standardization called JPEG privacy and security to resolve these issues. The forgery detection of a JPEG image is the key objective of JPEG privacy and security. In this paper, we propose a novel forgery detection method that is watermark-based, for authenticating JPEG image integrity in the discrete cosine transform (DCT) domain. The proposed method aims at 100% detection accuracy for typical forgeries on JPEG image DCT blocks and to counter a well-known forgery attack called collage attack. For this purpose, the method splits the host image into a group of blocks (GOB). A watermark is generated by collaborating with the neighboring GOBs and is embedded into the GOBs. The experimental results using various sample images show the superiority of the proposed method, exhibiting a negligible visual difference between the original and watermarked JPEG images.

INDEX TERMS JPEG privacy and security, image watermarking, integrity authentication, image forgery detection.

I. INTRODUCTION

Recently, it is reported that several billions of digital images are produced per day and shared via social media on the Internet [1]. It is also reported that a majority of these images are in the form of JPEG (ITU-T T.81 (1992) | ISO/IEC 10918-1) compressed file format [2]. Meanwhile, well-known image processing tools have resulted in numerous forged images causing serious social problems. The JPEG committee (SC29/WG1) recognized that the current JPEG file format is vulnerable to image forgery and initiated a new activity called JPEG Privacy and Security [2]. JPEG Privacy and Security intends to provide a degree of confidence while sharing JPEG image files by offering technical solutions for resolving the privacy and security issues. Therefore, forgery detection of both the image data and metadata composing a JPEG file is a key objective of JPEG Privacy and Security.

Image forgery detection techniques can be classified into three categories based on their nature of integrity authentication mechanism, namely, watermark-based, signature-based, and passive techniques [3]–[5]. A watermark-based technique embeds an invisible noise-like set of bits (called a watermark), which is sensitive to modification, into the original image and distributes the watermarked image. Authentication is performed by checking the existence of the watermark in the image being examined.

A signature-based technique extracts a set of features from the original image that is significantly smaller than the original image in size. This set is stored as a file and characterized as an intrinsic signature of each image. When such an image is forged, different signatures are extracted from the original and forged images correspondingly, and the authenticity is determined by comparing these two signatures. The main difference between watermark- and signature-based techniques is that the latter do not demand an embedding process that requires the original image to be changed (even if the change is normally negligible to human eyes); however, it necessitates an image to always be accompanied by the stored signature for its authentication [6]. Another difference is that most watermark-based techniques may identify the modified region in a forged image whereas the signature-based technique may not [7]–[9].

A passive technique [3], [4], [11] has an advantage that it requires only the image being examined for its detection process without any assistance from a signature or watermark. This technique assumes that there is a high probability for image forgeries leaving a clue disturbing the underlying statistical consistency of a natural scene image, even if the clue is not visible to human eyes. The statistical consistency that this technique uses for forgery detection includes the regularities of an image based on illumination, lens radial distortion, chromatic aberration, sensor pattern noise, dust pattern, camera response function, projective geometry, color filter array, inter pixel correlation, and JPEG compression distortion. However, the limitation of a passive technique is that the detection accuracy is much lower than that of the watermark- and signature-based techniques. Both watermark- and signature-based techniques aim for 100% detection accuracy for typical forgeries of natural scene images, whereas a passive technique is known to be typically accompanied by a certain number of both false positive and false negative detections [3].

In this paper, we propose a watermark-based scheme that can be used as a solution for the JPEG Privacy and Security standardization. Based on a thorough review of industrial demands, the JPEG committee has defined several basic requirements for its solutions of authenticating JPEG images in the call for proposal of JPEG Privacy and Security as summarized in Table 1 [2]. During the last decades, many watermarking algorithms for authenticating JPEG images have been proposed. Examples of them are preferably referred to [12]–[17]. Compared to existing algorithms, the proposed scheme can be characterized in the following ways:

1) We design our scheme to fulfill all the requirements that the JPEG Privacy and Security demands whereas most existing algorithms need to be modified to fulfill the requirements.
2) In JPEG coding, images are encoded in the discrete cosine transform (DCT) domain. Most existing algorithms embed the watermark in the least-significant-bit (LSB) plane of the DCT coefficients. We verify that we may extend the LSB plane to the higher bit planes based on the magnitude of the DCT coefficients and find more flexible places to embed the watermark in Sect. II-A.
3) Many existing algorithms are vulnerable to the collage attack which will be introduced in Sect. II-D. We design our watermarking scheme to be robust to this attack.

This paper is organized as follows. The proposed scheme is described in Sect. II. The experimental results showing the performance of the proposed scheme are presented in Sect. III. Finally, Sect. IV concludes the paper.

II. PROPOSED SCHEME

A. FINDING WATERMARK-EMBEDDABLE DCT COEFFICIENTS BITS

Figure 1 shows the basic encoding structure of a JPEG image compression standard. The input image is converted to a YCbCr representation. Chroma components Cb and Cr are optionally subsampled. Each of the Y, Cb, and Cr color components is subjected to an 8 × 8 block-based DCT, and the resulting DCT coefficients are rearranged into a

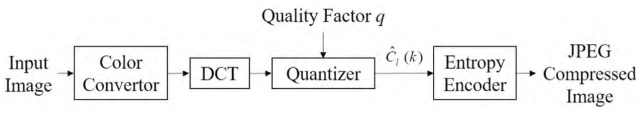


FIGURE 1. Basic encoding structure of JPEG image coding.

one-dimensional (1-D) vector by zigzag ordering [18]. Lossy compression is performed by a quantizer whose compression rate is controlled by an input parameter called the quality factor denoted as q in Fig. 1. In this paper, we denote the kth ($0 \leq k \leq 63$) quantized DCT coefficient in the lth ($0 \leq l < L$) block of the input image by $\hat{C}_l(k)$, as denoted in Fig. 1. For simplicity, herein, we use 1-D index l for representing a block located in a two-dimensional (2-D) space, where L is the total number of 8×8 blocks in the input image. The quantized DCT coefficients are losslessly compressed by an entropy encoder, and the output JPEG compressed image is constructed.

When we use the watermark-based technique for image forgery detection, the embedded watermark is required to be losslessly extracted in watermark detecting process when the watermarked image is not modified. Therefore, we need to embed the watermark into the quantized DCT coefficients $\hat{C}_l(k)$s for the case of JPEG images. Regarding the size of a watermark, embedding a larger sized watermark implies a better detection performance. However, doing so inevitably results in the worsening of the quality of the watermarked image. Therefore, a tradeoff between the size of watermark and quality of watermarked image is required to be made, and determining the locations to embed the watermark bits is a key feature for the design of a better watermark-based technique.

We perform a test to identify the optimal locations to embed the watermark bits in $\hat{C}_l(k)$s. Here, $\hat{C}_l(0)$s and $\hat{C}_l(k)$s for a non-zero k are called DC and AC coefficients, respectively, and the DC coefficients reflect the block-by-block averaged pixels of an image. Therefore, modifying the DC coefficients is known to cause blocky distortions and is therefore, avoided for watermark embedding in this study. In JPEG compression, coefficients $\hat{C}_l(k)$s are in the form of integers. Therefore, our test aims to determine the optimal positions to embed the watermark bits in the bit planes (from the least to the most significant bit plane) of AC $\hat{C}_l(k)$s. To observe the visual effect of modifying the bits of AC $\hat{C}_l(k)$s, we perform the following test:

1) Given an original raw image, obtain the JPEG image (original JPEG image) by JPEG encoding with a selected chroma subsampling type and quality factor. Convert the original JPEG image to RGB color space and calculate the peak signal-to-noise ratio (PSNR) value, denoted as PSNR$_{JPEG}$, between the original raw image and the original JPEG image. We use the PSNR defined in [19] as an image quality metric because it has been most widely used in the research on image compression and our results obtained by using other metrics such as MRSE-based SNR [19] and SSIM [20] yielded the same conclusion.

TABLE 1. Basic requirements of JPEG privacy and security [2].

