# Reversible watermarking resistant to cropping attack

S.W. Weng, Y. Zhao and J.-S. Pan

**Abstract:** A reversible digital watermarking scheme based on a blockwise difference expansion (DE) method is proposed here. The proposed scheme differs from previous ones in its robustness against cropping and collage attacks. In order to easily detect the correct cropped position for a cropped image, a locating pattern is embedded into the carrier image by a modified patchwork algorithm. Incurred overhead, payload, and authentication information are then collected and embedded into the carrier image by a blockwise DE method. Within the proposed scheme, image ID and block index also serve as parts of the inputs to the hash function. This endows the scheme with capability in resisting collage attacks. The experimental results show that the scheme can achieve a good robustness against cropping attacks as well as collage attacks.

## 1 Introduction

For some critical applications such as the law enforcement, medical and military image system, it is crucial to restore the original image without any distortions. The watermarking techniques satisfying those requirements are referred to as 'reversible watermarking'. Reversible watermarking is designed so that it can be removed to completely restore the original image.

The concept of reversible watermark firstly appeared in the patent owned by Eastman Kodak [1]. Honsinger *et al.* [1] utilised a robust spatial additive watermark combined with modulo additions to achieve reversible data embedding. Goljan *et al.* [2] proposed a two cycles flipping permutation to assign a watermarking bit in each pixel group. Celik *et al.* [3] presented a high capacity, reversible data-embedding algorithm with low distortion by compressing quantisation residues. Tian [4] presented a reversible data embedding approach based on expanding the pixel value difference between neighbouring pixels, which will not overflow or underflow after expansion. Thodi and Rodrguez exploited the inherent correlation among the neighbouring pixels in an image region using a predictor. Xuan *et al.* [6] embedded data into high-frequency coefficients of integer wavelet transforms with the companding technique, and utilised histogram modification as a preprocessing step to prevent overflow or underflow caused by the modification of wavelet coefficients.

Celik *et al.* [7] presented a novel framework for lossless authentication watermarking by computing authentication information prior to reversible embedding steps. The advantage was the ability to reduce the computational requirement in situations when either the authentication failed or the reconstruction was not needed. Meanwhile, this framework also allowed for efficient tamper localisation using the hierarchical image authentication scheme. However, if the marked image was cropped, for example, the middle part of the marked image was left by cropping the edges way, the hierarchical relationship was destroyed. Thus, it was hardly to locate the cropped positions in Celik's method.

The above algorithms are still not satisfactory for some cases which need to resist some attacks, such as cropping attack. To solve the above problem, a block-based reversible watermarking scheme is presented in this paper. A locating pattern is introduced to keep track of the embedding position. With the help of the locating pattern, the correct cropped position for a cropped image can easily be detected. The locating pattern is embedded into the carrier image by a modified patchwork algorithm. Actually, it just makes modifications to the pixels at some fixed positions in each block. Incurred overhead, payload and authentication information are then collected and embedded into those unmodified pixel values by a difference expansion (DE) method. As image ID and block index serve as parts of input to the hash function, the scheme also can efficiently resist collage attack.

## 2 Relevant technologies

In this section, a brief introduction on the patchwork algorithm is given first, and this is followed by a subsection on modulo additions. Finally, the DE method is briefly explained.

### 2.1 Patchwork algorithm

In the patchwork algorithm [8], each group is equally divided into two pseudorandom sets of pixels, that is, patches $A$ and $B$. A given intensity $\delta$ is added to the sample values of patch $A$, whereas $\delta$ is subtracted from the sample values of patch $B$ as follows:

$$\alpha_i = a_i + \delta, \quad \beta_i = b_i - \delta \tag{1}$$

where $a_i$ and $b_i$ are sample values of patches $A$ and $B$, respectively.

The detection process starts with the subtraction of sample values between two patches. The expected value of $S'_n = \sum_{i=1}^{n}(\alpha_i - \beta_i)$ is used to decide whether the pixel contains watermark bits or not, where $n = |A| = |B|$, the operator $|\cdot|$ is the size of a set or the length of a sequence.



doi:10.1049/iet-ifs:20060091

Paper first received 14th June and in revised form 4th December 2006

S. W. Weng and Y. Zhao are with the Institute of Information Science, Beijing Jiaotong University, People's Republic of China

J.-S. Pan is with the Department of Electronic Engineering, Kaohsiung University of Applied Sciences, Taiwan

E-mail: wswweiwei@126.com

Note that the patchwork algorithm is based on the following assumption: the expected value of $S_n = \sum_{i=1}^{n}(a_i - b_i)$ is zero.

