Robust Lossless Image Data Hiding Designed for Semi-Fragile Image Authentication

Zhicheng Ni, Yun Q. Shi, *Fellow, IEEE*, Nirwan Ansari, *Senior Member, IEEE*, Wei Su, *Senior Member, IEEE*, Qibin Sun, and Xiao Lin, *Senior Member, IEEE*

***Abstract*—Recently, among various data hiding techniques, a new subset, lossless data hiding, has received increasing interest. Most of the existing lossless data hiding algorithms are, however, fragile in the sense that the hidden data cannot be extracted out correctly after compression or other incidental alteration has been applied to the stego-image. The only existing semi-fragile (referred to as robust in this paper) lossless data hiding technique, which is robust against high-quality JPEG compression, is based on modulo-256 addition to achieve losslessness. In this paper, we first point out that this technique has suffered from the annoying salt-and-pepper noise caused by using modulo-256 addition to prevent overflow/underflow. We then propose a novel robust lossless data hiding technique, which does not generate salt-and-pepper noise. By identifying a robust statistical quantity based on the patchwork theory and employing it to embed data, differentiating the bit-embedding process based on the pixel group's distribution characteristics, and using error correction codes and permutation scheme, this technique has achieved both losslessness and robustness. It has been successfully applied to many images, thus demonstrating its generality. The experimental results show that the high visual quality of stego-images, the data embedding capacity, and the robustness of the proposed lossless data hiding scheme against compression are acceptable for many applications, including semi-fragile image authentication. Specifically, it has been successfully applied to authenticate losslessly compressed JPEG2000 images, followed by possible transcoding. It is expected that this new robust lossless data hiding algorithm can be readily applied in the medical field, law enforcement, remote sensing and other areas, where the recovery of original images is desired.**

***Index Terms*—Lossless data hiding, reversible data hiding, robust lossless data hiding, semi-fragile authentication, watermarking.**

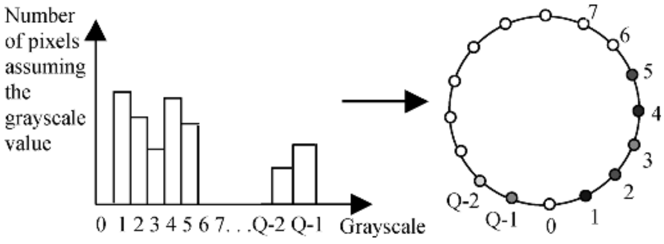
Number of pixels assuming the grayscale value
0 1 2 3 4 5 6 7. . .Q-2 Q-1 Grayscale
7
6
5
4
3
2
1
0
Q-1
Q-2

Fig. 1. Histogram mapping onto a circle.

I. Introduction

DATA hiding has recently been proposed as one of the promising techniques for the purposes of authentication, fingerprinting, security, data mining, and copyright protection. In data hiding, pieces of information represented by some data are hidden in a cover media. Many image data hiding algorithms have been proposed in the past several years. In most cases, the cover media will experience some permanent distortion due to data hiding and cannot be inverted back to the original media.

In some applications, such as in the fields of law enforcement and medical imaging systems, in addition to perceptual transparency, it is desired to reverse the marked media back to the original cover media without any distortions after the hidden data are retrieved for some legal considerations. In military imaging systems and remote sensing, high precision is required; and in some scientific research, experiments are expensive. Under these circumstances, reversibility mentioned above is also desired. The marking techniques satisfying this reversibility requirement are referred to as reversible, lossless, distortion-free, or invertible data hiding techniques. According to our survey [1], recently many lossless data hiding techniques have been proposed, such as those reported in [2]–[9]. However, most of them are fragile in the sense that the hidden data cannot be recovered after compression or other incidental alteration has been applied to the marked image. Thus, far, De Vleeschouwer *et al.*'s method [9] is the only existing robust (or semi-fragile) lossless data hiding technique against high-quality JPEG compression. This technique can be applied to semi-fragile image authentication. That is, on the one hand, if the marked image does not change at all, the hidden data can be extracted out correctly, and the original image can be recovered losslessly, and hence the image is rendered authentic. On the other hand, if the marked image goes through compression to some extent, the hidden data can still be correctly extracted for semi-fragile authentication if the hidden data represent the

Manuscript received September 26, 2006; revised June 27, 2007 and June 28, 2007. The work has been supported in part by the Digital Data Embedding Technologies group of the Air Force Research Laboratory, Rome Research Site, Information Directorate, Rome, NY, under contract F30602-03-1-0264 and in part by New Jersey Commission of Science and Technology via New Jersey Center of Wireless Network and Internet Security. This paper was recommended by Associate Editor M. Barni.

Z. Ni was with the Department of Electrical and Computer Engineering, New Jersey Insitute of Technology, Newark, NJ 07102 USA. He is now with LSI, Inc., Allentown, PA 18109 USA (e-mail: zn2@njit.edu).

Y. Q. Shi and N. Ansari are with the Department of Electrical and Computer Engineering, New Jersey Insitute of Technology, Newark, NJ 07102 USA (e-mail: shi@adm.njit.edu; nirwan.ansari@njit.edu).

W. Su is with the U.S. Army Communication Electronics RD&E Center, Fort Monmouth, NJ 07703 USA (e-mail: wei.su@us.army.mil).

Q. Sun is with the Institute for Infocomm Research, 119613 Singapore (e-mail: qibin@i2r.a-star.edu.sg).

X. Lin is with the Institute for Infocomm Research, 119613 Singapore and also with Fortemedia Inc., 20050 China (e-mail: linxiao@fortemedia.com.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2008.918761

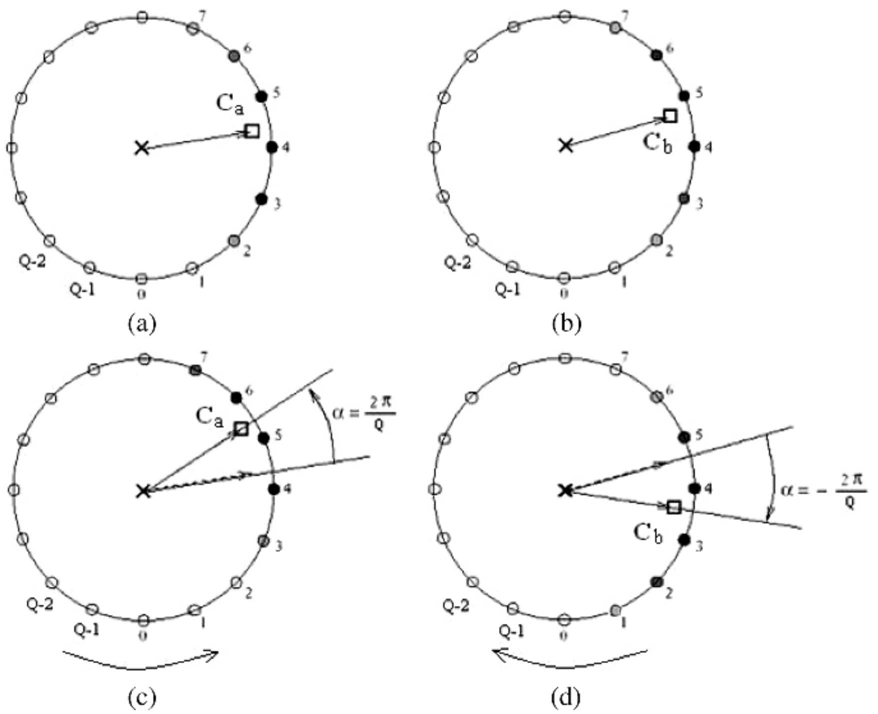

Fig. 2. Data (bit "1") embedding diagram.

compressed version of the image. Semi-fragile authentication may be more practical than fragile authentication for many applications since it allows some incidental modification, e.g., image compression, through which the perceived content of the image has not been changed.

The main idea of De Vleeschouwer *et al.*'s method is based on the patchwork theory [10] and modulo-256 addition. Below is a brief overview of the algorithm.

