A REVERSIBLE WATERMARKING SCHEME FOR JPEG-2000 COMPRESSED IMAGES

Sabu Emmanuel
School of Computer Engineering
Nanyang Technological University
Singapore - 639798
asemmanuel@ntu.edu.sg

Heng Chee Kiang
Global Software Group
Motorola Electronic Pte Ltd
Singapore – 569087
ckheng@motorola.com

Amitabha Das
School of Computer Engineering
Nanyang Technological University
Singapore - 639798
asadas@ntu.edu.sg

ABSTRACT

In this paper, we present a novel reversible watermarking scheme for image authentication for JPEG/JPEG-2000 coded images. Since the watermarking scheme is reversible, the exact original image can be recovered from the watermarked image. The watermarking scheme makes use of finite state machine principles. The proposed scheme is asymmetric as the watermark extraction key is different than its embedding key. The algorithm is implemented and tested for its visual quality, compression overhead, execution time overhead and payload capacity. It is found that the algorithm has high visual quality, high payload capacity, low compression overhead and low execution time overhead.

Keywords: Reversible Watermarking, Image Authentication, JPEG-2000, Finite State Machine Based Watermarking

1. INTRODUCTION

Media such as image, video, audio etc. are increasingly becoming digital due to the easiness in processing, storage and transmission. For certain applications of digital media such as for legal evidence, digital commerce, security surveillance etc. any modification/tampering done to media has to be detected. Fragile or semi-fragile watermarking techniques can be employed to detect tampering and thus provide media authentication function. Watermarking techniques embed a watermark in the media by modifying the media data. Often these modifications are irreversible, i.e. the original data may not be able to be recovered from the watermarked data even after extracting the watermark. This irreversible modification may not be admissible especially for legal evidence and surveillance applications. Therefore these applications require reversible watermarking techniques that support reversibility in addition to fragility or semi-fragility. In this paper we are concerned with the reversible watermarking of digital images, so further discussion applies to digital images.

Several reversible watermarking techniques are proposed in the literature. Works by Fridrich et.al. [3][4] and Celik et.al. [2] need to compress the original features and transmit the compressed bit-stream as a part of the embedded payload in order to make the watermarking reversible. Tian [8] and Alattar [1] proposed a difference expansion transform-based reversible watermarking, which does not require compressed original features for supporting reversibility. However, these schemes may suffer from low embedding capacity. Macq [5] devised a reversible watermarking technique using reversible addition. But this scheme suffers from salt-and-pepper visual artifact. Vleeschouwer et.al. [9] try to reduce the salt-and-pepper artifact by using a circular interpretation of bijective transforms for reversible watermarking. Some of the recent works involve the use of JPEG/JPEG-2000 domain for watermarking due to the popularity of the coding scheme. Fridrich et. al. [4] propose an invertible authentication watermark for JPEG Images. The scheme embeds a message authentication code in quantized discrete cosine transform (DCT) coefficients. Whereas Sun et. al. [6] propose a semi-fragile watermark for JPEG-2000 compressed images which are not reversible.

In this paper we propose a reversible, fragile, asymmetric watermarking scheme for JPEG/JPEG-2000 coded images. In section 2 we present the proposed algorithm, in section 3 results and discussion and in section 4 conclusion.

2. THE PROPOSED ALGORITHM

The proposed watermarking algorithm is a fragile watermarking algorithm, which can be implemented in JPEG/JPEG-2000 codec. For JPEG, watermarking will be performed on quantized DCT coefficients. But in JPEG-2000 [7], the watermarking will be performed on quantized/ranged DWT coefficients. For both JPEG-2000 and JPEG, the DC values will not be modified so as to maintain high visual quality. For color images, we embed the watermark only in the Y component of the image. We further discuss the algorithm with respect to JPEG-2000 implementation only.

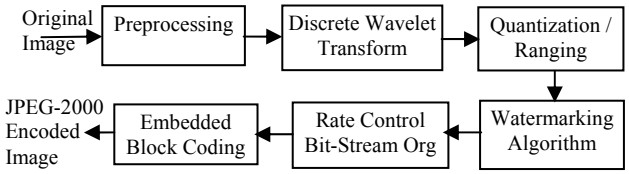

Figure (1): Location of Watermarking Algorithm in JPEG-2000 Coding Pipeline.