Requirements	Descriptions
Support JPEG backward compliant codestreams	JPEG legacy decoders shall be able to (partially) decode codestreams that carry JPEG Privacy & Security functionality, i.e. non-protected codestream parts should be processable without causing legacy codecs to crash. Codecs equipped with JPEG Privacy & Security functionality shall be able to process the JPEG Privacy & Security related syntax as well. Hence, JPEG codestreams carrying JPEG Privacy & Security extension shall be decodable by legacy JPEG decoders even if metadata and codestream are protected.
Support encrypting metadata and image data independently	The protection mechanisms shall support independent protection of metadata and image data. This will facilitate the deployment of independent privacy policies on these distinct data components.
Support hierarchical levels of access and multiple protection levels for metadata and image protection	Since different metadata description fields and spatial or quality components of the image data might require different privacy and security policies, hierarchical metadata and codestream protection syntax shall be supported.
Privacy policies need to be evaluated and allow access to partial or complete metadata and image data	Access control should not only be to the complete image, but also for specific parts of the image and metadata. This full or partial access has to be evaluated (i.e. authorized) in order to decide if access is provided or not for a specific user, based on the existing policies.
Support common protection tools	JPEG Privacy & Security shall support a set of common protection tools: encryption, authentication, hash, certification, watermarking, digital signature and fingerprinting. Signaling of these tools will be defined by the standard.
Support box-based file format	JPEG Privacy & Security shall comply with the file format requirements as specified by JPEG Systems. Hence, this imposes the adoption of a box-based file format as specified in ISO/IEC 19566-1 [10].
Support provenance functionality	JPEG Privacy & Security shall support provenance functionality to allow tracking modifications made to an image as well as guaranteeing the integrity of an image. With this respect the JPEG Privacy & Security standard shall specify how to embed the information into an image and how to check the integrity.
Support identification of the master image	JPEG Privacy & Security shall be able to store tractable information to allow identification and assessment of the master image. Global unique identifier shall also be supported to identify derived or modified images from the master image.
Support to security features for metadata and content	JPEG Privacy & Security shall support signaling mechanisms to avoid stripping off metadata, especially IPR information.

2) Select the coefficients $\hat{C}_l(k)$s whose absolute value is greater than or equal to 2^m.
3) Flip the pth ($p \leq m$) LSB among m LSBs of all the selected coefficients and save the resulting JPEG image (flipped JPEG image). Convert the flipped JPEG image to RGB color space and calculate the PSNR value, denoted as PSNR_{F-JPEG}, between the original raw image and flipped JPEG image.
4) Compare the PSNR decrease by calculating the difference between PSNR_{JPEG} and PSNR_{F-JPEG}.

We perform our test for each of Y, Cb, and Cr color components for the quality factors of 90, 80, 70, and 60, respectively. We also calculate the PSNR differences with three chroma subsampling types: 4:4:4, 4;2:2, and 4:2:0. For our test, we selected 12 sample images including a variety of indoor and outdoor shots taken from scanners, cellular phones, digital cameras and camcorders. These are shown in Fig. 2. It is noted that our test results for other images yielded the same conclusion in terms of designing our watermarking scheme.

Tables 2–4 list our test results. We obtained the PSNR values: PSNR_{JPEG} and PSNR_{F-JPEG}, averaged them over all sample images, and obtained the difference: PSNR_{JPEG} - PSNR_{F-JPEG}, for each case of bit flipping. It is noted that all the cases on the left-bottom triangular area in Tables 2–4 are blank because p is less than or equal to m. We also have some blank columns on the right-hand side of the tables because we could not find any coefficient satisfying the corresponding condition in all sample images. We need to determine the allowable minimum value of m for the pth bit plane by increasing p and their dependency on different color components, quality factors, and chroma subsampling types. From the analysis of Tables 2–4, we may conclude the following:

1) The dependency on both the quality factors and chroma subsampling types is almost negligible.
2) The dependencies on Cb and Cr chroma components are very similar.
3) The dependency of Y component is different from that on the chroma components.

We set the value m for the pth bit plane such that it ensures a less than 0.25 dB and 0.1 dB PSNR difference for the Y and chroma components, respectively, in any of our observations for all sample images. We also compared the visual difference

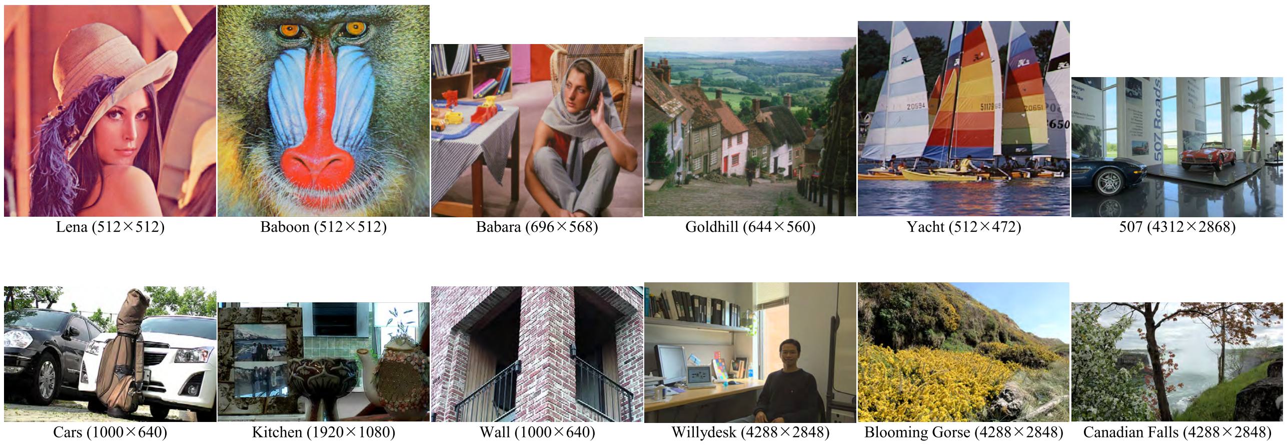


FIGURE 2. Images selected for our test to identify the optimal locations to embed the watermark bits.

TABLE 2. Decrease in $PSNR_{F-JPEG}$ compared with $PSNR_{JPEG}$ ($PSNR_{JPEG}$ - $PSNR_{F-JPEG}$)(dB) with 4:4:4 chroma subsampling.

$q=60$: $PSNR_{JPEG}=34.332$

$Y_{comp.}$	m=1	2	3	4	5	6
p=1	1.4090	0.5725	0.2099	0.0570	0.0084	0.0002
2		1.8906	0.7665	0.2214	0.0337	0.0013
3			2.3657	0.7929	0.1291	0.0047
4				2.3863	0.4705	0.0150
5					1.4585	0.0603
6						0.1334

Cb_{comp}	m=1	2	3	4	5	6
p=1	0.1620	0.0276	0.0027	0.0001		
2		0.1091	0.0107	0.0002		
3			0.0416	0.0009		
4				0.0033		
5						
6						

Cr_{comp}	m=1	2	3	4	5	6
p=1	0.1370	0.0226	0.0024	0.0002		
2		0.0893	0.0097	0.0008		
3			0.0386	0.0032		
4				0.0122		
5						
6						

$q=70$: $PSNR_{JPEG}=35.280$

$Y_{comp.}$	m=1	2	3	4	5	6
p=1	1.3827	0.5729	0.2260	0.0732	0.0151	0.0012
2		1.8983	0.8198	0.2801	0.0587	0.0048
3			2.5126	0.9890	0.2267	0.0185
4				2.8789	0.8066	0.0730
5					2.3466	0.2577
6						0.7366

Cb_{comp}	m=1	2	3	4	5	6
p=1	0.1941	0.0383	0.0047	0.0002		
2		0.1502	0.0187	0.0009		
3			0.0737	0.0036		
4				0.0135		
5						
6						

Cr_{comp}	m=1	2	3	4	5	6
p=1	0.1622	0.0316	0.0042	0.0004	0.0000	
2		0.1246	0.0165	0.0016	0.0001	
3			0.0648	0.0061	0.0002	
4				0.0237	0.0009	
5					0.0035	
6						