1) First, each bit of the hidden data is associated with a group of pixels, e.g., a block in an image. Each group is randomly divided into two sets of pixels with an equal size, named zones A and B. The histogram of each zone is mapped to a circle (Fig. 1) where positions on the circle are indexed by the corresponding grayscale values, and the weight of a position is the number of pixels assuming the corresponding grayscale value.
2) In Fig. 2, vectors C_a and C_b point from the center of the circle to the *mass* center of zones A and B, respectively. Since zones A and B are two pseudo-randomly divided sets of equal size within the same block, it is highly probable that vectors C_a and C_b are similar to each other. Slight rotation of these two vectors in opposite directions allows embedding a bit of information in the block. Specifically, C_a may be rotated clockwise and C_b anticlockwise to embed a bit "0," while C_a is rotated anticlockwise and C_b clockwise to embed a bit "1." As to the pixel grayscale values, the rotations of these vectors correspond to shifting of the associated grayscale values by a corresponding amount.
3) The data extraction process is actually the inverse process of the data embedding. The extraction process first partitions the image into blocks and zones A and B in the same way as that used in the embedding process. Histogram of each zone is mapped onto the corresponding circle as in the embedding process. For both zones, the center of mass is computed. Let V denote the difference of the orientation angles between the vectors C_a and C_b. The sign of V provides information of the rotation directions during the embedding process and hence enables bit retrieval. After data extraction, C_a and C_b can be rotated back to the original position, thus achieving the reversibility.

Apparently, the angle difference between the vectors of C_a and C_b depends on all of the pixel grayscale values in zones A and B, respectively. From the patchwork theory, it can be seen that this method is possibly robust against high-quality JPEG compression. The experimental results have verified this claim. However, our extensive investigation has revealed a severe problem suffered by this technique.

In the data embedding process, one may encounter the overflow/underflow problem, which means that after data embedding, the grayscale values of some pixels in the marked image may exceed the upper bound (255 for a gray level image having eight-bit per pixel) and/or the lower bound (0 for eight-bit gray images). This situation will necessitate the use of truncation, hence violating the principle of lossless data hiding. Therefore, avoiding overflow/underflow problem is a key issue in lossless data hiding. From Fig. 2, it is noted that modulo-256 addition is used to handle the overflow/underflow problem in achieving losslessness in this method. Therefore, this algorithm generates the salt-and-pepper noise. That is, in doing modulo-256 addition, a very bright pixel with a large grayscale value close to 255 will be possibly changed to a very dark pixel with a small grayscale value close to 0, and vice versa. One example is shown in Fig. 3 when the algorithm in [9] is applied to a medical image, Mpic1. Obviously, severe salt-and-pepper noise has been ob-

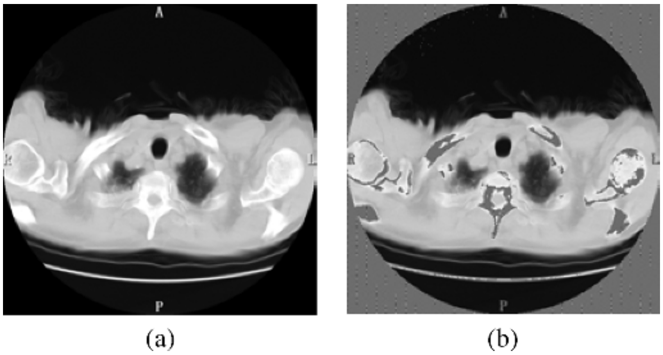

Fig. 3. Medical image, Mpic1. (a) Original. (b) Marked.

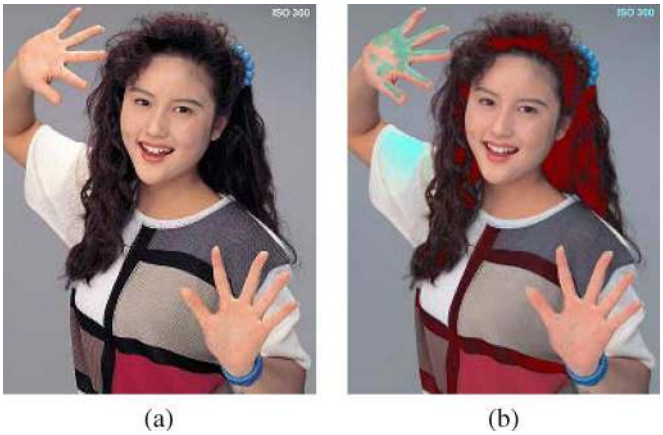

Fig. 4. JPEG2000 test image, woman. (a) Original. (b) Marked.

TABLE I
TEST RESULTS FOR EIGHT MEDICAL IMAGES WITH BLOCK SIZE AS 8, EMBEDDING LEVEL AS 6

Images (512x512)	PSNR of marked image (dB)	Data embedding capacity (bits)	Robustness (bpp)
Mpic1	9.93	100	0.8
Mpic2	4.94	100	1.6
Mpic3	28.10	100	0.8
Mpic4	28.21	100	0.4
Mpic5	28.21	100	0.8
Mpic6	5.86	100	2.0
Mpic7	10.52	100	0.8
Mpic8	6.26	100	1.6

TABLE II
TEST RESULTS FOR EIGHT JPEG2000 TEST IMAGES WITH BLOCK SIZE AS 20, EMBEDDING LEVEL AS 8

Images (1536x1920)	PSNR of marked image (dB)	Data embedding capacity (bits)	Robustness (bpp)
N1A (Woman)	17.73	1410	0.8
N2A	17.73	805	2.2
N3A	23.73	1410	0.6
N4A	19.67	1410	1.2
N5A	17.28	1410	1.2
N6A	23.99	1410	0.6
N7A	20.66	1410	1.4
N8A	14.32	805	1.4

served. The noise is so dense that the name "*salt-and-pepper*" becomes *improper*. Medical images often contain many rather dark and/or rather bright pixels, hence suffering from severe salt-and-pepper noise. Not only for medical images, the salt-and-pepper noise may be severe for daily-life images as well. Fig. 4 presents an example of a rather severe salt-and-pepper noise case on a color image, Woman (one of the JPEG2000 test image). There, the algorithm is applied to the Red component of the image. The salt-and-pepper noise manifests itself as severe color distortion. Specifically, the color of a half of her hair area has changed from black to red, while the color of most of her right-hand palm area has changed from the flesh color to green. Note that authors of [9] also proposed an optional method in the same paper to overcome the salt-and-pepper noise. However, as stated in their paper, this new method is a fragile, instead of semi-fragile, lossless data hiding method.

Another drawback is that the marked image does not have high enough peak signal-to-noise ratio (PSNR) with respect to the original image. Tables I and II summarize the performance of [9] applied to eight medical images and eight JPEG2000 test images. Note that block size means the number of pixels along one side of the square block, while embedding level represents the amount of grayscale value change for each pixel in the block when either bit "1" or "0" is embedded into the block. The reason that the embedding capacity is different among eight JPEG2000 test images is as follows. That is, for those images with a capacity of 1410 bits, BCH (31,6,7) code is used, while for those images with a capacity of 805 bits, BCH (63,7,15) is used. The BCH code is mainly used to improve the robustness against JPEG compression because the JPEG compression will introduce error bits in the payload. It will be detailed illustrated later in this paper (Section II-C). It is observed from Table I that, when 100 information bits are embedded in eight 512 × 512 medical images, the PSNR is below 30 dB. Five marked images suffer from severe salt-and-pepper noise with their PSNR only 10 dB or below. In Table II, when 805 or 1410 bits are embedded in eight 1536 × 1920 JPEG2000 color test images, the PSNR is below 25 dB. Five marked images suffer from severe salt-and-pepper noise resulting in PSNR below 20 dB. Note that in Tables I and II, *robustness (bpp)* means the surviving bit rate in the unit of bpp (bits per pixel), i.e., when a compressed image has a data rate above or equal to this bit rate the hidden data can be retrieved without error.

Therefore, from the above experimental results, our observation is that all of the lossless data hiding algorithms based on modulo-256 addition (e.g., [2], [9]) are not acceptable for many practical usages. Thus, a new robust lossless data hiding technology that do not use modulo-256 addition and hence can avoid the above-mentioned drawbacks is called for.

The rest of the paper is organized as follows. The proposed algorithm is described in Section II. Experimental results are presented in Section III, and conclusions are drawn in Sections IV.

II. A NOVEL ROBUST LOSSLESS IMAGE DATA HIDING ALGORITHM

In order to be robust against image compression, we may select a robust parameter to embed data. In this proposed

+	-	+	-	+	-	+	-
-	+	-	+	-	+	-	+
+	-	+	-	+	-	+	-
-	+	-	+	-	+	-	+
+	-	+	-	+	-	+	-
-	+	-	+	-	+	-	+
+	-	+	-	+	-	+	-
-	+	-	+	-	+	-	+

Fig. 5. Difference pair pattern.

algorithm, the following statistical quantity is selected as the parameter.

A. A Robust Statistical Quantity Used to Embed Data

Consider a given 8 × 8 image block, we split it into two sets A and B as shown in Fig. 5, i.e., set A consists of all pixels marked by "+," denoted by a_i, the set B "-," b_i. Each set has 32 pixels.