Figure (1) depicts the location of watermark embedding algorithm in the JPEG-2000 coding pipeline [7]. The discrete wavelet transform (DWT) in JPEG-2000 decomposes input image bit stream into a collection of subbands, which may be organized into increasing resolution levels, L_0, L_1, L_2, etc. The lowest resolution level L_0 consists of only LL subband whereas other resolution levels L_1, L_2, etc. consist of LH, HL, HH subbands. The DWT coefficients are then either quantized or ranged, as can be seen in figure (1). Ranging is preferred for lossless

compression whereas for lossy compression, quantization is preferred. The quantized/ranged subband coefficients to be watermarked are then fed to the watermark embedding algorithm.

2.1 Watermark Embedding Algorithm

We consider the subband coefficients (to be watermarked) as a 1-dimentional signal obtained by raster scanning (scanning from left to right and then top to bottom) the 2-dimentional subband coefficients. Let the 1-dimentional subband coefficients be denoted as $\boldsymbol{X}$. Let us select N coefficients from $\boldsymbol{X}$ at a time in a non-overlapping manner to form non-overlapping segments $\boldsymbol{X}_r$. Thus,

$$X_r(k) = \begin{cases} X(k+rN) & 0 \le k \le (N-1) \\ 0 & otherwise \end{cases} \quad (1)$$

Next we compute an N-bit state for each segment $\boldsymbol{X}_r$ of length N. In order to create an N-bit state, we will first need to determine the threshold value, A which is selected a random value biased towards the average of the subband coefficients being watermarked. Next we compute the N-bit state $\boldsymbol{BM}_r$ for the segment $\boldsymbol{X}_r$ using the expression

$$BM_r(k) = \begin{cases} 0 & A > X_r(k), \quad 0 \le k \le (N-1) \\ 1 & A \le X_r(k), \quad 0 \le k \le (N-1) \end{cases} \quad (2)$$

To embed a single watermark bit into $\boldsymbol{X}_r$ (of length N), the algorithm employs a state group table (SGT). Since there are N bits representing a state, there are 2^N distinct states possible in the SGT. We group these 2^N states into two groups. A random distinct 2^{N-1} states form a state group 0 (SG_0), which represents a watermark embedded bit 0 and the rest 2^{N-1} states form a state group 1 (SG_1), which represents a watermark embedded bit 1. The states in SG_0 and SG_1 are to be kept secret and hence the composition of SGT forms part of the key for embedding and extraction.

In order to embed a watermark bit 0 in $\boldsymbol{X}_r$, the N-bit state $\boldsymbol{BM}_r$ is computed which is then matched with the entries in SG_0. If $\boldsymbol{BM}_r$ is present in SG_0 the watermark bit 0 is considered embedded. Therefore no modification is needed on $\boldsymbol{X}_r$. But if $\boldsymbol{BM}_r$ is not present in SG_0, modification is needed on $\boldsymbol{X}_r$. The modification on $\boldsymbol{X}_r$ is done in such a way that the corresponding $\boldsymbol{BM}_r$ is matched to a state in SG_0.

Similarly, in order to embed a watermark bit 1, if the $\boldsymbol{BM}_r$ is present in SG_1 no modification is needed on $\boldsymbol{X}_r$, and if $\boldsymbol{BM}_r$ is not present in SG_1 modification is needed on $\boldsymbol{X}_r$. The modification on $\boldsymbol{X}_r$ is done in such a way that the corresponding $\boldsymbol{BM}_r$ is matched to a state in SG_1.