$q=80$: $PSNR_{JPEG}=36.583$

$Y_{comp.}$	m=1	2	3	4	5	6
p=1	1.3684	0.5801	0.2403	0.0878	0.0240	0.0037
2		1.9137	0.8699	0.3375	0.0965	0.0156
3			2.6306	1.1648	0.3623	0.0593
4				3.2710	1.2271	0.2283
5					3.3472	0.7857
6						2.2378

Cb_{comp}	m=1	2	3	4	5	6
p=1	0.2213	0.0518	0.0095	0.0009	0.0000	
2		0.2031	0.0378	0.0037	0.0001	
3			0.1478	0.0148	0.0003	
4				0.0575	0.0013	
5					0.0050	
6						

Cr_{comp}	m=1	2	3	4	5	6
p=1	0.1838	0.0437	0.0078	0.0009	0.0001	
2		0.1749	0.0312	0.0035	0.0003	
3			0.1222	0.0141	0.0013	
4				0.0549	0.0050	
5					0.0191	
6						

$q=90$: $PSNR_{JPEG}=38.92$

$Y_{comp.}$	m=1	2	3	4	5	6
p=1	1.3237	0.5657	0.2439	0.1019	0.0375	0.0111
2		1.8631	0.8820	0.3862	0.1470	0.0430
3			2.6647	1.3153	0.5441	0.1680
4				3.6053	1.7396	0.6101
5					4.4141	1.9012
6						4.6645

Cb_{comp}	m=1	2	3	4	5	6
p=1	0.2997	0.0812	0.0200	0.0033	0.0003	
2		0.3178	0.0796	0.0134	0.0010	
3			0.3069	0.0531	0.0039	
4				0.2058	0.0162	
5					0.0631	
6						

Cr_{comp}	m=1	2	3	4	5	6
p=1	0.2368	0.0679	0.0171	0.0028	0.0003	0.0000
2		0.2660	0.0686	0.0112	0.0014	0.0001
3			0.2654	0.0440	0.0055	0.0004
4				0.1704	0.0213	0.0018
5					0.0835	0.0069
6						0.0240

between the original and flipped JPEG image and concluded that we cannot detect any visual difference in the eight times enlarged versions of images being compared with this PSNR difference. We finally decided our watermark embeddable bits in the bit plane of AC coefficients as denoted in Table 5 and its usages are described in Appendix B and C.

B. PROPOSED WATERMARK EMBEDDING SCHEME

In addition to benign modifications on image data such as enlarging, reducing, resizing, cropping, recompressing, sharpening, enhancing, and deblurring, we design the proposed scheme to be able to detect malicious modifications including copy-move, region duplication, image splicing, image compositing, and block exchanging [3], [4], [11] along with the modification of metadata in a JPEG image file. Figure 3 shows the proposed watermark embedding scheme. The input JPEG image is first decoded by the entropy decoder, and the quantized DCT coefficients $\hat{C}_l(k)$s are extracted. For each 8×8 DCT block, the watermark-embeddable bits are determined as described in Sect. II-A.

The entire set of 8×8 DCT blocks is partitioned into a tree-structured group of blocks (GOB) such that each GOB has at least T_b watermark-embeddable bits. The T_b is a thresholding value given by the user to determine the size of the GOBs. We embed the watermark for each GOB. (We will localize a modified region by indicating the GOBs that include the

TABLE 3. Decrease in PSNR$_{F-JPEG}$ compared with PSNR$_{JPEG}$ (PSNR$_{JPEG}$- PSNR$_{F-JPEG}$)(dB) with 4:2:2 chroma subsampling.

q=60: PSNR$_{JPEG}$=33.557

Y$_{comp.}$	m=1	2	3	4	5	6
p=1	1.2205	0.4890	0.1774	0.0477	0.0070	0.0002
2		1.6508	0.6554	0.1859	0.0280	0.0012
3			2.0787	0.6768	0.1082	0.0042
4				2.1005	0.3984	0.0139
5					1.2661	0.0571
6						0.1271

Cb$_{comp}$	m=1	2	3	4	5	6
p=1	0.1724	0.0306	0.0036	0.0002		
2		0.1213	0.0142	0.0006		
3			0.0556	0.0023		
4				0.0085		
5						
6						

Cr$_{comp}$	m=1	2	3	4	5	6
p=1	0.1406	0.0230	0.0028	0.0003		
2		0.0921	0.0110	0.0013		
3			0.0433	0.0051		
4				0.0192		
5						
6						

q=70: PSNR$_{JPEG}$=34.394

Y$_{comp.}$	m=1	2	3	4	5	6
p=1	1.1702	0.4751	0.1850	0.0593	0.0121	0.0009
2		1.6218	0.6857	0.2307	0.0472	0.0038
3			2.1720	0.8312	0.1855	0.0151
4				2.5010	0.6701	0.0593
5					2.0228	0.2119
6						0.6159

Cb$_{comp}$	m=1	2	3	4	5	6
p=1	0.1931	0.0412	0.0061	0.0004		
2		0.1628	0.0250	0.0017		
3			0.0983	0.0065		
4				0.0248		
5						
6						

Cr$_{comp}$	m=1	2	3	4	5	6
p=1	0.1626	0.0316	0.0042	0.0005	0.0000	
2		0.1260	0.0168	0.0021	0.0001	
3			0.0657	0.0081	0.0004	
4				0.0323	0.0015	
5					0.0053	
6						

q=80: PSNR$_{JPEG}$=35.523

Y$_{comp.}$	m=1	2	3	4	5	6
p=1	1.1187	0.4630	0.1892	0.0686	0.0190	0.0028
2		1.5819	0.7002	0.2666	0.0753	0.0119
3			2.2118	0.9481	0.2865	0.0462
4				2.7844	0.9981	0.1787
5					2.8541	0.6308
6						1.8699

Cb$_{comp}$	m=1	2	3	4	5	6
p=1	0.2110	0.0522	0.0100	0.0012	0.0001	
2		0.2039	0.0397	0.0048	0.0002	
3			0.1541	0.0191	0.0007	
4				0.0745	0.0030	
5					0.0116	
6						

Cr$_{comp}$	m=1	2	3	4	5	6
p=1	0.1791	0.0421	0.0073	0.0009	0.0001	
2		0.1674	0.0300	0.0039	0.0006	
3			0.1179	0.0152	0.0024	
4				0.0596	0.0090	
5					0.0346	
6						

q=90: PSNR$_{JPEG}$=37.512

Y$_{comp.}$	m=1	2	3	4	5	6
p=1	1.0117	0.4221	0.1799	0.0744	0.0274	0.0081
2		1.4502	0.6671	0.2865	0.1079	0.0312
3			2.1245	1.0139	0.4068	0.1233
4				2.9392	1.3599	0.4577
5					3.6601	1.4965
6						3.8928

Cb$_{comp}$	m=1	2	3	4	5	6
p=1	0.2606	0.0718	0.0194	0.0034	0.0004	0.0000
2		0.2799	0.0755	0.0137	0.0015	0.0000
3			0.2907	0.0539	0.0061	0.0002
4				0.2081	0.0244	0.0008
5					0.0960	0.0034
6						0.0128

Cr$_{comp}$	m=1	2	3	4	5	6
p=1	0.2228	0.0610	0.0153	0.0025	0.0003	0.0000
2		0.2383	0.0610	0.0101	0.0014	0.0001
3			0.2373	0.0406	0.0054	0.0005
4				0.1586	0.0212	0.0019
5					0.0827	0.0079
6						0.0264

TABLE 4. Decrease in PSNR$_{F-JPEG}$ compared with PSNR$_{JPEG}$ (PSNR$_{JPEG}$- PSNR$_{F-JPEG}$)(dB) with 4:2: chroma subsampling.

q=60: PSNR$_{JPEG}$=32991

Y$_{comp.}$	m=1	2	3	4	5	6
p=1	1.0909	0.4306	0.1548	0.0414	0.0061	0.0003
2		1.4834	0.5811	0.1634	0.0243	0.0011
3			1.8796	0.6022	0.0953	0.0045
4				1.9008	0.3506	0.0141
5					1.1328	0.0570
6						0.1274