For each block, we calculate the difference value α, which is defined as the arithmetic average of differences of grayscale values of pixel pairs within the block. We may choose a pair as two horizontally neighboring pixels (one is marked as "+," another "-," and each pixel is used only once). Below is the formula to calculate α, where n is the number of pixel pairs in the block. In this example, n is equal to 32

$$\alpha = \frac{1}{n}\sum_{i=1}^{n}(a_i - b_i).$$

Since the pixel grayscale values in a local block are often highly correlated and have spatial redundancy, the difference value α is expected to be very close to zero. The experimental results have supported this observation. The distribution of the difference value α of blocks of the "Boat" image is shown in Fig. 6. Note that most values of α are very close to zero (or the mean value of this distribution is zero). The α distributions of other images also follow this pattern. Another point we should mention is that, according to patchwork theory, different ways to split sets A and B are possible. We have tried many other strategies to split sets A and B. Among them, the splitting strategy shown in Fig. 5 seems to have the least variance of α values. Hence, we select this splitting strategy for achieving the least visual distortion of the marked images versus the original image.

Since the difference value α is based on the statistics of all pixels in the block, even though the pixels in the block has small change after JPEG compression, this statistics moment value α is not easy to change. Hence, this value, α, is intrinsically robust against JPEG/JPEG2000 compression and other small incidental alteration. The introduced ECC (Section II-C) will further improve the robustness against JPEG/JPEG2000 compression. Therefore, we select this difference value α as the robust quantity for data embedding.

Note that the block size is not necessary to be 8 × 8. It can be any other even number. But odd block size is not allowed because otherwise the pattern does not always contain couples of pixels. Since, as shown below, each block is used to embed one bit, the block size will thus affect data embedding capacity.

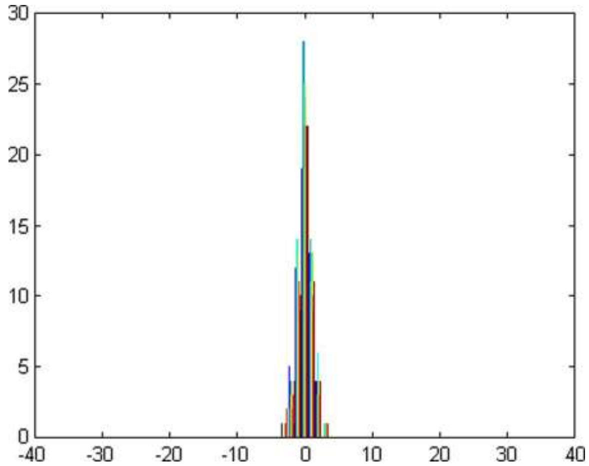

Fig. 6. Distribution of difference value α.

Hence, the larger the block size, the lower the data embedding capacity. The robustness of embedded bits, on the other hand, will be stronger if the block size is larger. Therefore, a compromise between the data embedding capacity and robustness of hidden data needs to be made according to specific applications.

B. Differentiating Bit-Embedding Schemes Based on Different Grayscale Distributions Within a Block of Pixels

We divide a cover image into nonoverlapping blocks. Note that some border areas of the image may not be covered by the nonoverlapping blocks and, hence, will not be used for data embedding. Then one bit is embedded in each block. The main idea for bit-embedding is that the difference value α is kept within specified thresholds K and $-$K (usually K is less than 5 in our numerous experiments) to embed bit "0" and the difference value α is shifted beyond the threshold K or $-$K to embed bit "1." As analyzed in Section I, although modulo-256 addition can avoid the overflow/underflow problem, we have decided not to use it since it will lead to the annoying salt-and-pepper noise. In order to overcome the overflow/underflow problem, we classify the blocks into four different categories and use different bit-embedding schemes for each category. Theory and experimental results show that this technique successfully solves the overflow/underflow problem and avoids the salt-and-pepper noise at the same time. Below are the detailed bit-embedding steps. The block diagram of the algorithm is shown in Fig. 7. In the algorithm, the shift quantity (also referred to as embedding level) β is usually twice of the specified threshold K. Note that shifting α towards the right-hand side (refer to Fig. 9) means adding a fixed shift quantity, β, to the grayscale value of each pixel marked by "+" in set A, and shifting α towards the left-hand side (refer to Fig. 9) means subtracting a fixed shift number, β, from the grayscale value of each pixel marked by "+" in set A. In the whole bit-embedding process, the grayscale values of pixels marked by "-" in set B are kept intact, thus reducing the distortion caused by the bit-embedding. Since error bits may be introduced in data embedding, error correction coding (ECC) is applied to correct them.

Category 1: The pixel grayscale values of a block under consideration are far enough away from the two bounds of the his-

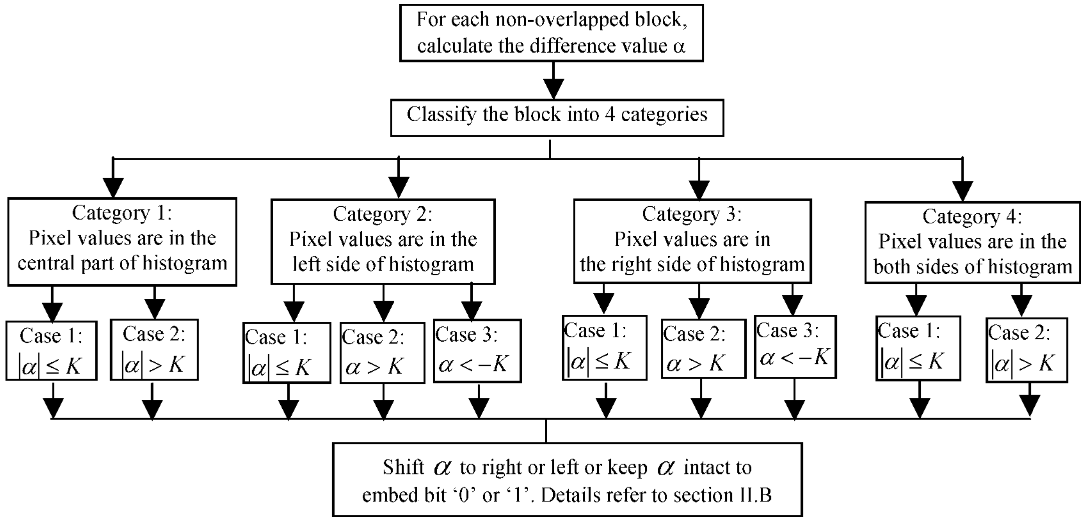
For each non-overlapped block, calculate the difference value α
Classify the block into 4 categories
Category 1: Pixel values are in the central part of histogram
Category 2: Pixel values are in the left side of histogram
Category 3: Pixel values are in the right side of histogram
Category 4: Pixel values are in the both sides of histogram
Case 1: $|\alpha| \le K$
Case 2: $|\alpha| > K$
Case 1: $|\alpha| \le K$
Case 2: $\alpha > K$
Case 3: $\alpha < -K$
Case 1: $|\alpha| \le K$
Case 2: $\alpha > K$
Case 3: $\alpha < -K$
Case 1: $|\alpha| \le K$
Case 2: $|\alpha| > K$
Shift α to right or left or keep α intact to embed bit '0' or '1'. Details refer to section II.B

Fig. 7. Bit embedding algorithm in each block (where K is threshold, α is difference value).

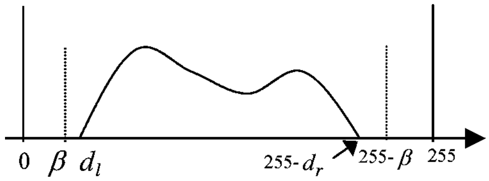

Fig. 8. Block histogram for category 1.

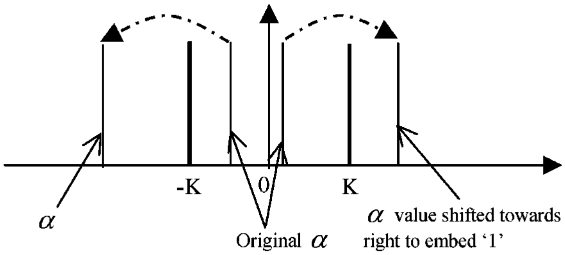

Fig. 9. Embedding a bit "1."