### 2.2 Modulo additions

Modulo additions [9] can be represented by the formula $i_w = i \oplus w = C\lfloor i/C \rfloor + \mathrm{mod}\,(i + w, C)$. The invertible subtraction is defined as: $i_w - w = i_w \oplus (-w)$, where $C$ is the cycle length, $i$ and $i_w$ represent a pixel value and its marked value, respectively, and $w$ is a watermark bit. The operator $\lfloor \cdot \rfloor$ rounds the value towards $-\infty$, and the operator $\oplus$ denotes modulo additions. For $C = 256$, modulo 256 addition can be presented by the following permutation: $0 \to 1, 1 \to 2, \ldots, 253 \to 254, 255 \to 0$ for the case $w = 1$.

### 2.3 DE method

For an 8-bit greyscale-valued pair $(p_x, p_y)$, the average $l$ and the difference $h$ are calculated according to (2)

$$l = \left\lfloor \frac{p_x + p_y}{2} \right\rfloor, \quad h = p_x - p_y \tag{2}$$

The inverse integer transform is as follows:

$$p_x = l + \left\lfloor \frac{h+1}{2} \right\rfloor, \quad p_y = l - \left\lfloor \frac{h}{2} \right\rfloor \tag{3}$$

To prevent possible overflows (>255) and underflows (<0) problems, that is, to restrict $p_x$, $p_y$ to be within the range of [0, 255], it is equivalent to have

$$|h| < \min\,(2(255 - l), 2l + 1) \tag{4}$$

$h$ is left shifted by one bit and then one bit $w \in \{0, 1\}$ is inserted into the LSB of $h'$. Mathematically, this is equivalent to

$$h' = 2 \times h + w \tag{5}$$

This reversible data-embedding operation is called DE. Each $h$ is classified to be in one of three categories: expandable set (ES), changeable set (CS), and non-changeable set (NS).

If the expanded difference value $h'$ given by (5) satisfies (4), $h$ is classified to be in set ES.

For each $h$ (not in ES), if its changed difference value $h'$ based on (6) satisfies inequality (4), $h$ is considered to be in CS.

$$h' = 2\left\lfloor \frac{h}{2} \right\rfloor + w \tag{6}$$

The rest of the $h'$ s belong to NS.

A location map is created by assigning '1' to $h$ in ES and '0' to the others. It is losslessly compressed as a part of embedded data. Thereby there are three parts of information that need to be embedded: (i) the bits from the compressed location map; (ii) the original LSBs of the changeable differences; and (iii) the payload.

Each $h$ in NS is kept unaltered during embedding. For each $h$ in CS, it can carry one bit. However, its LSB also needs to be embedded. Thereby $h$ in NS and CS makes no or little contribution to hiding capacity. It means that only $h$ in ES can carry an information bit without introducing any additional overhead. As a result, the hiding capacity is mainly determined by number of the expandable difference minus bit length of the compressed location map.

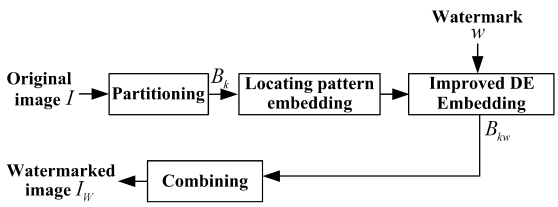


**Fig. 1** *Embedding process of the proposed reversible watermarking scheme*

## 3 Embedding process

The block diagram for the proposed reversible watermarking scheme is illustrated in Fig. 1. An 8-bit greyscale image $I$ with size $R \times C$ is partitioned into non-overlapping $r$-by-$c$ blocks $B_k$ $(1 \le k \le K)$, where $K = (R/r) \times (C/c)$.

### 3.1 Locating pattern

Similar to the method in [10], a zero matrix called the 'basic pattern' in this paper is produced with dimensions $(r, c)$. In the basic pattern, a pseudo-random number generator seeded with a user key is used to select two patches $A$ and $B$. Each patch accounts for a small portion (e.g. 50 points) of all $r \times c$ points. Each point value in patch $A$ is increased by $\delta$ whereas each point value in patch $B$ is decreased by $\delta$.

A $R \times C$ locating pattern $L$ is formed by tiling the basic pattern, that is, periodically replicating the basic pattern to the desired size. All the basic patterns in the pattern $L$ are denoted as $l_k$ $(1 \le k \le K)$. We set up one-to-one correspondence between each image block and each basic pattern in $x$ and $y$ coordinates.