Cb$_{comp}$	m=1	2	3	4	5	6
p=1	0.1742	0.0318	0.0040	0.0002		
2		0.1253	0.0152	0.0005		
3			0.0606	0.0019		
4				0.0071		
5						
6						

Cr$_{comp}$	m=1	2	3	4	5	6
p=1	0.1366	0.0222	0.0030	0.0003		
2		0.0866	0.0105	0.0010		
3			0.0425	0.0043		
4				0.0157		
5						
6						

q=70: PSNR$_{JPEG}$=33.741

Y$_{comp.}$	m=1	2	3	4	5	6
p=1	1.0296	0.4126	0.1593	0.0511	0.0101	0.0007
2		1.4348	0.5977	0.1994	0.0407	0.0034
3			1.9370	0.7267	0.1608	0.0141
4				2.2431	0.5837	0.0547
5					1.8003	0.1955
6						0.5801

Cb$_{comp}$	m=1	2	3	4	5	6
p=1	0.1950	0.0416	0.0069	0.0005		
2		0.1613	0.0262	0.0018		
3			0.1050	0.0072		
4				0.0272		
5						
6						

Cr$_{comp}$	m=1	2	3	4	5	6
p=1	0.1575	0.0288	0.0041	0.0005	0.0001	
2		0.1159	0.0166	0.0020	0.0000	
3			0.0641	0.0079	0.0003	
4				0.0308	0.0011	
5					0.0045	
6						

q=80: PSNR$_{JPEG}$=34.744

Y$_{comp.}$	m=1	2	3	4	5	6
p=1	0.9604	0.3912	0.1584	0.0568	0.0156	0.0023
2		1.3690	0.5961	0.2244	0.0631	0.0099
3			1.9332	0.8091	0.2392	0.0382
4				2.4532	0.8522	0.1485
5					2.5180	0.5296
6						1.6209

Cb$_{comp}$	m=1	2	3	4	5	6
p=1	0.2031	0.0506	0.0100	0.0014	0.0000	
2		0.1994	0.0398	0.0050	0.0001	
3			0.1545	0.0198	0.0004	
4				0.0777	0.0016	
5					0.0064	
6						

Cr$_{comp}$	m=1	2	3	4	5	6
p=1	0.1700	0.0389	0.0072	0.0010	0.0001	
2		0.1537	0.0276	0.0036	0.0004	
3			0.1069	0.0149	0.0015	
4				0.0580	0.0064	
5					0.0230	
6						

q=90: PSNR$_{JPEG}$=36.513

Y$_{comp.}$	m=1	2	3	4	5	6
p=1	0.8286	0.3400	0.1436	0.0590	0.0215	0.0063
2		1.1990	0.5391	0.2283	0.0849	0.0249
3			1.7819	0.8269	0.3251	0.0972
4				2.5049	1.1201	0.3656
5					3.1554	1.2347
6						3.3744

Cb$_{comp}$	m=1	2	3	4	5	6
p=1	0.2351	0.0659	0.0175	0.0031	0.0003	0.0000
2		0.2587	0.0706	0.0127	0.0015	0.0000
3			0.2738	0.0503	0.0061	0.0001
4				0.1971	0.0242	0.0002
5					0.0938	0.0009
6						0.0028

Cr$_{comp}$	m=1	2	3	4	5	6
p=1	0.2026	0.0553	0.0136	0.0021	0.0002	0.0000
2		0.2171	0.0539	0.0088	0.0011	0.0001
3			0.2095	0.0345	0.0042	0.0005
4				0.1356	0.0170	0.0020
5					0.0657	0.0072
6						0.0254

modified region in the watermark detecting process described in Sect. II-C.) Parameter T_b determines the size of both the GOB and watermark embedded in a GOB. A large T_b results in large-sized GOBs so that the localization of the modified

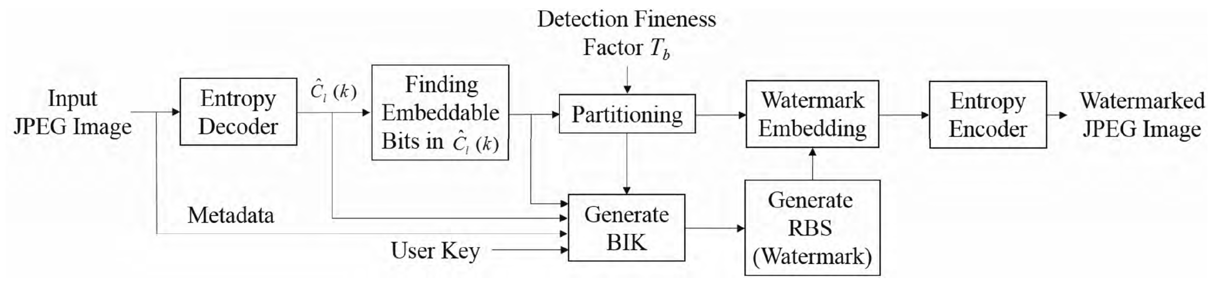


FIGURE 3. Proposed watermark embedding scheme.

TABLE 5. Watermark embeddable bits in the bit plane of AC DCT coefficients represented based on Tables 2-4.

Y component		m					
		1	2	3	4	5	6
p	1			○	○	○	○
	2					○	○
	3						○
	4						
	5						
	6						

Cb and Cr component		m					
		1	2	3	4	5	6
p	1			○	○	○	○
	2				○	○	○
	3					○	○
	4						○
	5						
	6						

region is accomplished coarsely. In contrast, a large-sized GOBs also embeds a large-sized watermark in a GOB, so that we may be able to reduce the false decision rate. Therefore, this value is a tradeoff between the security and localization fineness. Our partitioning process is briefly described as follows and the detailed description is preferably referred to Appendix A.

1) Set the entire set of blocks be a mother (root) GOB.
2) For each mother GOB, select a horizontal or vertical direction for which the length of the GOB is larger. Divide this GOB into two equally spaced child GOBs in the selected direction. (If the length of the GOB in the selected direction is odd, division is performed so that the length of the left or upper child GOB is smaller than the corresponding right or lower child GOB by one for the horizontal or vertical direction, respectively.) If both the child GOBs have more than T_b watermark-embeddable bits, they are declared as two mother GOBs. Otherwise, the mother GOB is declared as a child GOB and the branch chain from the root GOB to this GOB is saved.
3) Repeat step 2) until there is no mother GOB.

In step 2), the branch chain is a sequence of bits indicating how a child GOB has been divided from the root. If a mother GOB is divided horizontally and a current GOB is an upper (lower) child GOB of the mother GOB, 0 (1) is concatenated to the mother GOB's branch chain to generate the child GOB's branch chain. Similarly, if a mother GOB is divided vertically, 0 (1) is concatenated for a left (right) child GOB.

When the partitioning is complete, we generate an intrinsic watermark for each child GOB. We generate a random bit sequence (RBS) whose size is same as the number of watermark-embeddable bits in the GOB as the watermark embedded in the GOB. We will perform the image integrity authentication by checking the coincidence of the embedded watermark. Therefore, a different RBS is required to be generated for a modified GOB. For each GOB, we concatenate the branch chain from the root to the current GOB and all the bits of DCT coefficients not decided as the watermark-embeddable bits in the GOB. We hash the result by a cryptographically secure hashing function and generate the so-called block identifying key (BIK) by encrypting the hash with a user key and the metadata of the input JPEG image. The RBS is generated using this BIK and embedded as the watermark in the watermark-embeddable bits. The pseudocode of embedding process is listed in Appendix B. After performing watermark embedding for all GOBs, we perform the entropy encoding with watermarked DCT coefficients and obtain the watermarked JPEG image.