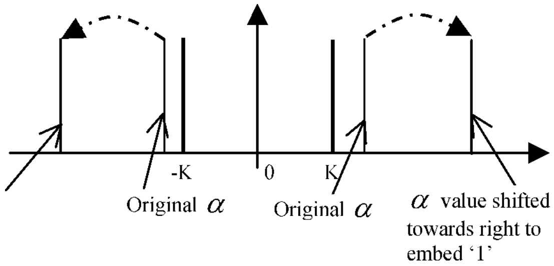

Fig. 10. Embedding a bit "1."

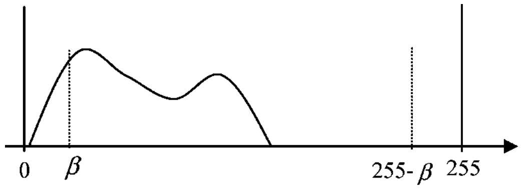

Fig. 11. Block histogram for category 2.

togram (0 and 255 for an 8-bit grayscale image). That is, the distance $d = \min(d_l, d_r)$ satisfies $d \ge \beta$ (where β is the shift quantity), as shown in Fig. 8.

In this category, we further consider the following two cases according to the value of α.

Case 1) The difference value α is located between the thresholds K and $-$K.

1) If the to-be-embedded bit is "1," we shift the difference value α by a quantity β towards the right-hand side or left-hand side depending on if α is positive or negative. Refer to Fig. 9.
2) If the to-be-embedded bit is "0," the pixel value of that block is intact.

Case 2) The absolute value of α exceeds the threshold K.

In order to keep the lossless data hiding principle, no matter whether the to-be-embedded bit is "0" or "1," it always embeds bit "1" by shifting the difference value α further away from 0 by a quantity β, as shown in Fig. 10. In this way, it may introduce an error bit, which will be corrected by using ECC later.

Category 2: Some pixel grayscale values of the block under consideration are very close to the lower bound of the histogram, i.e., 0 for an 8-bit grayscale image, while no pixel grayscale values are close to the upper bound of the histogram, as shown in Fig. 11.

In this category, we further consider three different cases according to the value α.

Case 1) The value α is located between the thresholds K and $-$K.

1) If the to-be-embedded bit is "1," we always shift the difference value α by a quantity β towards the right-hand side beyond the threshold K. Refer to Fig. 12.
2) If the to-be-embedded bit is "0," the pixel value of that block is intact.

Case 2) The value α is located on the right-hand side beyond the threshold K.

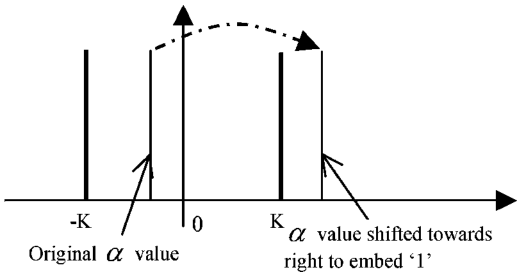

Fig. 12. Embedding a bit "1."

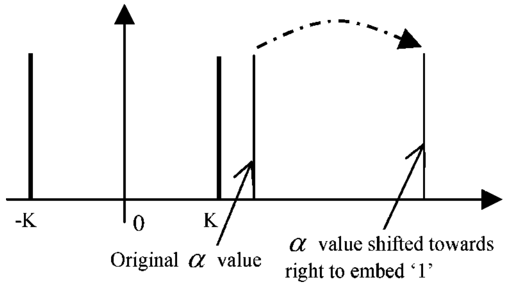

Fig. 13. Embedding a bit "1."

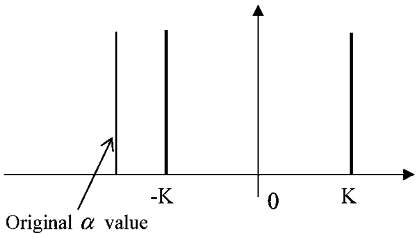

Fig. 14. The pixel grayscale value of the block is unchanged. Bit "0" is assumed to be embedded.

No matter whether the to-be-embedded bit is "0" or "1," it always embeds bit "1" by shifting the difference value α by a quantity β, and thus making α further away from the zero point, as shown in Fig. 13. In this way, it may introduce an error bit, which will be corrected by using ECC.

Case 3) The value α is located on the left-hand side beyond the threshold $-$K.

In this case (Fig. 14), we do not change pixel grayscale values for the block. We always embed bit "0" into the block regadrless the to-be-embedded bit is bit "0" or bit "1." In decoding, the grayscale value distribution of the block is first examined. Once Case 3 of Category 2 is identified, bit "0" is extracted, and the grayscale values of this block will remain unchanged. The possibly introduced error bit will be corrected by using ECC.

Category 3: Some pixel grayscale values of the block under consideration are very close to the upper bound of the histogram, while no pixel grayscale values are close to the lower bound of the histogram, as shown in Fig. 15.

Category 3 is similar to Category 2 except that the distribution of grayscale values of the block is close to the upper bound instead of the lower bound of the histogram. Hence, the data embedding algorithm of Category 3 is similar to that of Category 2 except shifting difference value α by a quantity β to the left-hand side instead of to the right-hand side.

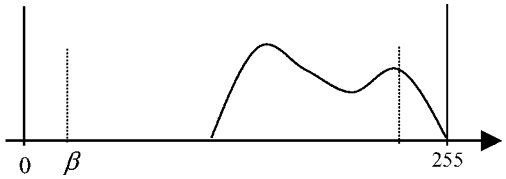

Fig. 15. Block histogram for category 3.

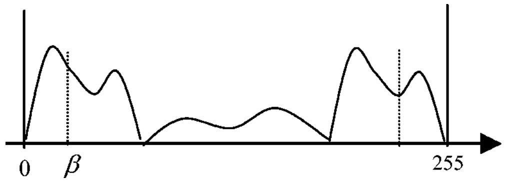

Fig. 16. Block histogram for category 4.

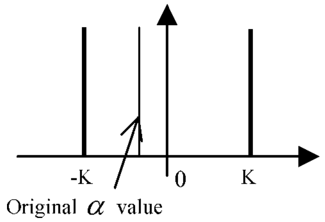

Fig. 17. Embedding a bit "0."

Category 4: Some pixel grayscale values of the block under consideration are close to the upper bounds, while some pixel grayscale values are close to the lower bounds of the histogram, as shown in Fig. 16.

In this category, we further consider two different cases according to the α value.

Case 1) The value α is located between the thresholds K and $-$K.

No matter whether the to-be-embedded bit is "0" or "1," it always embeds bit "0" by keeping the difference value α intact, as shown in Fig. 17. In this way, it may introduce an error bit, which is to be corrected by using ECC.

Case 2) The absolute value α is beyond the threshold K.

In this case (Fig. 18), we do not change pixel grayscale values for the block. We always embed bit "0" into the block regadrless the to-be-embedded bit is bit "0" or bit "1." In decoding, the grayscale value distribution of the block is first examined. Once Case 2 of Category 4 is identified, bit "0" is extracted, and the grayscale values of this block will remain unchanged. The possibly introduced error bit will be corrected by using ECC. Same as case 3 of category 2 and case 2 of category 3.

The previous mentioned four categories cover all situations that a block may encounter. The detailed description of the bit-embedding procedure apparently demonstrates that the modified pixel grayscale value is still in the range of [0,255], and hence no overflow/underflow problem will take place. It is noted

TABLE III
CATEGORY AND CASE OCCURRENCE OF SOME TEST IMAGES

	Category 1		Category 2			Category 3			Category 4	
	Case 1	Case 2	Case 1	Case 2	Case 3	Case 1	Case 2	Case 3	Case 1	Case 2
Lena	1024	0	0	0	0	0	0	0	0	0
Baboon	999	3	22	0	0	0	0	0	0	0
Boat	987	0	27	0	0	8	0	0	2	0
Mpic1	536	0	319	1	0	147	0	0	19	2
Mpic2	141	0	646	0	0	146	0	0	88	3
Mpic3	1012	0	0	0	0	0	0	0	12	0
Mpic4	1012	0	0	0	0	1	0	0	11	0
Mpic5	1012	0	0	0	0	1	0	0	11	0
Mpic6	242	0	646	0	0	82	0	0	52	2
Mpic7	517	0	451	0	0	44	0	0	11	1
Mpic8	366	0	593	0	0	57	0	0	8	0
N1A	3313	0	927	0	0	794	0	0	86	0
N2A	1479	0	1404	1	2	847	0	0	1382	5
N3A	4143	1	408	0	0	475	0	1	92	0
N4A	3151	0	160	0	0	1546	0	0	263	0
N5A	1542	1	114	0	0	3027	8	11	405	12
N6A	4273	0	79	0	0	707	0	0	61	0
N7A	3089	3	1009	0	0	169	1	1	839	9
N8A	822	0	2178	1	1	265	0	1	1841	11

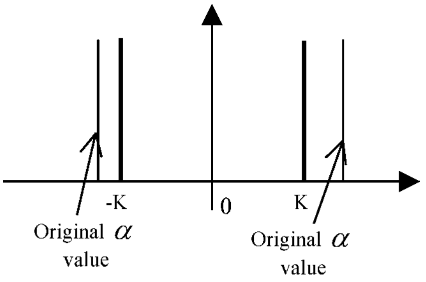

Fig. 18. The pixel grayscale value of the block is unchanged. Bit "0" is assumed to be embedded.

that after data embedding some error bits will be generated. They will be corrected by using error correction code as illustrated in the next section.