For a point value $l_k(x, y)$ $(1 \le x \le r, 1 \le y \le c)$ in patch $A$, its corresponding pixel value $B_k(x, y)$ in the $k$th image block is combined in set $P_a^k$. For each point in patch $B$, its corresponding pixel belongs to set $P_b^k$. For one pixel value in $P_a^k$ or $P_b^k$, if it is in the interval $[\delta, 255 - \delta]$, it is added or subtracted by $\delta$ using modulo 256 addition. If not, it is kept unaltered. Meanwhile, a map called *pMap* is created by assigning '1' to all changed pixels and a '0' to the others in $P_s^k = P_a^k \cup P_b^k$.

For example, for a $512 \times 512$ image, if it is split into image blocks of size $32 \times 32$, there are $m = |A| \times K = 12\,800$ pixels in set $P_a = P_a^1 \cup P_a^2 \cup \cdots \cup P_a^K$ or $P_b = P_b^1 \cup P_b^2 \cup \cdots \cup P_b^K$. $m$ is large enough to ensure the expected value of $S_m = \sum_{i=1}^{m}(a_i - b_i)$ to be zero, where $a_i$ and $b_i$ are pixel values in sets $P_a$ and $P_b$, respectively. Note that the user key and $|A|$ need to be transmitted to the receiver.

### 3.2 DE embedding

DE embedding consists of two stages. Stage 1 is designed to embed the locating pattern $L$ described in Section 3.1.

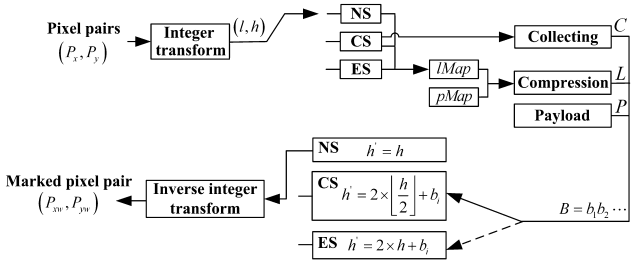


**Fig. 2** *DE embedding*

In Stage 2, DE embedding is as shown in Fig. 2. In the $k$th image block $B_k$, adjacent pixels in $\bar{P}_s^k$ are grouped into pairs in a predefined order. Integer transform (2) is applied to each pair to calculate the average values $l$ and the difference values $h$. Differences are classified into three disjoint sets, namely ES, CS and NS, described in Section 2.3.

A location map called *lMap* is created by assigning '1' to $h$ in ES and '0' to the others. *lMap* and *pMap* are concatenated into a bitstream and then losslessly compressed by an arithmetic encoder into bitstream $\mathcal{M}$. The original LSBs of all $h$ in CS are collected into bitstream $\mathcal{C}$.

Image ID, block index $k$ and all pixel values in $B_k$ serve as inputs to a hash function such as MD5 [11] to produce a 128-bit hash value. High 64 bits of this hash value are bitwise XORed with its low 64 bits to form a 64-bit hash value. And then it is combined with the binary representation of $k$ into bitstream $\mathcal{O}$.

Bitstream $\mathcal{M}$, bitstream $\mathcal{C}$, bitstream $\mathcal{O}$, and payload $\mathcal{P}$ are combined together into a bitstream $\mathcal{B}$. For ES, bits in $\mathcal{B}$ are embedded into the LSBs of $h'$ according to (5). For CS, the LSB of each $h$ is replaced with the bit in $B$ as described in (6). Finally, inverse integer transform (3) is applied to obtain the marked block $B_{kw}$.

## 4 Reversible watermarking extraction

The extraction framework of a reversible watermark is shown in Fig. 3.

At the receiving end, with the help of the received patch size and the secret key, each basic pattern is created by following the procedure discussed in Section 3.1.

$R'$ ($R' \le R$) and $C'$ ($C' \le C$) are used to denote the height and width of the given image, respectively. The height and width of the locating pattern $L'$ are denoted by $(\lceil R'/r \rceil + 1)r$ and $(\lceil C'/c \rceil + 1)c$. Accordingly, $N = (\lceil R'/r \rceil + 1) \times (\lceil C'/c \rceil + 1)$ basic patterns are presented in the pattern $L'$.