C. PROPOSED WATERMARK DETECTION SCHEME

The watermark detecting process is simple, as shown in Fig. 4. The input JPEG image being examined is entropy decoded. Obtaining of the embeddable DCT coefficient bits, partitioning, and generating RBS are achieved similar to the watermark embedding process described in Sect. II-B. For each GOB, the data in the embeddable DCT coefficient bits are extracted and compared with the generated RBS. If they are different, this GOB is declared as a modified region. The pseudocode of detecting process is listed in Appendix C.

When we generate the RBS in the watermark embedding process, we use four input parameters: the branch chain from the root to the current GOB, all the bits of DCT coefficients not decided as the watermark-embeddable bits in the GOB, the metadata, and the user key. Therefore, we may detect any modifications, including both benign and malicious modifications as previously mentioned in Sect. II-B, changing one of these four parameters.

It is noted that even though the watermark-based image forgery detection techniques aim at 100% detection accuracy, they cannot ensure a 0% false negative detection rate owing to the use of a finite-size watermark. In our scheme, the data in the embeddable DCT coefficient bits is compared with the generated RBS for authenticating each GOB. When the number of extracted embeddable DCT coefficient bits in a GOB (the size of watermark assumed to be embedded for this GOB

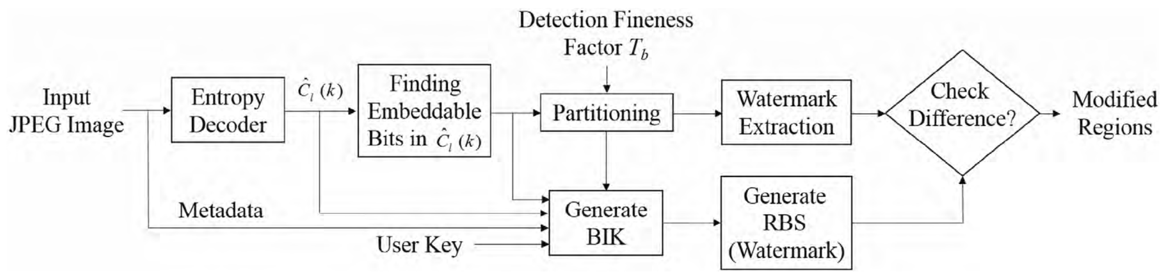


FIGURE 4. **Proposed watermark detection scheme.**

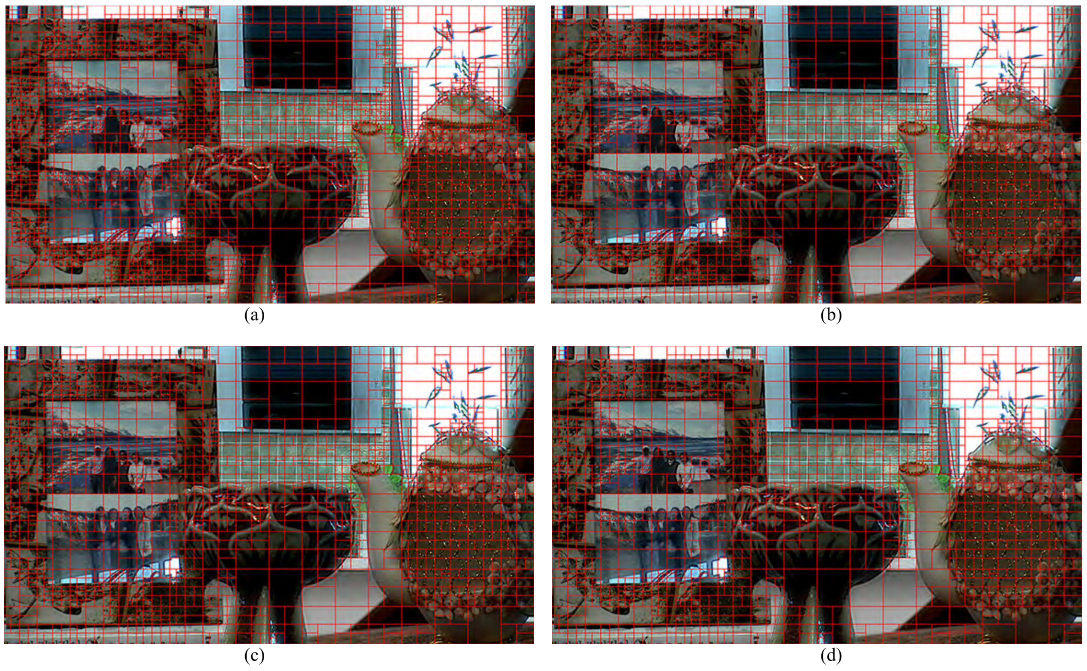


FIGURE 5. **Example of block partitioning for a sample image of Kitchen: (a) $T_b = 10$, (b) $T_b = 15$, (c) $T_b = 18$, and (d) $T_b = 20$.**

in the embedding process) is N_{GOB} (this value is ensured to be greater than T_b by our partitioning process), the probability of the extracted bits coinciding with the generated RBS by accident is $2^{-N_{GOB}}$, which is almost negligible for large a sufficiently large N_{GOB} but still not zero.

D. IMPROVEMENT FOR COUNTERMEASURE AGAINST COLLAGE ATTACK

Collage attack is a well-known attack that is a collection of malicious attacks specially designed for weakening the watermark detecting mechanism in the localized forged regions [5], [7], [21]. This attack, which was first introduced by Holliman and Memon [22], considers the case in which the watermark embedded in each block depends only on the content of the block. This is referred to as block-wise independence. There may be an attacker who fully understands the watermarking mechanism and has a database of the images already authenticated by the same user key. When the watermarking mechanism is block-wise independent,

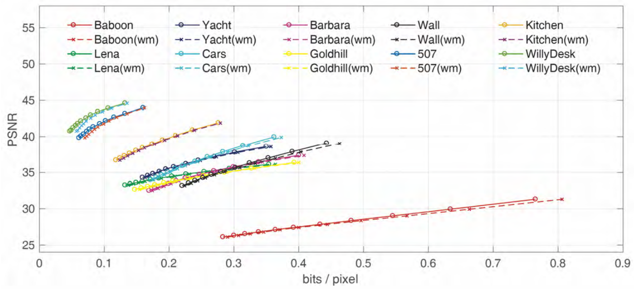


FIGURE 6. **Comparison of the PSNR performances of the original and watermarked JPEG images for the sample images. '(wm)' denotes the watermarked JPEG images.**

a malicious attacker may locate an already authenticated block with the same watermark from this database and perform the replacement attack on the block under detection. It has been known that most block-wise independent watermarking schemes are vulnerable to this type of collage attack.

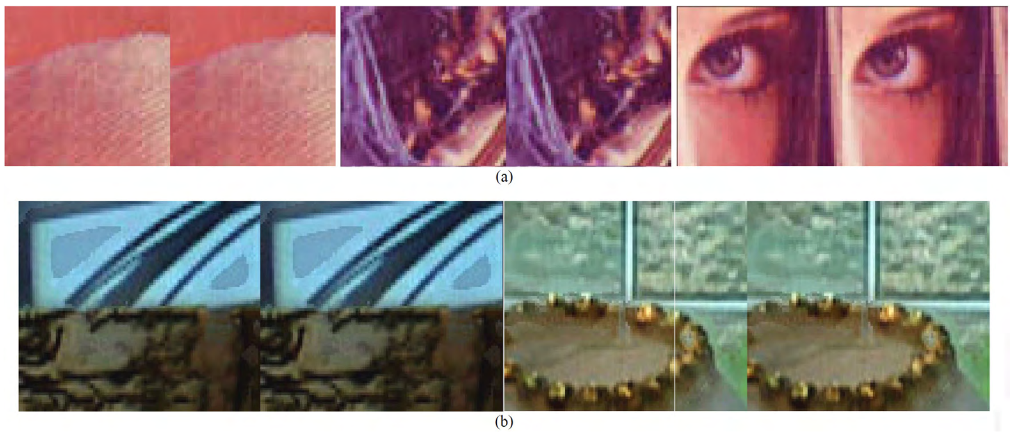


FIGURE 7. Visual comparison of the original and watermarked JPEG images for the sample images of (a) Lena and (b) Kitchen. Left is the selected part of the encoded JPEG image at quality factor 60. The right is the watermarked JPEG image of that part.