C. Error Correction Code

Table III provides statistics of the Categories and their Cases (shown in Fig. 7) for some test images used in this work. These test images listed in Table III can be grouped into three sets. The first set consists of three commonly used images: Lena, Baboon and Boat. The second set consists of eight medical images. The last set is eight JPEG2000 test images. The first two sets of images are of 512 $\times$ 512 (block size is 8 $\times$ 8). The last set of images is of 1536 $\times$ 1920 (block size is 24 $\times$ 24). Table III contains the category and case occurrence for these test images. Two observations can be made from Table III. The first observation from Table III that we can make is as follows. That is, Case 1 occurs most often. Specifically, if we add all of Case 1 from all of four Categories, we have the total appearance of Case 1 for three commonly used images occupying 99.71%, for eight medical images 99.71%, for eight JPEG2000 images 99.375%, That is, average speaking, for all of the above-mentioned test images, the occurrence rate of Case 1 is more than 99%. This verifies our statistical analysis presented in Section II-A, i.e., the theoretical deduction that the distribution of α is heavily at zero and a small range surrounding zero as shown in Fig. 6.

The second observation is that Case 1 of Category 4 appears more often in eight medical images than in three commonly used images, while appears most often in eight JPEG2000 test images. In particular, Case 1 of Category 4 appears in N2A by 13.5% of total blocks and in N8A by 18% of total blocks (in deriving these percentage numbers, the assumption that binary 0 and 1 are equally probable is used). This observation has led us to use a stronger error correction code (the BCH (63,7,15) code as shown below) in lossless data embedding into the JPEG2000 images in our experimental works.

The bit-embedding process, as discussed in Section II-B, may introduce some error bits. Our thorough experiments on some commonly used images (Lena, Baboon, Boat, etc.), eight medical images, all of 1096 images in the CorelDRAW database, and all of eight JPEG2000 test images have verified that the raw error bits may take place in Case 2 of Category 1, Cases 2 and 3 of Category 2, Cases 2 and 3 of Category 3, and Category 4. These cases, however, are rather limited as shown in Table III. The detailed experimental results about raw bits error are shown in the Table IV.

In order to recover the information bits correctly, error correction code (ECC) is utilized to correct error bits at the price of sacrificing some data embedding capacity. Considering that Bose–Chadhuri–Hocquenghem (BCH) codes are a powerful class of cyclic codes that provide a large selection of block lengths, code rates, alphabet sizes, and error-correcting capability [11], our algorithm includes a few BCH codes for selection. Specifically, they are BCH (15,11,1), BCH (15,7,2), BCH (15,5,3), BCH (31,6,7), and BCH (63,7,15), respectively. The utilization of these BCH codes can then facilitate the tradeoff between the coding ratio of ECC (hence the payload) and the error-correcting capability of ECC (hence robustness of lossless data hiding). For example, BCH (63,7,15) code is the most powerful code among these several codes in terms of

TABLE IV
RAW BITS ERROR ON DIFFERENT TEST IMAGES

Image	Size	Payload (bits)	Average raw error bits	Raw error bits percentage (%)
Lena	512×512	1024	0	0
Baboon	512×512	1024	2	0.2
Boat	512×512	1024	1	0.1
1096 images in CorelDraw database	512×768	1536	3	0.2
8 medical image	512×512	1024	14	1.4
8 JPEG2000 color test images	1536×1920	5120	367	7.2

error correction capability. It can correct 15 random error bits within a codeword of 63 bits with the price of more redundant bits, resulting in a smallest payload among these codes. For example, in the worst case of experiments on eight JPEG2000 test images, even though there are about 7.2 % raw error bits. All BCH (15,5,3), BCH (31,6,7) and BCH (63,7,15) can correct at least 20 percent random raw error bits. This error correction capability is much higher than experimental requirement. Hence, after the introduction of error correction code, the payload can be recovered without error bits.

D. Permutation of Message Bits

For some images, error bits may be concentrated in some areas in an image, which possibly leads to too many error bits in one codeword, thus causing error in data extraction. In this case, even the powerful code BCH (63,7,15) cannot correct all of the error bits. To efficiently combat this type of *bursts of errors*, which may fail our algorithm, some kind of permutation scheme should be used together with ECC. As pointed out in [12], combining ECC and permutation is an effective and efficient strategy to efficiently combat both random error and bursts of errors. Here we should mention that the permutation is applied on the ECC coded message instead of the original message. For the sake of security, the message bits are permuted using a secret key in the proposed algorithm.

E. Data Embedding Pseudo Code and Diagram

After theoretical analysis, experiments on thousands of images, and the tradeoff consideration, we deduce the empirical numbers for some parameters' high limit and low limit. For instance, the smallest block size is 8. The biggest block size is 20. The threshold K is from 2 to 5. The embedding level β is twice the K, which means β is from 4 to 10. Error correction code is selected among five types of BCH codes described above. Then, the pseudo code of the whole algorithm is as following.

```
for BlockSize = SmallBlockSize to LargeBlockSize
    for EmbeddingLevel $\beta$ = low to high
        According the payload, select the suitable BCH
        code. If no BCH is found, back to step 2.
        Permutation
        Set threshold K = half of the embedding level $\beta$
        Embed one bit for each block as shown in
        Section II-B
    end
end
```

The corresponding diagram is shown in Fig. 19.

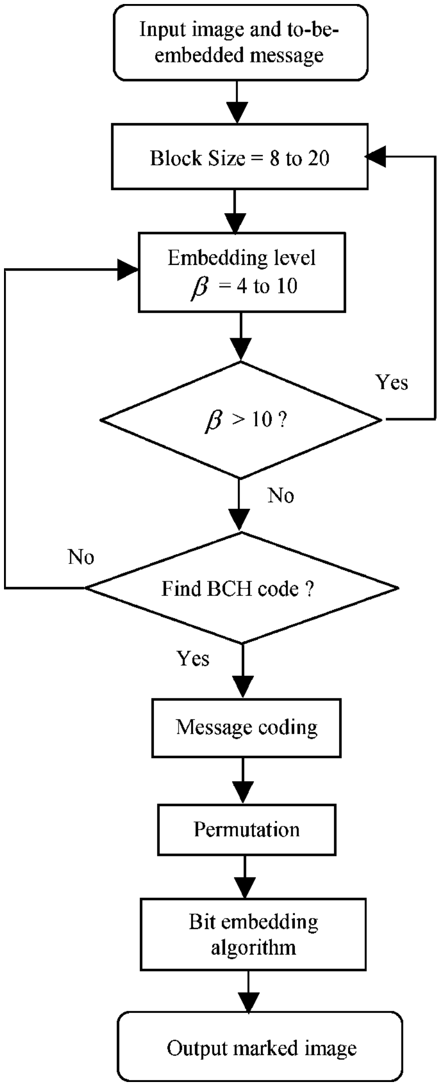

Fig. 19. Block diagram of data embedding.

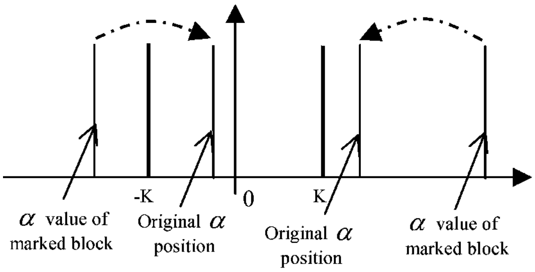
α value of marked block
-K
Original α position
0
K
Original α position
α value of marked block

Fig. 20. Extracting bit "1."

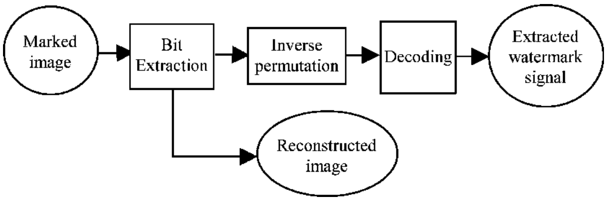

Fig. 21. Block diagram of data extraction.