The finite state transition table (FSTT) specifies the new state into which the $\boldsymbol{X}_r$ would transit so that the current watermark bit gets embedded. The schema of the FSTT table is FSTT(*present state*, *watermark bit to embed*, *next state*). The entries in the FSTT are designed in such a way that a transition must cause $\boldsymbol{BM}_r$ to match the correct SG group. The proposed algorithm uses three kinds of modification actions namely, addition, subtraction and swapping. The operation addition increases one or more coefficients (k^{th} in $\boldsymbol{X}_r$ by an amount d_r^k). The operation swapping swaps two coefficients in $\boldsymbol{X}_r$. And the subtraction decreases one or more coefficients (k^{th} in $\boldsymbol{X}_r$ by d_r^k). The d_r^k is determined by the following expression:

$$d_r^k = |A - X_r(k)| \qquad 0 \le k \le N-1 \quad (3)$$

The algorithm will decide one of the three execution path, which will determine whether to swap the coefficients, perform arithmetic operations (addition and subtraction) on the coefficients, or simply do nothing. The algorithm takes $\boldsymbol{BM_r}$ (the present state) and $\boldsymbol{BM_r'}$ (the next state) from FSTT and carry out the XOR ($\oplus$) to obtain $\boldsymbol{SA_r}$, using the expression:

$$SA_r(k) = BM_r(k) \oplus BM_r'(k) \quad 0 \le k \le N-1 \quad (4)$$

Once $\boldsymbol{SA_r}$ is determined, the algorithm will use the expression below to decide whether swapping, arithmetic operations or no modification action should be taken. Thus no additional information needs to be communicated to the detector to identify the operation used. The symbol '$\wedge$' represents AND operation.

$$\textit{modificiation action} = \begin{cases} \textit{swap} & \left(\sum_{k=0}^{N-1} SA_r(k) \textit{ is even and} \neq 0\right) \wedge \left(\sum_{k=0}^{N-1} BM_r(k) == \sum_{k=0}^{N-1} BM_r'(k)\right) \\ \textit{no modification} & \sum_{k=0}^{N-1} SA_r(k) == 0 \\ \textit{arithmetic operation} & \textit{otherwise} \end{cases} \quad (5)$$

Swap operation perform an exchange between a pair of coefficients in $\boldsymbol{X_r}$ whereby the affected positions for swapping are stated in $\boldsymbol{SA_r}$. The algorithm for performing the swap operation on $\boldsymbol{X_r}$ will use $\boldsymbol{SA_r}$ and $\boldsymbol{BM_r}$ to determine which coefficients will be swapped. The algorithm for performing the swapping operation is:

```
let the position p be i where i = 0, 1, ..., N-1 the first SA_r(i) == 1
while not all coefficients in the segment rth are swapped
    for i := 0 to N-1 do
      if  ((BM_r[i] != BM_r[p]) and (SA_r[i] == 1) and (SA_r[p] == 1)) then
          swap X_r[i] and X_r[p]
          set SA_r[i] and SA_r[p] as 0
          break for loop
      endif
    endfor
    let the position p be i where i = 0, 1, ..., N-1 the first SA_r(i) == 1
endwhile
```

No modification implies that the coefficients are not modified in any way. The coefficients are assumed to hold the embedded watermark bit. Arithmetic operations perform addition and subtraction to the coefficients in $\boldsymbol{X_r}$ using the expression below:

$$X_r^w(k) = \begin{cases} X_r(k) + 2d_r^k & BM_r'(k) > BM_r(k) \\ X_r(k) - 2d_r^k & BM_r'(k) < BM_r(k) \\ X_r(k) & BM_r'(k) == BM_r(k) \end{cases} \qquad 0 \le k \le N-1 \quad (6)$$

$\boldsymbol{X^w}$ is constructed by appending each $\boldsymbol{X_r^w}$ segments.

The embedding key k consists of SGT, FSTT, A, N, L_n, the embedded subband name HL/LH. The FSTT needs to be kept secret by the seller for non-reversible watermarking. Thus for tamper detection/authentication, the watermark extraction key k^* would contain only SGT, A, N, L_n, the

embedded subband name HL/LH. But for reversible watermarking the key needed to reverse the watermark is *k'*, which contains FSTT, *A*, *N*, L_n, the embedded subband name HL/LH. Since SGT can be derived from FSTT, the *k'* can be used for extracting the watermark as well. In addition the current state and the next state in FSTT should possess a one to one mapping property for reversible watermarking. But for non-reversible watermarking the property can be many to one mapping as well between current state and next state.