### 4.1 Stage 1: locating

Align the top-left pixel of the given image with the top-left point of the pattern $L'$. Translate the given image against the pattern $L'$. Test all translations of the given image against the pattern $L'$ such that the top-left corner is within the top-left block of the pattern $L'$.

At each translation, evaluate $S'_m = (\sum_{i=1}^{m} a_i - \sum_{i=1}^{m} b_i)$ for the overlapping region between the given image and the pattern $L'$. Determine the translation that produces the maximal $S'_m$, and denote the translation parameters by $(x'_1, y'_1)$. $(x'_1, y'_1)$ actually is the correct cropped position.

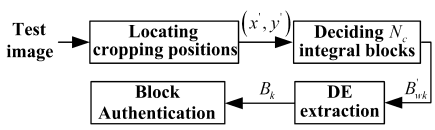


**Fig. 3** *Extraction process of the proposed scheme*

By $(x'_1, y'_1)$, the remaining $N_c$ integral blocks can be obtained.

### 4.2 Stage 2: decoding and authentication

For the $k$th marked image block $B'_{kw}$, $1 \le k \le N_c$, adjacent pixels in set $\bar{P}_s^k$ are grouped into pairs in the same way as in embedding. Integer transform (2) is applied to each pair to evaluate the average value $l_w$ and difference value $h_w$.

Inequality (4) is employed to unambiguously distinguish the changeable set from the nonchangeable set. If $h_w$ satisfies (4), $h_w$ is classified into the changeable set denoted by CH. Otherwise, it is in the nonchangeable set.

The LSBs of $h_w$ in CH are collected into bitstream $\mathcal{B}$. By identifying the end of message in $\mathcal{B}$, the bits from the start until the end of message are decompressed by an arithmetic decoder to retrieve the location maps, that is, *lMap* and *pMap*. For each $h_w$ in CH, if its location map value in *lMap* is 1, $h_w$ is classified into the set ES′. Otherwise, it belongs to set CS′. The bits after the end of message in $\mathcal{B}$ are recognised into bitstream $\mathcal{C}$, bitstream $\mathcal{O}$ and payload $\mathcal{P}$.

The original differences are restored as follows:

$$\begin{cases} h = \left\lfloor \dfrac{h_w}{2} \right\rfloor, & h_w \in \mathrm{ES}' \\ h = 2 \times \left\lfloor \dfrac{h_w}{2} \right\rfloor + b_1, & h_w \in \mathrm{CS}' \\ h = \lfloor h_w \rfloor, & h_w \in \mathrm{NS}' \end{cases} \tag{7}$$

where $b_1 \in \mathcal{C}$.

Finally, inverse integer transform (3) is applied to reconstruct the original pixel values.

For each pixel in $P_s^k$, if its location map value in *pMap* is 1, invertible subtraction is applied to restore its original value. If not, it remains intact. Finally, the whole original image block can be reconstructed.

As illustrated in Fig. 4, we compare the authentication hash in $\mathcal{O}$ with the hash of the retrieved image block. If they match, the image block is authentic. Otherwise, the image block is deemed non-authentic.

## 5 Resistance to collage attack

Fragile watermarking is designed to detect any tiny alternation to the pixel values. Hence, the tampered areas can accurately be located by checking the presence of fragile watermarking. However, blockwise fragile watermarking is vulnerable to collage attack, which assembles blocks of several authentic images or swaps blocks of the same image to forge a new authentic image.

Our scheme, a blockwise fragile watermark scheme, also suffers collage attack. In order to resist collage attack, image ID and block index are added to the inputs of the hash function, as in [12]. The disadvantage is that the receiver who wants to check the image must be informed of the exact ID of the image. The concrete process is illustrated in Fig. 4, where $\mathbf{H}(\cdot)$ is a hash function, the function **XOR** represents bitwise exclusive-or operation.

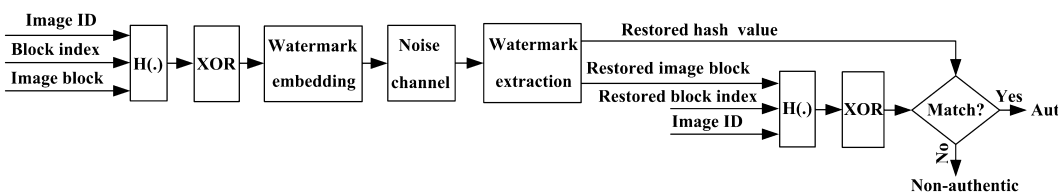


**Fig. 4** *Authentication process*


**Fig. 5** *Embedding and restoration without cropping attack*

*a* Original image
*b* Watermarked image without cropping attack
*c* Restored image