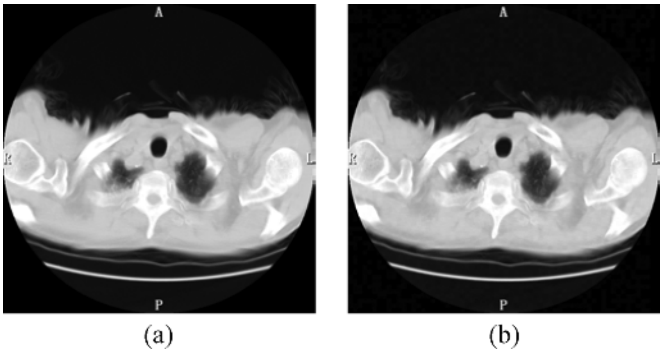

Fig. 22. Medical image, Mpic1. (a) Original. (b) Marked.

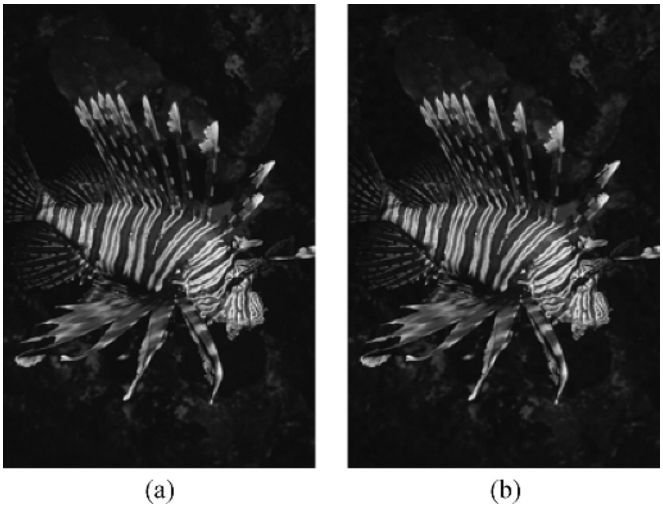

Fig. 23. CorelDraw image. (a) Original. (b) Marked.

F. Data Extraction

Data extraction is actually the reverse process of data embedding and is much simpler than data embedding. For a given marked image, we first split it into nonoverlapping blocks and then calculate the difference value α for each block in the same way as that used in data embedding. The main steps are described below.

1) If the absolute difference value α is larger than the threshold K, the grayscale value distribution of the block is then examined. If the block is identified as Case 3 in Category 2, Case 2 in Category 3, or Case 2 in Category 4, the bit "0" is extracted, and the block remains unchanged. Otherwise, bit "1" is extracted and the difference value $alpha$ is shifted back towards the zero point by adding or subtracting the quantity β, depending on whether the difference value α is negative or positive. In this way, the difference value α is reverted back to its original value, which means each pixel grayscale value in set A is back to its original value, as shown in Fig. 20.
2) If the absolute value of the difference value α is less than the threshold K, then bit "0" is extracted, and nothing to do on the pixel grayscale value of that block.
   Note that by combining this step and the above step in data extraction, it is obvious that all pixel grayscale values will be the same as in the original image.
3) After data extraction, the inverse permutation and the ECC decoding are applied, respectively, so as to obtain the original information bits correctly. In this way, we can extract the original information bits and recover the original image without any distortion. The whole data extraction diagram is depicted in Fig. 21.

III. Experimental Results

We have successfully applied our proposed algorithm to commonly used grayscale images such as Lena, Baboon and Boat, etc., eight medical images, eight JPEG2000 color test images, and all of 1096 images in the CorelDraw image database. For color images, only one color plane is applied by the algorithm. Note that there is no salt-and-pepper noise in all of tests since the proposed algorithm does not use modulo-256 addition. The embedding capacity can range from 512 to 1024 bits for the purpose of authentication, and it can be adjusted by changing the block size for other applications. As shown later the PSNR is much higher than that obtained by using the method in [9]. It is noted that the data embedding capacity and the PSNR of the marked image versus the original image can be adjusted according to the requirements. Since these two performance parameters are usually conflicting each other in the sense that if the embedding capacity is improved, the PSNR will drop and vice versa, there is usually a tradeoff between the data embedding capacity and the PSNR of the marked image versus the original image for a targeted application. The tested images can resist the JPEG/JPEG2000 compression with the surviving bit rate ranging from 2.0 bpp (bite per pixel) to 0.2 bpp (bite per pixel). In other words, the hidden data can be retrieved with no error when image compression is applied to marked images with the resultant bit rate in the unit of bpp equal to or above the above-mentioned surviving bit rate.

The following (Figs. 22–24) are some test examples. Note that no visible artifacts exist, indicating a significant performance improvement has been achieved as compared with [9]. For the Woman color image, we applied the data embedding in

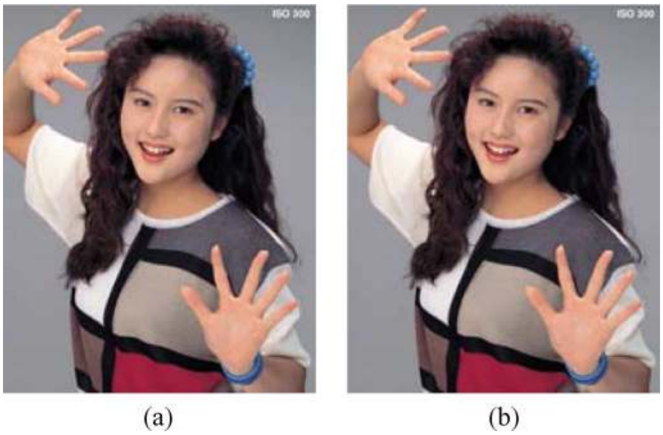

Fig. 24. JPEG2000 test image, woman. (a) Original. (b) Marked.

TABLE V
TEST RESULTS FOR COMMONLY USED $512 \times 512 \times 8$ GRAYSCALE IMAGES

	Lena	Baboon	Boat
PSNR (dB)	40.2	38.7	40.5
Capacity (bits)	792	585	560
Robustness (bpp)	0.8	1.6	1.0

TABLE VI
TEST RESULTS FOR ALL OF 1096 IMAGES IN CORELDRAW DATABASE

Images (512×768)	PSNR of marked image (dB)			Data embedding capacity (bits)	Robustness (bpp)		
	Max	Min	Avg	714	Max	Min	Avg
	45.2	37.4	40.2		2.0	0.2	1.21

the red component, same as that applied in the Fig. 4. Notice that there is no color distortion at all in the marked Woman image as shown in Fig. 24(b) and the PSNR is above 41 dB. If there is any noticeable difference in the image shown in printed paper, that may be caused during the scaling down of geometric size or PDF conversion.

Tables V–VIII summarize the test results for three commonly used images, 1096 images in the CorelDraw database, eight medical images, and eight JPEG2000 test images, respectively. Note that once the payload requirement is fixed, the program can automatically select the optimum combination of block size, embedding level and BCH code to obtain the best PSNR. Hence, these three parameters are not fixed for every image in Tables V and VI. Also note that the block size and the embedding level, whose meaning will be defined later in this section, for Tables VII and VIII are the same as used in Tables I and II, respectively in order to make the results compatible to that achieved by using the algorithm in [9]. For the same consideration, 100 information bits are embedded as listed in both Tables I and VII because 100 information bits are also embedded in 512 by 512 grayscale images in [9] (majority voting is used, which is to be explained later in this section).