2.2 Tamper Detection Process

The tamper detection process use a watermark extraction key, *k**, which is different than the embedding key, *k*. The extraction key, *k** is consisting of SGT, *A*, *N*, L_n and the embedded subband name HL/LH. Firstly, the non-overlapping (watermarked subband) segments $\boldsymbol{X}_r^w$ are extracted from $\boldsymbol{X}^w$ using the parameters *N*, L_n and the embedded subband name HL/LH in *k** and the equation (1). The *N*-bit state $\boldsymbol{BM}_r'$ is then computed from $\boldsymbol{X}_r^w$ using the parameter *A* in *k** and the equation (2). The *N*-bit state $\boldsymbol{BM}_r'$ is then used to look-up the SGT to extract the watermark bit embedded in the $\boldsymbol{X}_r^w$. The extracted watermark bits are organized into a vector, $\boldsymbol{W}'$, which is then compared against the original embedded watermark vector, $\boldsymbol{W}$, to determine whether modifications have been made to the watermarked image. Let the length of $\boldsymbol{W}$ and $\boldsymbol{W}'$ be *l*. Using the expression below:

$$c = \sum_{i=0}^{l-1} \left(\begin{cases} 0, & W(i) == W'(i) \\ 1, & W(i) <> W'(i) \end{cases} \right) \tag{7}$$

If *c* is non-zero, it is declared that the watermarked image has been modified/tampered.

2.3 Watermark Reversibility Algorithm

The reversibility algorithm requires the key *k'*, which contains FSTT, *A*, *N*, L_n, the embedded subband name HL/LH. Firstly, the non-overlapping segments $\boldsymbol{X}_r^w$ are extracted from $\boldsymbol{X}^w$ using the parameters *N*, L_n and the embedded subband name HL/LH in *k'* and the equation (1). The *N*-bit state $\boldsymbol{BM}_r'$ is then computed from $\boldsymbol{X}_r^w$ using the parameter *A* in *k'* and the equation (2). Using $\boldsymbol{BM}_r'$, FSTT and reverse modification action, we can revert $\boldsymbol{X}^w$ to $\boldsymbol{X}$. Using the r^{th} segment $\boldsymbol{BM}_r'$ and its corresponding $\boldsymbol{BM}_r$ from FSTT, we can find out the reverse modification actions to convert $\boldsymbol{X}_r^w$ to $\boldsymbol{X}_r$. There are three types of reverse modification actions, swapping, arithmetic operations and no modification. To determine which reverse modification action is appropriate, the algorithm, first, calculates $\boldsymbol{SA}_r'$ using the expression below.

$$\mathrm{SA_r}'(k) = \mathrm{BM_r}'(k) \oplus \mathrm{BM_r}(k) \quad 0 \le k \le N-1 \tag{8}$$

Once $\boldsymbol{SA}_r'$ is computed, the algorithm uses the expression below to find out the appropriate reverse modification action, with $\wedge$ taken to represent AND operation:

$$\text{reverse modification action} = \begin{cases} \text{swap} & \left(\sum_{k=0}^{N-1} SA_r'(k) \text{ is even and} \neq 0\right) \wedge \left(\sum_{k=0}^{N-1} BM_r'(k) == \sum_{k=0}^{N-1} M_r(k)\right) \\ \text{no reverse modification} & \sum_{k=0}^{N-1} SA_r'(k) == 0 \\ \text{arithmetic operation} & \text{otherwise} \end{cases} \tag{9}$$

Swap operation performs an exchange between a pair of coefficients in $\boldsymbol{X}_r^w$. The algorithm for performing the swap operation on $\boldsymbol{X}_r^w$ will use $\boldsymbol{SA}_r'$ and $\boldsymbol{BM}_r'$ to determine which coefficients will be swapped. The algorithm for performing the swapping operation is:

```
let the position p be i where i = 0, 1, ..., N-1 the first SA_r'(i) == 1
while not all coefficients are swapped
    for i := 0 to N-1 do
      if (BM_r'[i] <> BM_r'[p]) and (SA_r'[i] == 1) and (SA_r'[p] == 1) then
        swap X_r[i] and X_r[p]
        set SA_r'[i] and SA_r'[p] as 0
        break for loop
      endif
    endfor
    let the position p be i where i = 0, 1, ..., N-1 the first SA_r'(i) == 1
endwhile
```

No reverse modification implies that the coefficients need not be reverse modified since the watermark coefficients are the same as the original coefficients.