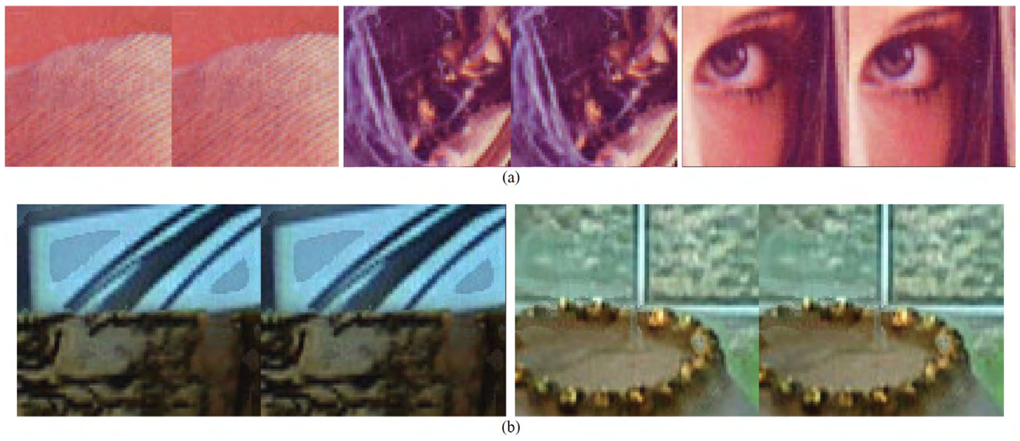


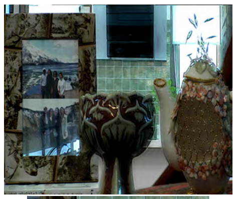
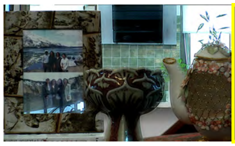
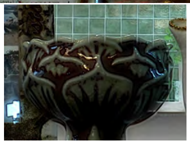
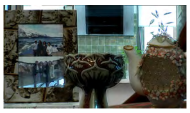
FIGURE 8. Visual comparison of the original and watermarked JPEG images for the sample images of (a) Lena and (b) Kitchen. Left is the selected part of the encoded JPEG image at quality factor 90. The right is the watermarked JPEG image of that part.

In our watermark embedding scheme described in Sect. II-B, we embedded the watermark into GOB independently. Although the size of GOBs is variable, it is still possible that a collage attacker may find an already authenticated GOB with the same watermark from the database and perform the GOB replacement attack. We avoid this attack by chaining the connected GOBs. In the proposed scheme discussed in Sect. II-B, we concatenated the branch chain from the root to the current GOB and all the bits of DCT coefficients not decided as the watermark-embeddable bits in the GOB for generating the BIK. Therefore, the generated BIK is block-wise independent. We modify this BIK generating procedure to be block-wise dependent. When we watermark a GOB, we also select all the neighboring GOBs connected to this GOB, and additionally concatenate all the branch chains from the root to each neighboring GOB and all the bits of DCT coefficients not decided as the watermark-embeddable bits in all those neighboring GOBs. This improved procedure results in an RBS generation for a GOB that is always dependent on the neighboring GOBs. When a collage attack occurs, the watermark of the attacking GOB that was generated using the neighboring GOBs of the attacking GOB cannot be same as the RBS generated using the GOBs that are different from the neighboring GOBs of the attacked GOB. Therefore, in our improved scheme, when a GOB is forged by a collage attack, the attacked GOB and the neighboring GOBs connected to the attacked GOB are detected as a modified region.

III. EXPERIMENTAL RESULTS

A. EXPERIMENTAL SETUP

In the proposed scheme, the parameter T_b for the block partitioning is selected by a tradeoff between the security and localization fineness. We display several examples of the block partitioning with different values of T_b in Fig. 5 for a sample image of Kitchen. We have chosen the value to be 15 for the experiment presented in this paper. In this case, the maximum false positive decision rate for a GOB is $2^{-15} = 0.0000305$, which is almost negligible. For the hashing function, we use the SHA-512 algorithm whose output length is 512 bits because its maximum input message size is $(2^{128}-1)$ bits, which suffices for our usage.

Modification type	Modified image	Detection result	Modification type	Modified image	Detection result
Original watermarked JPEG image					
Metadata			Resizing		
Enlarging			Cropping		
Reducing			Re- Compressing, sharpening, enhancing, deblurring		

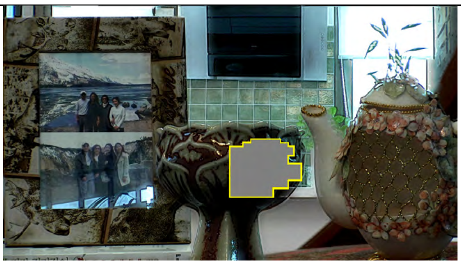
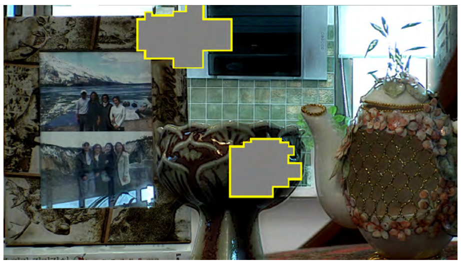
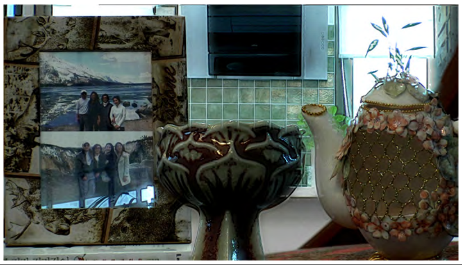
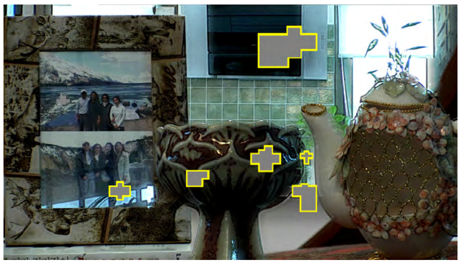
Modification type	Modified image	Modified image
Copy-move, region duplication, image splicing		
Block exchanging		
Collage attack		

FIGURE 9. **Examples of detection results.**

B. QUALITY COMPARISON OF WATERMARKED AND ORIGINAL IMAGE

For an objective test to compare the image qualities of the original and watermarked JPEG images, we calculated their PSNR values relative to the original raw image. The PSNR curves obtained from the sample images in Fig. 2 are displayed in Fig. 6. As can be seen, the PSNR difference is almost negligible. The PSNR metric used in this study supports color images, and its definition can be found in [19].

TABLE 6. Execution times of proposed watermark embedding and detecting scheme (in seconds).

Sample Image	Image size	Embedding					Detecting				
		q=60	70	80	90	Avg.	q=60	70	80	90	Avg.
Yacht	512×472	0.259	0.274	0.336	0.393	0.316	0.242	0.272	0.325	0.390	0.307
Lena	512×512	0.284	0.293	0.298	0.384	0.320	0.280	0.283	0.299	0.362	0.309
Baboon	512×512	0.298	0.326	0.398	0.705	0.432	0.290	0.311	0.379	0.685	0.416
Goldhill	664×560	0.350	0.401	0.434	0.536	0.430	0.357	0.378	0.447	0.546	0.435
Barbara	696×568	0.401	0.469	0.530	0.794	0.549	0.414	0.474	0.513	0.681	0.521
Cars	1000×640	0.789	0.882	1.047	1.903	1.155	0.825	0.927	1.032	1.821	1.151
Wall	1000×640	0.963	1.091	1.485	2.769	1.577	0.955	1.163	1.532	3.162	1.703
Kitchen	1920×1080	2.607	2.934	2.981	4.810	3.333	2.566	2.862	2.921	4.802	3.288
Willydesk	4288×2848	16.517	16.485	17.451	20.608	17.765	16.53	16.464	17.535	20.636	17.791
507	4312×2868	15.549	16.085	17.224	24.521	18.345	15.322	16.047	18.931	22.353	18.163

For a visual comparison of the watermarked and original JPEG images, encoded with quality factors 60 and 90, portions of a pair of original and watermarked JPEG images including various types of regions such as textured regions, homogeneous regions, edge regions, and adjacent regions were selected and eight times enlarged. These are presented in Figs. 7 and 8. The visual comparison shows that the visual differences between the original and watermarked JPEG images are indiscernible. Note that our results for the other images yielded the same conclusions in terms of visual comparison.