In order to compare the performance between the modulo-256 based algorithm [9] and the proposed algorithm in a more accurate way, we have conducted a set of experiments on the eight medical images. Note that all the data shown below are the average of test results on these eight medical images. In this set of experiments, we change block size and embedding level to observe the PSNR of marked images versus original images, and the robustness against image compression. The block size represents the number of pixels along one side of a square block. The embedding level denotes the amount of grayscale value change within the block for a bit “1” or “0” to be embedded into the block. Apparently, the larger the embedding level, the lower the PSNR of marked image, and the stronger the robustness of hidden data against image compression, and vice versa. In order to make comparison accurate, we follow the description of [9] to embed 100 information bits into image. When the number of blocks, hence the embedding capacity is large, we repeatedly embedding the same 100 information bits. After hidden data extraction, we use majority voting to decode the hidden information bits. Therefore, it is obvious that, for each given block size, the embedding capacity for two different methods is the same. It is known that the PSNR of a marked image versus its original image depends mainly on embedding level. Fig. 25 demonstrates that our proposed algorithm has much higher PSNR than the algorithm presented in [9] does no matter what the block size is. Here we should mention frankly that from the experiments, for those test images without overflow problems (such as all pixel values are within 20–235 range), the PSNR and robustness performance from the algorithm presented in [9] are almost same as our proposed method. The robustness comparison of two algorithms is shown in Fig. 26, where the bit-per-pixel (bpp) means that at that

TABLE VII
TEST RESULTS FOR EIGHT MEDICAL IMAGES WITH BLOCK SIZE AS 8, EMBEDDING LEVEL AS 6

Images (512×512)	PSNR of marked image (dB)	Data embedding capacity (bits)	Robustness (bpp)
Mpic1	37.6	100	0.4
Mpic2	37.7	100	0.8
Mpic3	37.6	100	0.4
Mpic4	37.6	100	0.8
Mpic5	37.6	100	0.4
Mpic6	37.6	100	1.2
Mpic7	37.6	100	0.4
Mpic8	37.6	100	0.8

TABLE VIII
TEST RESULTS FOR EIGHT JPEG2000 COLOR TEST IMAGES WITH BLOCK SIZE AS 20, EMBEDDING LEVEL AS 8

Images (1536×1920)	PSNR of marked image (dB)	Data embedding capacity (bits)	Robustness (bpp)
N1A (Woman)	41.5	1410	0.8
N2A	41.3	805	1.2
N3A	41.3	1410	0.8
N4A	41.4	1410	0.8
N5A	41.3	1410	0.8
N6A	41.2	1410	0.4
N7A	41.2	1410	0.8
N8A	41.2	805	1.2

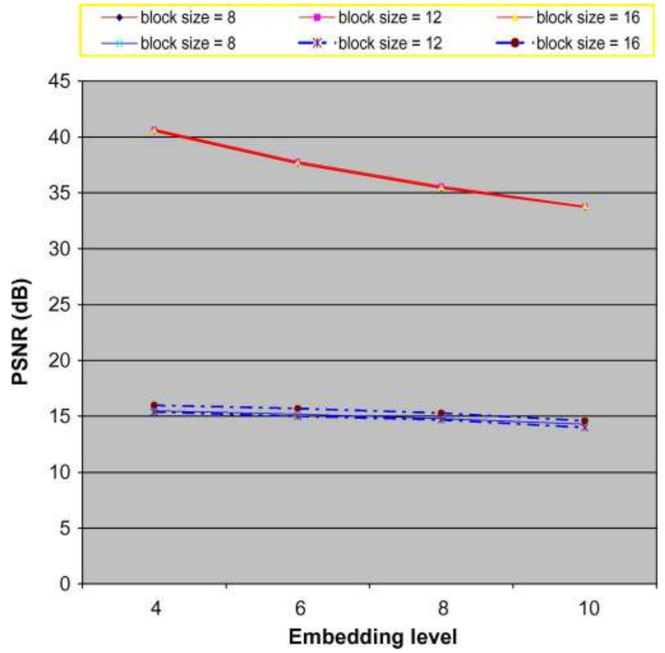

Fig. 25. PSNR comparison (three red solid lines at the top are associated with the proposed algorithm, three blue dashed lines at the lower portions are with the algorithm reported in [9]), vertical axis stands for PSNR of marked image versus cover image, while horizontal axis stands for embedding level.

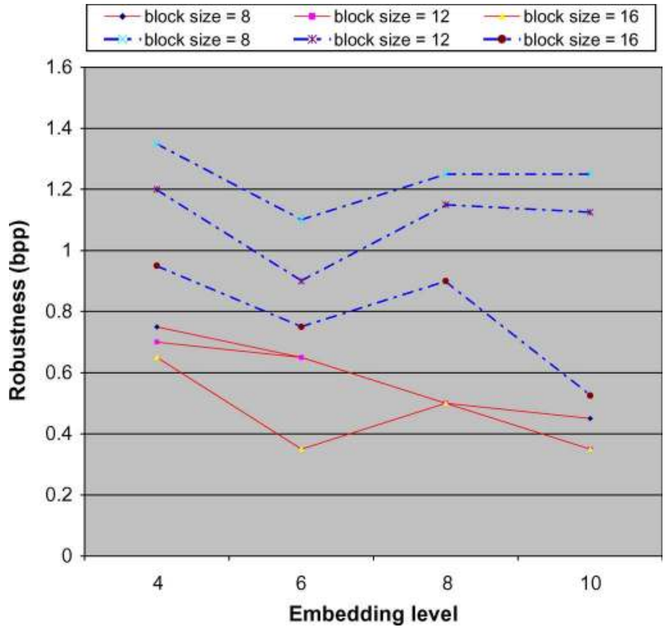

Fig. 26. Robustness comparison (three red solid lines at the lower portion are associated with the proposed algorithm, three blue dashed line at the top are with the algorithm reported in [9]), vertical axis stands for robustness in terms of bpp and horizontal axis stands for embedding level.

compression level, the embedded 100 bits can still be fully recovered. Fig. 26 shows that our proposed algorithm (red continuous lines) has lower bpp (higher robustness) than that proposed in [9] (blue dashed lines). It is also observed that larger embedding level and block size will lead to stronger robustness. This observation is consistent to the theoretical analysis. In summary, better performance has been achieved with the proposed method, compared with that in [9].

IV. Conclusion

We have proposed a novel robust lossless image data hiding scheme, which employs a robust statistical quantity to mitigate the effect of image compression and small incidental alteration for data embedding. It utilizes different bit-embedding strategies for groups of pixels with different pixel grayscale value distributions. It employs error correction codes together with permutation scheme. Consequently, it has successfully avoided using modulo-256 addition to achieve losslessness, thus eliminating the annoying salt-and-pepper noise. This technique has the following advantages: 1) no salt-and-pepper noise; 2) applicable to virtually all images (the algorithm has been successfully tested in some commonly used images, eight medical images, all of 1096 images in the CorelDRAW database, and all of eight JPEG2000 test images); 3) average PSNR of marked images is above 38 dB; 4) robust to JPEG/JPEG2000 compression; and 5) data embedding capacity ranges from 512 bits to 1024 bits (often sufficient for authentication purpose), and the embedding capacity can be adjusted according to the requirement.

This proposed scheme [13] can be utilized to embed some digital signature related data to authenticate losslessly compressed JPEG2000 images, followed by possible transcoding [14]. The unified authentication framework provides both fragile and semi-fragile authentication. The former is for data integrity verification, while the latter for content integrity verification. Further more, there are lossy and lossless two modules in the semi-fragile authentication. The robust lossless data hiding scheme reported here is used for the lossless module. Specifically, if a losslessly compressed JPEG2000 image has not been altered before authentication, the hidden data can be extracted correctly, the image will be classified as authentic and the original image can be recovered exactly. If the losslessly compressed JPEG2000 image has experienced further transcoding such as lossy compression, it will be rendered authentic as long as the compression is not so severe that the content has been changed. At this point, the hidden data can be extracted correctly. Of course, the original image will not be able to be recovered. If the lossy compression is so severe that the resultant bit rate is lower than the specified minimum surviving bit rate, the hidden data will not be extracted correctly, and the image will be rendered nonauthentic. If the content of the losslessly compressed image has been altered, then even the hidden data can be extracted out without error, the extracted data will render the altered image nonauthentic because the hidden data dismatch the altered content. For detail, please refer to [14]. This unified authentication framework for JPEG2000 images has been included in the Security Part of JPEG2000 (known as JPSEC) IS (International Standard), JPSEC ISO/IEC 15444-8:2007, April 2007.

Acknowledgment

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Air Force Research Laboratory, or the U.S. Government.

References

[1] Y. Q. Shi, Z. Ni, D. Zou, C. Liang, and G. Xuan, "Lossless data hiding: Fundamentals, algorithms and applications," in *Proc. IEEE Int. Symp. Circuits Syst.*, Vancouver, BC, Canada, May 2004, vol. II, pp. 33–36.

[2] C. W. Honsinger, P. Jones, M. Rabbani, and J. C. Stoffel, "Lossless recovery of an original image containing embedded data," U.S. Patent 6 278 791, Aug. 21, 2001.

[3] J. Fridrich, M. Goljan, and R. Du, "Invertible authentication," in *Proc. SPIE Photonics West, Security and Watermarking of Multimedia Contents III*, San Jose, CA, Jan. 2001, vol. 397, pp. 197–208.