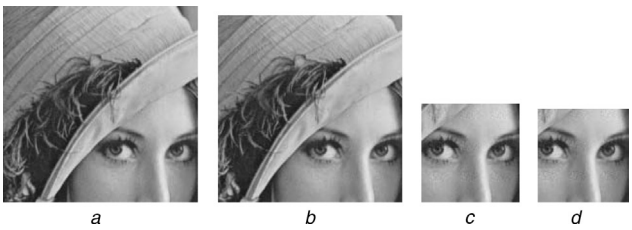


**Fig. 6** *Cropped images and corresponding restored images for 'Lena'*

*a* Watermarked image in the cropping region [120, 150, 320, 350]
*b* Restored image of *a*
*c* Watermarked image in the cropping region [220, 250, 320, 350]
*d* Restored image of *c*

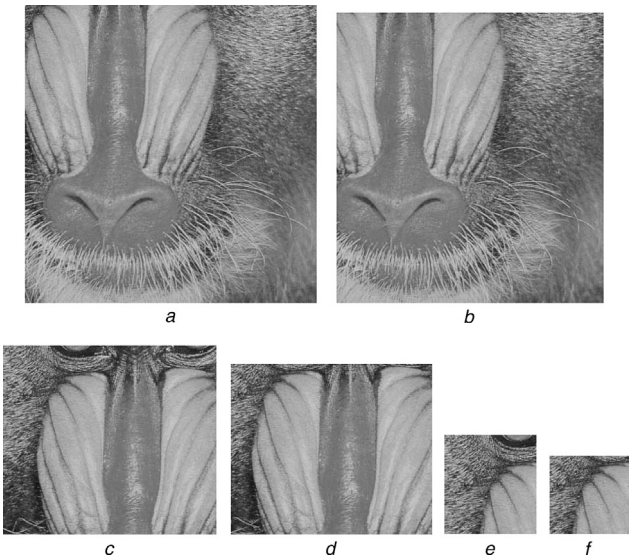


**Fig. 7** *Cropped images and corresponding restored images for 'Baboon'*

*a* Watermarked image in the cropping region [130, 120, 512, 512]
*b* Restored image of *a*
*c* Watermarked image in the cropping region [70, 80, 320, 360]
*d* Restored image blocks of *c*
*e* Watermarked image in the cropping region [70, 80, 200, 200]
*f* Restored image blocks of *e*

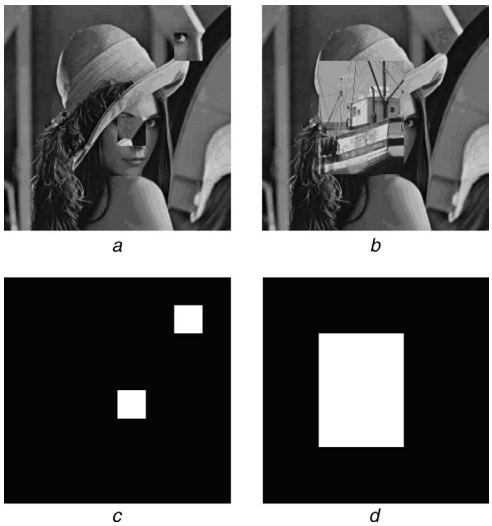


**Fig. 8** *Robustness against collage attack*

*a* Tampered image by swapping two blocks of the same image
*b* Tampered image by copying a portion of another image
*c* Result of fragile detection corresponding to *a*
*d* Result of fragile detection corresponding to *b*

## 6 Experimental results

We experimented with our method on several commonly used images, including 'Lena' and 'Baboon' of size $512 \times 512$. They are divided into $32 \times 32$ image blocks. A binary random sequence derived from a uniform noise was used as payload in our experiments. Unless otherwise noted, 50 point pairs are selected in each basic pattern.

As reported in Fig. 5, for the marked images without being cropped, the scheme can restore the original images without any distortion.

With the help of the locating pattern, our method is robust against random cropping. The cropping accuracy is determined by the $\delta$ and the patch size $|A|$ per block. At a fixed $|A|$ value, a $\delta$ value of 4 is sufficient to locate cropping positions for slight cropping attacks. For severe cropping attacks, $\delta$ needs to be increased; in general, $\delta = 8$ will work well.