C. EXPERIMENTAL RESULTS AGAINST VARIOUS FORGERIES

Figure 9 shows examples of the detection results against several modifications. The detected regions are denoted by grey color. Metadata modification, geometric modifications: enlarging; reducing; resizing; cropping; and affine transforming; and image processing modifications: recompressing; sharpening; enhancing; and deblurring, inevitably change the RBS for all GOBs. This causes the entire area of the image under detection to be declared as a modified region. In contrast, modifications including copy–move; region duplication; image splicing; block exchanging; doodling; and collage attack, change only the GOBs including the modified region. In this case, all the GOBs connected to the modified region are declared as a modified region marked with the yellow polygon in Fig. 9.

D. COMPUTATIONAL COMPLEXITY

The execution times of our watermark embedding and detecting scheme were calculated on a platform powered by a quad-core 3.6-GHz CPU and 20-GB RAM. Table 6 shows the execution times for our sample JPEG images encoded with several values of quality factor q. The listed execution times of detecting scheme were calculated for the case of unmodified input images. In our both embedding and detecting scheme, the most time-consuming process is the partitioning process whose computing time is highly dependent on the number of produced GOBs in the input image. It is observed that the embedding and the detecting time for the same input image are almost same because their partitioning process produces the same number of GOBs when the detecting image is not modified. We may also observe that the computing time increases as the size of the input image, the quality factor, and the complexity of the input image increases, respectively, because these three factors increase the number of GOBs.

IV. CONCLUSION

Herein, we describe a new watermark-based scheme for the authentication of JPEG image integrity. Because the proposed scheme performs watermarking by modifying the DCT coefficients, it can test the integrity without any additional signature information, and it is possible to achieve high detection accuracy for the forgeries in the spatial domain. Moreover, the proposed method is a technique that can control the detection fineness. The proposed partitioning method divides the host image into groups called GOBs containing multiple DCT blocks to generate watermarks containing the information of the neighboring GOBs, which is a technique that detects the collage attack. Thus, the proposed method enables the control of the visual quality while providing high detection accuracy for typical JPEG images modified in the DCT and spatial domains.

We evaluated the performance of our method against typical forgeries. We also performed objective tests to evaluate the quality of the watermarked images. For the objective tests, we compared the results using the PSNR metric. Our experiments showed a high detection accuracy against various forgeries and a negligible difference both objectively and visually between the original and watermarked JPEG images.

APPENDIX

This appendix introduces the pseudo codes for the proposed GOB partitioning, watermark embedding, and watermark detecting.

A. GOB PARTITIONING

Pseudocode 1 Partitioning (MotherGOB, Chain, T_b)

```
Input: Mother GOB, branch chain, and $T_b$ value
Output: Child GOBs and branch chain for each child GOB
1:   Chain1 $\leftarrow$ Chain
2:   Chain2 $\leftarrow$ Chain
```

```
3: if (width $\geq$ height then)
4:     ChildGOB1 $\leftarrow$ Left half of the MotherGOB
5:     ChildGOB2 $\leftarrow$ Right half of the MotherGOB
6: else
7:     ChildGOB1 $\leftarrow$ Top half of the MotherGOB
8:     ChildGOB2 $\leftarrow$ Bottom half of the MotherGOB
9: Nbits1 $\leftarrow$ Number of embeddable bits in ChildGOB1
10: Nbits2 $\leftarrow$ Number of embeddable bits in ChildGOB2
11: if (Nbits1 $\geq T_b$) and (Nbits2 $\geq T_b$) then
12:     Concatenate binary 0 to Chain1
13:     Concatenate binary 1 to Chain2
14:     Partitioning(ChildGOB1, Chain1, $T_b$)
15:     Partitioning(ChildGOB2, Chain2, $T_b$)
16: else
17:     ChildGOB $\leftarrow$ MotherGOB
18:     Save ChildGOB and Chain
19: Return
```

B. WATERMARK EMBEDDING

Pseudocode 2 Embedding (arrayGOB, arrayRBS)

Input: Array of a GOB (composed of Y, Cb, and Cr components) and its corresponding RBS array (generated by the method described in Sect. II-D)

Output: Array of the GOB watermarked with the RBS

```
1: EmbedYcomp(arrayGOB[Y], arrayRBS[Y])
2: EmbedCbCrcomp(arrayGOB[Cb], arrayRBS[Cb])
3: EmbedCbCrcomp(arrayGOB[Cr], arrayRBS[Cr])
where
procedure EmbedYcomp(GOB, RBS)
1: for each block $b$ in GOB do
2:     for each AC coefficient $coef$ in $b$ do
3:     $c \leftarrow 0$
4:     flag $\leftarrow$ 0 // flag for the sign of coefficient
5:     if $coef < 0$ then
6:         $c \leftarrow$ abs($coef$)
7:         flag $\leftarrow$ 1 // current coefficient is negative
8:     else $c \leftarrow coef$
   // Replace the watermark-embeddable bit
9:     $idx \leftarrow 1$ // bit index in RBS
10:        if $c \geq 2^3$ then
11:          $c$:1st LSB $\leftarrow$ RBS:$idx$ // $idx$th bit in RBS
12:          $idx \leftarrow idx + 1$ // move bit index to the next bit
13:        if $c \geq 2^5$ then
14:          $c$:2nd LSB $\leftarrow$ RBS:$idx$
15:          $idx \leftarrow idx + 1$ // move bit index to the next bit
16:    if $c \geq 2^6$ then
17:          $c$:3rd LSB $\leftarrow$ RBS:$idx$
18:          $idx \leftarrow idx + 1$ // move bit index to the next bit
19:      if flag = 1 then
20:          $coef \leftarrow -c$
21:        else
22:          $coef \leftarrow c$
procedure EmbedCbCrcomp (GOB, RBS)
1:    for each block $b$ in GOB do
2:        for each AC coefficient $coef$ in $b$ do
3:        $c \leftarrow 0$
4:        flag $\leftarrow$ 0 // flag for the sign of coefficient
6:        if $coef < 0$ then
7:          $c \leftarrow$ abs($coef$)
8:          flag $\leftarrow$ 1 // current coefficient is negative
9:        else $c \leftarrow coef$
         // Replace the watermark-embeddable bit
10:         $idx \leftarrow 1$ // bit index in RBS
11:         if $c \geq 2^3$ then
12:           $c$:1st LSB $\leftarrow$ RBS:$idx$ // $idx$th bit in RBS
13:           $idx \leftarrow idx + 1$ // move bit index to the next bit
14:         if $c \geq 2^4$ then
15:           $c$:2nd LSB $\leftarrow$ RBS:$idx$
16:           $idx \leftarrow idx + 1$ // move bit index to the next bit
17:         if $c \geq 2^5$ then
18:         $c$:3rd LSB $\leftarrow$ RBS:$idx$
19:           $idx \leftarrow idx + 1$ // move bit index to the next bit
20:         if $c \geq 2^6$ then
21:           $c$:4th LSB $\leftarrow$ RBS:$idx$
22:           $idx \leftarrow idx + 1$ // move bit index to the next bit
23:         if flag = 1 then
24:           $coef \leftarrow -c$//go back to negative value
25:         else
26:           $coef \leftarrow c$
```

C. WATERMARK DETECTING

Pseudocode 3 Detecting (arrayGOB, arrayRBS)

Input: Array of a GOB (composed of Y, Cb, and Cr components) and its corresponding RBS array (generated by the method described in Sect. II-D)