[4] M. Goljan, J. Fridrich, and R. Du, "Distortion-free data embedding," in *Proc. 4th Inf. Hiding Workshop*, Pittsburgh, PA, Apr. 2001, pp. 27–41.

[5] G. Xuan, J. Zhu, J. Chen, Y. Q. Shi, Z. Ni, and W. Su, "Distortionless data hiding based on integer wavelet transform," in *Proc. IEEE Int. Workshop Multimedia Signal Process.*, St. Thomas, U.S. Virgin Islands, Dec. 2002, vol. 38, no. 25, pp. 1646–1648.

[6] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," in *Proc. IEEE Int. Symp. Circuits Syst.*, Bangkok, Thailand, May 2003, pp. 912–915.

[7] M. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Reversible data hiding," in *Proc. Int. Conf. Image Process. 2002*, Rochester, NY, Sep. 2002, pp. 157–160.

[8] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 8, pp. 890–896, Aug. 2003.

[9] C. De Vleeschouwer, J. F. Delaigle, and B. Macq, "Circular interpretation of bijective transformations in lossless watermarking for media asset management," *IEEE Trans. Multimedia*, vol. 5, no. 1, pp. 97–105, Mar. 2003.

[10] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM Syst. J.*, vol. 35, no. 3–4, pp. 313–336, 1996.

[11] J. G. Proakis, *Digital Communication*, 4th ed. New York: McGraw-Hill, 2000.

[12] S. B. Wicker, *Error Control System for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[13] Z. Ni, Y. Q. Shi, N. Ansari, W. Su, Q. Sun, and X. Lin, "Robust lossless iamge data hiding," in *Proc. IEEE Int. Conf. Multimedia Expo*, Taipei, Taiwan, R.O.C., Jun. 2004, pp. 2199–2202.

[14] Z. Zhang, Q. Sun, X. Lin, Y. Q. Shi, and Z. Ni, "A unified authentication framework for JPEG2000 images," in *Proc. IEEE Int. Conf. Multimedia Expo*, Taipei, Taiwan, R.O.C., Jun. 2004, pp. 915–918.

[15] *Secure JPEG 2000 (JPSEC)*, ISO/IEC 15444-8:2007, Apr. 2007.

Zhicheng Ni received the B.E. degree in electrical engineering from Southeast University, Nanjing, China, the M.E. degree in electrical engineering from Nanyang Technological University, Singapore in 1994 and 2000, respectively, and the Ph.D. degree in electrical engineering from New Jersey Institute of Technology, Newark, in 2004.

From 1994 to 1998, he was an Electronic Engineer in the Power Research Institute, China. From 2005 to 2007, he was a Senior Research and Software Engineer in WorldGate, PA. He is currently a Senior Member of Technical Staff with LSI, Inc., Allentown, PA. His research interests include digital watermarking, image data hiding, computer vision, document image processing, authentication, video compression, and DSP. He has three patents and three journal papers.

Yun Q. Shi (M'90–SM'93–F'05) received the B.S. and M.S. degrees from Shanghai Jiao Tong University, Shanghai, China, and the M.S. and Ph.D. degrees from University of Pittsburgh, Pittsburgh, PA.

He is a Professor of Electrical and Computer Engineering at New Jersey Institute of Technology. Newark, since 1987. His research interests include visual signal processing and communications, digital multimedia data hiding and information assurance, theory of multidimensional systems and signal processing. Some of his research projects have been supported by several federal and New Jersey State funding agencies. He is an author/coauthor of 200 papers, a book on image and video compression, three book chapters on image data hiding and one book chapter on digital image processing. He holds two U.S. patents and has additional 20 U.S. patents pending.

Dr. Shi is the founding Editor-in-Chief of *LNCS Transactions on Data Hiding and Multimedia Security*. He was Technical Program Chair of ICME07, Co-Chair of IWDW06, IWDW07, MMSP05, and Co-General Chair of MMSP02.

Nirwan Ansari (S'78–M83–SM'94) received the B.S.E.E. degree (*summa cum laude*) from the New Jersey Institute of Technology (NJIT), Newark, in 1982, the M.S.E.E. degree from the University of Michigan, Ann Arbor, in 1983, and the Ph.D. degree from Purdue University, West Lafayette, IN, in 1988.

He joined the Department of Electrical and Computer Engineering, NJIT, as an Assistant Professor in 1988, and has been a Full Professor since 1997. He authored *Computational Intelligence for Optimization* (Kluwer, 1997) with E.S.H. Hou and translated into Chinese in 2000, and coedited *Neural Networks in Telecommunications* (Norwell, MA: Kluwer, 1994) with B. Yuhas. He is a Senior Technical Editor of the *IEEE Communications Magazine*, and also serves on the editorial board of *Computer Communications*, the *ETRI Journal*, and the *Journal of Computing and Information Technology*. His current research focuses on various aspects of broadband networks and multimedia communications. He has also contributed approximately 100 refereed journal articles, plus numerous conference papers and book chapters.

Dr. Ansari initiated (as the General Chair) the First IEEE International Conference on Information Technology: Research and Education (ITRE'03), was instrumental, while serving as its Chapter Chair, in rejuvenating the North Jersey Chapter of the IEEE Communications Society, which received the 1996 Chapter of the Year Award and a 2003 Chapter Achievement Award, served as Chair of the IEEE North Jersey Section and in the IEEE Region 1 Board of Governors during 2001–2002, and has been serving in various IEEE committees, such as TPC Chair/Vice Chair of several conferences. He was the 1998 recipient of the NJIT Excellence Teaching Award in Graduate Instruction, and a 1999 IEEE Region 1 Award. He is frequently invited to deliver keynote addresses, tutorials, and talks. He has been selected as an IEEE Communications Society Distinguished Lecturer (2006–2007).

Wei Su (M'89–SM'95) received the B.S. degree in electrical engineering and the M.S. degree in systems engineering from Shanghai Jiao Tong University, China, in 1983 and 1987, respectively, and the Ph.D. degree in electrical engineering from The City University of New York, New York, in 1992.

He has been a Senior Research Engineer in the U.S. Army Communication Electronics Research Development and Engineering Center (CERDEC), Fort Monmouth, NJ, since 1998. From 1991 to 1997, he was with U.S. Army Research Laboratory, Fort Monmouth. His research interests include wireless communication, signal and image processing, and adaptive control.

Dr. Su was a recipient of Superior Civilian Service Award and Medals, 2005 Army Research and Development Achievement Award, Army Material Command Top 10 Employee Nomination, 2004 and 2007 AOC International Research and Development Awards, and 2002 Thomas Alva Edison Patent Award. He is also recognized by Army CERDEC for Inventor's Wall of Honor and Electronic Warfare & Information Operations Association for Electronic Warfare Technology Hall of Fame.

Qibin Sun received the Ph.D. degree in electrical engineering from University of Science and Technology of China (USTC), Anhui, China, in 1997.

Since 1996, he has been with the Institute for Infocomm Research, Singapore, where he is responsible for industrial as well as academic research projects in the area of media security, image and video analysis, etc. He worked in Columbia University, New York, during 2000–2001 as a Research Scientist. He is the Head of Delegates of Singapore in ISO/IEC SC29 WG1(JPEG). He led the effort to successfully bring the robust image authentication technology into ISO JPEG2000 standard Part 8 (Security). He has published more than 120 papers on international journals and conferences.

Dr. Sun actively participates in professional activities such IEEE ICME, IEEE ISCAS, IEEE ICASSP and ACM MM, etc. He now serves as the member of Editorial Board in IEEE Multimedia Magazine, the member of Editorial Board in LNCS Transactions on Data Hiding and Multimedia Security, the member of Editorial Board in EUROSIP on Information Security and the Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS ON VIDEO TECHNOLOGY. He and his team won the best paper award in ICME2006 and the best student paper award in ICME2007.

Xiao Lin (M'00–SM'02) received the Ph.D. degree from the Electronics and Computer Science Department, University of Southampton, Southampton, U.K., in 1993.

He worked with Centre for Signal Processing (CSP) for about 5 years as a Research Fellow and Senior Research Fellow. After leaving CSP he joined DeSOC Technology as a Technical Director. He then joined Institute for Infocomm Research, Singapore, in 2002, where he was a Member of Technical Staff, Lead Scientist, and Principal Scientist and managed the Media Processing Department until 2006. He is now with Fortemedia Inc., China, as a Senior Director. He actively participated in the International Standards such as MPEG-4, JPEG2000, and JVT.

Dr. Lin was awarded the 2007 National Technology Award by Republic of Singapore in recognition of his contribution to MPEG-4 audio standard.