For the watermarked 'Lena' image, under slight cropping attack, $\delta \leq 4$ is enough, for example, $\delta = 4$ is used in Fig. 6*a*. For those smaller cropped images containing just 2–3 integral blocks, $\delta$ is set to 8 as in Fig. 6*c*.

For the cropped image shown in Fig. 7*a*, $\delta = 3$ is used in the finding of the cropped position. However, $\delta$ needs to be updated because of different sizes of cropped images. For size $251 \times 281$ in Fig. 7*c*, $\delta = 4$ is set, whereas for size $131 \times 121$, $\delta$ is set to 6. But when 120 points are considered in patches, $\delta = 1$ will work well for size $131 \times 121$. In general, $\delta \leq 8$ can deal with most cropping attacks.

In order to resist the collage attack, we use the image ID and the block index as parts of the input to the hash function and the scheme performs well, as expected. Fig. 8 presents the robustness of our method against collage attack.

The PSNR values of the watermarked images are tabulated in Table 1, along with the embeddable payload size

**Table 1: Payload, overhead and PSNR under $\delta = 4$**

| Image | PSNR, dB | | Overhead, bit | Payload, bit | Overhead / payload, % |
|---|---|---|---|---|---|
| Lena | 39.8352 | 33.5759 | 27 143 | 91 129 | 0.2979 |
| Baboon | 35.3150 | 29.1978 | 26 846 | 91 426 | 0.2936 |

and overhead data. As reported in Table 1, there are still plenty of spaces to embed payload in addition to the overhead data. The PSNR values in the second column are generated just after embedding overhead, whereas in the third column, they are created after embedding overhead data and the payload.

## 7 Conclusion

With the introduction of the locating pattern, our method is robust against random cropping. Additional information incurred by the locating pattern is embedded into the host images using a DE method. The experimental results verify that cropping accuracy is determined by the $\delta$ and the patch size. By selecting the suitable parameters, our method can achieve a good robustness against cropping attack as well as collage attack.

## 8 Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (No. 90604032, No. 60373028), the Specialised Research Fund for the Doctoral Program of Higher Education, Program for New Century Excellent Talents in University, 973 Program (No. 2006CB303104) and the Specialised Research Foundation of BJTU.

## 9 References

1 Honsinger, C.W., Jones, P., Rabbani, M., and Stoffel, J.C.: 'Lossless recovery of an original image containing embedded data'. US patent no. 6278791, 2001

2 Goljan, M., Fridrich, J., and Du, R.: 'Distortion-free data embedding for images'. 4th Information Hiding Workshop, LNCS, vol. 2137, (Springer-Verlag, New York, 2001) pp. 27–41

3 Celik, M.U., Sharma, G., Tekalp, A.M., and Saber, E.: 'Reversible data hiding'. Proc. ICIP, 2002, vol. 2, pp. 157–160

4 Tian, J.: 'Reversible data embedding using a difference expansion', *IEEE Trans. Circuits Syst. Video Technol.*, 2003, **13**, (8), pp. 890–896

5 Thodi, M., and Rodrguez, J.J.: 'Prediction-error based reversible watermarking'. Proc. ICIP, Singapore, October 2004, vol. 3, pp. 1549–1552

6 Xuan, G.R., Yang, C.Y., Zhen, Y.Z., and Shi, Y.Q.: 'Reversible data hiding using integer wavelet transform and companding technique'. Proc. IWDW, 2004

7 Celik, M.U., Sharma, G., and Tekalp, A.M.: 'Lossless watermarking for image authentication: a new framework and an implementation', *IEEE Trans. Image Process.*, 2006, **15**, (4), pp. 1042–1049

8 Bender, W., Gruhi, D., and Morimoto, N.: 'Techniques for data hiding', *IBM Syst. J.*, 1996, **35**, (3–4), pp. 313–336

9 Fridrich, J., Goljan, J., and Du, R.: 'Invertible authentication', *Proc. SPIE*, 2001**, 3971**, pp. 197–208

10 Butman, M., and Hel-Or, H.Z.: 'Multi-level watermarking with independent decoding'. IEEE Proc. Int. Conf. Image Processing, 2001, vol. 2, pp. 514–517

11 Menezes, A., Van Oorchot, P., and Vanstone, S.: 'Handbook of applied cryptography' (CRC, Boca Raton, FL, 1997)

12 Holliman, M., and Memon, N.: 'Counterfeiting attacks on oblivious block-wise independent invisible watermarking schemes', *IEEE Trans. Image Process.*, 2002, **9**, (3), pp. 432–441