Output: Detection result

```
1: DetectYcomp(arrayGOB[Y],arrayRBS[Y]
2: DetectCbCrcomp(arrayGOB[Cb],arrayRBS[Cb])
3: DetectCbCrcomp(arrayGOB[Cr], arrayRBS[Cr])
where
procedure DetectYcomp(GOB, RBS)
1: for each block $b$ in GOB do
2:     for each AC coefficient $coef$ in $b$ do
3:     $c \leftarrow 0$
4:     if $coef < 0$ then
5:         $c \leftarrow$ abs($coef$)
6:     else $c \leftarrow coef$
       // Extract a watermark
7:     $idx \leftarrow 1$ // bit index in Watermark
8:     if $c \geq 2^3$ then
9:         Watermark:$idx \leftarrow c$:1st LSB
10:        $idx \leftarrow idx + 1$ // move bit index to the next bit
11:      if $c \geq 2^5$ then
12:        Watermark:$idx \leftarrow c$:2nd LSB
13:        $idx \leftarrow idx + 1$ // move bit index to the next bit
14:      if $c \geq 2^6$ then
15:        Watermark:$idx \leftarrow c$:3rd LSB
16:        $idx \leftarrow idx + 1$ // move bit index to the next bit
17: if RBS = Watermark then
18:     return no forgery
19: else
20:     return forgery detected
```

```
procedure DetectCbCrcomp(GOB, RBS)
1: for each block $b$ in GOB do
2:   for each AC coefficient $coef$ in $b$ do
3:     $c \leftarrow 0$
4:     if $coef < 0$ then
5:       $c \leftarrow$ abs($coef$)
6:     else $c \leftarrow coef$
       // Extract a watermark
7:       $idx \leftarrow 1$ // bit index in Watermark
8:       if $c \geq 2^3$ then
9:         Watermark:$idx \leftarrow c$:1st LSB
11:          $idx \leftarrow idx + 1$ // move bit index to the next bit
12:        if $c \geq 2^4$ then
13:          Watermark:$idx \leftarrow c$:2nd LSB
14:          $idx \leftarrow idx + 1$ // move bit index to the next bit
15:        if $c \geq 2^5$ then
16:          Watermark:$idx \leftarrow c$:3rd LSB
17:          $idx \leftarrow idx + 1$ // move bit index to the next bit
18:        if $c \geq 2^6$ then
19:          Watermark:$idx \leftarrow c$:4th LSB
20:          $idx \leftarrow idx + 1$ // move bit index to the next bit
21: if RBS = Watermark then
22:   return no forgery
23: else
24:   return forgery detected
```

REFERENCES

[1] M. Meeker, "Internet trends 2016—Code conference," Kleiner Perkins, Menlo Park, CA, USA, Jun. 2016.

[2] (2017). *JPEG Privacy and Security Final Call for Proposals*. [Online]. Available: https://jpeg.org/items/20170403_cfp_privacy_security.html

[3] G. K. Birajdar and V. H. Mankar, "Digital image forgery detection using passive techniques: A survey," *Digit. Invest.*, vol. 10, no. 3, pp. 226–245, Oct. 2013.

[4] H. Farid, "Image forgery detection," *IEEE Signal Process. Mag.*, vol. 26, no. 2, pp. 16–25, Mar. 2009.

[5] O. M. Al-Qershi and B. E. Khoo, "Passive detection of copy-move forgery in digital images: State-of-the-art," *Forensic Sci. Int.*, vol. 231, no. 1, pp. 284–295, 2013.

[6] C.-S. Lu and H.-Y. M. Liao, "Structural digital signature for image authentication: An incidental distortion resistant scheme," *IEEE Trans. Multimedia*, vol. 5, no. 2, pp. 161–173, Jun. 2003.

[7] F. Y. Shih, *Digital Watermarking and Steganography: Fundamentals and Techniques*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2017.

[8] W.-C. Hu, W.-H. Chen, D.-Y. Huang, and C.-Y. Yang, "Effective image forgery detection of tampered foreground or background image based on image watermarking and alpha mattes," *Multimedia Tools Appl.*, vol. 75, no. 6, pp. 3495–3516, Mar. 2016.

[9] C.-C. Lo and Y.-C. Hu, "A novel reversible image authentication scheme for digital images," *Signal Process.*, vol. 98, pp. 174–185, May 2014.

[10] *Information Technology: Scalable Compression and Coding of Continuous-Tone Still Images JPEG Systems Part 1: Packaging of Information Using Codestream and File Formats*, document ISO/IEC 19566-1:2015, ISO/IEC, 2015.

[11] M. A. Qureshi and M. Deriche, "A bibliography of pixel-based blind image forgery detection techniques," *Signal Process., Image Commun.*, vol. 39, pp. 46–74, Nov. 2015.

[12] D. Caragata, S. El Assad, and M. Luduena, "An improved fragile watermarking algorithm for JPEG images," *AEU-Int. J. Electron. Commun.*, vol. 69, no. 12, pp. 1783–1794, 2015.

[13] H. Wang, K. Ding, and C. Liao, "Chaotic watermarking scheme for authentication of JPEG Images," in *Proc. Int. Symp. IEEE Biometrics Secur. Technol. (ISBAST)*, Apr. 2008, pp. 1–4.

[14] V. G. Edupuganti and F. Y. Shih, "Authentication of JPEG images based on genetic algorithms," in *Multimedia Security: Watermarking, Steganography, and Forensics*. Boca Raton, FL, USA: CRC Press, 2017, p. 165.

[15] J. Fridrich, "Image watermarking for tamper detection," in *Proc. Int. Conf. Image Process. (ICIP)*, vol. 2, Oct. 1998, pp. 404–408.

[16] C. T. Li, "Digital fragile watermarking scheme for authentication of JPEG images," *IEE Proc.-Vis., Image Signal Process.*, vol. 151, no. 6, pp. 460–466, Dec. 2004.

[17] M. A. Suhail and M. S. Obaidat, "Digital watermarking-based DCT and JPEG model," *IEEE Trans. Instrum. Meas.*, vol. 52, no. 5, pp. 1640–1647, Oct. 2003.

[18] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*. New York, NY, USA: Springer, 1992.

[19] S. Choi, O.-J. Kwon, J. Lee, and Y. Kim, "A JPEG backward-compatible image coding scheme for high dynamic range images," *Digit. Signal Process.*, vol. 67, pp. 1–16, Aug. 2017.

[20] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[21] I. J. Cox, M. L. Miller, J. A. Bloom, J. Fridrich, and T. Kalker, *Digital Watermarking and Steganography*. San Mateo, CA, USA: Morgan Kaufmann, 2007.

[22] M. Holliman and A. N. Memon, "Counterfeiting attacks on oblivious block-wise independent invisible watermarking schemes," *IEEE Trans. Image Process.*, vol. 9, no. 3, pp. 432–441, Mar. 2000.

OH-JIN KWON received the M.S. degree in electrical engineering from the University of Southern California, Los Angeles, in 1991, and the Ph.D. degree in electrical engineering from the University of Maryland at College Park, College Park, in 1994. He was a Researcher with the Agency for Defense Development of Korea from 1984 to 1989 and the Head of the Media Lab, Samsung SDS Co., Ltd., Seoul, South Korea, from 1995 to 1999. Since 1999, he has been a Faculty Member with Sejong University, Seoul, South Korea, where he is currently a Professor. His research interests include image and video coding, fusion, watermarking, analyzing, and processing.

SEUNGCHEOL CHOI received the B.S. and M.S. degrees in computer science from Sejong University, Seoul, South Korea, in 1998 and 2001, respectively, and the Ph.D. degree in electronics engineering from Sejong University in 2017. He was a Researcher with Galaxia Communications from 2001 to 2013. He has been a Post-Doctoral Researcher with Sejong University since 2017. His research interests include image and video coding, high-dynamic range imaging, image processing, image fusion, and JPEG.

BEOMYEOL LEE received the B.S. degree in electrical engineering from Sejong University, Seoul, South Korea, in 2018. His research interests include image processing and watermarking.