Arithmetic operations perform addition and subtraction to the coefficients in $\boldsymbol{X}_r^w$ using the expression below:

$$\mathrm{X_r}(k) = \begin{cases} \mathrm{X_r^w}(k) - 2d_r^{wk} & \mathrm{BM_r'}(k) > \mathrm{M_r}(k) \\ \mathrm{X_k^w}(k) + 2d_r^{wk} & \mathrm{BM_r'}(k) < \mathrm{M_r}(k) \\ \mathrm{X_r^w}(k) & \mathrm{BM_r'}(k) == \mathrm{M_r}(k) \end{cases} \quad 0 \le k \le N-1 \tag{10}$$

where d_r^{wk} is computed using the expression:

$$d_r^{wk} = \left| A - X_r^w(k) \right| \quad 0 \le k \le N-1 \tag{11}$$

We carry out the reversibility operation segment-wise. When all the segments in the embedded subband in L_n are processed, all the coefficients in $\boldsymbol{X}^w$ will be successfully restored to $\boldsymbol{X}$.

3. RESULTS & DISCUSSION

Several images were tested using three testing procedures, which attempted to determine the quantitative measure of degradation, compression overhead and execution time profile.

The quantitative measure of degradation was based on objective analysis using Peak Signal-to-Noise Ratio (PSNR), $PSNRindB = 20 \log_{10}\left(255 \Big/ \left(\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M}[\mathrm{X}(i,j) - \mathrm{X^w}(i,j)]^2 \Big/ N \times M}\right)\right)$ where $\boldsymbol{X}$ is the original wavelet coefficients, $\boldsymbol{X}^w$ is the watermarked wavelet coefficients, *N* and *M* are the height and width of the image respectively.

Figure (2)(a) shows the plot of PSNR in dB versus resolution level. And it is noted that the PSNR improves as the embedded region moves towards higher resolution levels. This is expected as the higher resolution levels contribute less, details to the overall image.

The compression overhead assessment was based on encoded file size and uses the following equation

$$\begin{aligned} &\textit{compression overhead in percentage} \\ &= \left[\frac{\textit{size of watermarked JPEG} - \textit{2000 compressed image}}{\textit{size of original JPEG} - \textit{2000 compressed image}} - 1 \right] \times 100 \end{aligned} \tag{12}$$

Figure (2)(b) plots the maximum compression overhead versus *N* for L_3 at HL subband and can be found that it does not exceed 8%. This implied that the embedding process did not affect the compression output in a big way.

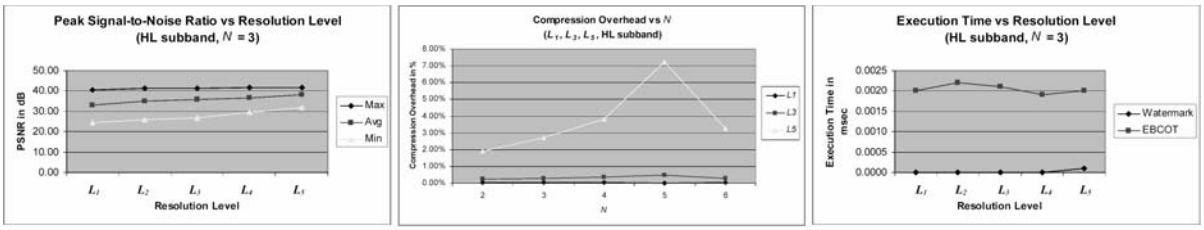

Figure (2): Plots of PSNR, Compression Overhead and Execution Time

Since EBCOT being the most time-consuming algorithm in JPEG-2000 codec, we carry out the execution time profile of the proposed watermarking algorithm against that of EBCOT. Figure (2)(c) illustrates the graph of Execution Time versus Resolution Level for HL Subband and $N = 3$. The execution time for the watermarking algorithm and EBCOT is collected using the native C runtime function, clock(). The graph demonstrates that the embedding algorithm does not add any significant execution time overhead to the codec relative to EBCOT. Since EBCOT takes four passes per bit planes, and with k-bit planes per coefficients, the EBCOT will be k x 4 times slower compared to coefficient-wise single pass algorithms. The proposed watermarking algorithm is coefficient-wise single pass algorithm.

The payload capacity of proposed watermarking algorithm for an image dimension of 800x600, at resolution level 3, L_3, HL subband with varying N is shown in table (1). The payload capacity would be higher if we use higher resolution level. For example, if we use L_6 instead of L_3 the payload capacity would be 30,000 at $N = 2$.

N	Payload Capacity in Bits	Embedding Key (*k*) Size in Bits	Extraction Key (*k**) Size in Bits	Reversion Key (k') Size in Bits
2	3750	80	40	72
3	2500	168	56	144
4	1875	384	96	320
5	1500	896	192	736
6	1250	2080	416	1696

Table (1): Payload Capacity and Key Sizes

The table (1) also shows the embedding key, extraction key and reversion key size in bits versus N. The size is computed assuming A, N, L_n and the name of the embedded subband (LH/HL) taking 8-bits each and the SGT, FSTT is calculated in bits. The sizes are small compared to the image size. Since the watermarking is done in the wavelet coefficient domain, the effect of watermark spreads on all the pixels in spatial domain. An attacker can attack this authentication by swapping the entire watermarked subband coefficients with another identically watermarked subband. To defeat this attack, the watermark embedder can use signed message (signed using the private key of embedder) consisting of embedding subband name, resolution level and image index as watermark ***W***. The image index is then communicated to the buyer along with the watermarked image. At the buyer side, the watermark ***W'*** **(**which is the signed message) is extracted first. The signature verification is done on the extracted signed message using the public key of embedder to obtain embedding subband name, resolution level and image index which is then compared with the communicated image index, and the resolution level and subband used for extraction of ***W'***. If they are equal the image is authenticated.

4. CONCLUSION

We have proposed a novel reversible (lossless) watermarking scheme using the concept of state tables and state transition. As a single-pass algorithm, it is fast and is also able to achieve high visual quality in terms of PSNR. Its high payload allows the seller to embed enough information to describe the image and include any extra information so required. Furthermore, the addition of watermark has only small effect on the compression ratio. The algorithm allows the extraction of the messages with a different extraction key than the embedding key (asymmetric key system) and it employs multiple modification actions to carry out the watermarking.

Acknowledgements: This research is supported by Singapore A*STAR SERC Grant (032 101 0006).

5. REFERENCES

[1] Alattar A.M. *Reversible Watermark Using the Difference Expansion of a Generalized Integer Transform*, IEEE Transactions on Image Processing, Vol. 13, No. 8, Aug. 2004.

[2] Celik M.U, Sharma G., Tekalp A.M., and Saber E. *Reversible Data Hiding*, Proc. ICIP 2002, Rochester, USA, Sept. 2002.

[3] Fridrich J., Goljan M. and Du R. *Invertible Authentication*, Proc. SPIE, Security and Watermarking of Multimedia Contents , San Jose, USA, Jan. 2001.

[4] Fridrich J., Goljan M. and Du R. *Invertible Authentication Watermark for JPEG Images*, Proc. SPIE, Security and Watermarking of Multimedia Contents, 2001, pp. 197 – 208.

[5] Macq B. *Lossless Multiresolution Transform for Image Authenticating Watermarking*, Proc. EUSIPCO, Tampere, Finland, Sept. 2000.

[6] Sun Q., Chang S., Kurato M. and Suto M. *A Quantitive Semi-Fragile JPEG2000 Image Authentication System*, Proc. of ICIP2002, Rochester, USA, 2002.

[7] Taubman D., Michael W. and Marcellin. *JPEG 2000 Image Compression Fundamental, Standards and Practice*, Kluwer Academic Publishers, Boston, 2002.

[8] Tian J. *Reversible Watermarking by Difference Expansion*, Proc. Workshop on Multimedia and Security at ACM MM 2002, Juan-les-Pins, France, Dec. 2002.

[9] Vleeschouwer C.De, Delaigle J.F., and Macq B. *Circular Interpretation of Histogram for Reversible Watermarking*, Proc. IEEE 4th Workshop Multimedia Signal Processing, Oct. 2001.