Block Size Analysis for Discrete Wavelet Watermarking and Embedding a Vector Image as a Watermark

Hayri Sever[1], Ahmet Şenol[1], and Ersin Elbaşı[2]
[1]Department of Computer Engineering, Çankaya University, 06790 Etimesgut Ankara
[2]College of Engineering and Technology, American University of the Middle East, Kuwait

Abstract: *As telecommunication and computer technologies proliferate, most data are stored and transferred in digital format. Content owners, therefore, are searching for new technologies to protect copyrighted products in digital form. Image watermarking emerged as a technique for protecting image copyrights. Early studies on image watermarking used the pixel domain whereas modern watermarking methods convert a pixel based image to another domain and embed a watermark in the transform domain. This study aims to use, Block Discrete Wavelet Transform (BDWT) as the transform domain for embedding and extracting watermarks. This study consists of 2 parts. The first part investigates the effect of dividing an image into non-overlapping blocks and transforming each image block to a DWT domain, independently. Then, effect of block size on watermark success and, how it is related to block size, are analyzed. The second part investigates embedding a vector image logo as a watermark. Vector images consist of geometric objects such as lines, circles and splines. Unlike pixel-based images, vector images do not lose quality due to scaling. Vector watermarks deteriorate very easily if the watermarked image is processed, such as compression or filtering. Special care must be taken when the embedded watermark is a vector image, such as adjusting the watermark strength or distributing the watermark data into the image. The relative importance of watermark data must be taken into account. To the best of our knowledge this study is the first to use a vector image as a watermark embedded in a host image.*

Keywords: *Watermarking, DWT, block, vector, SVG.*

Received Febrewary 7, 2017; accepted October 17, 2018

1. Introduction

The quantity of digital data has increased enormously in the last decade, as digital camera image quality has increased and the cost of such devices has decreased. The greatest increase in stored digital data has come as a result of cameras being embedded in smart phones. As internet bandwidth increases, web sites are also increasing the quantity and quality of their image and video content. Due to the massive volume of traffic, people must protect their digital property against theft and unauthorized use. Before the advent of watermarking, visible copyright signs were placed on images and cryptographic methods were used for such protection.

A disadvantage of encrypting images while transferring to a target site is that the image is completely defenceless following decryption at the target site. Placing a visible copyright sign on an image is minimally effective protection, as the copyright sign and script can be easily removed. Image watermarking for proving digital data ownership has emerged as a new technology to compensate for the above mentioned drawbacks. An image to be copyrighted is referred to as the host image. A watermark is embedded into a host image in a way that it is not detectable by the human eye; it remains embedded in t he host image as long as the image exists. Generally it is not possible to remove a watermark from a host image without diminishing image quality to a considerable extent. A watermark can be a Pseudo Random Number Sequence (PRNS) as in [4, 15, 18], a binary image logo as in [1, 2, 5, 8, 9, 11, 13, 21, 23], a gray scale logo as in [12, 17], a color image logo as in [3], biometric data such as the owner's voice, a Gabor face as in [7], or a QR code as in [10].

Fidelity is the most important criterion for measuring watermarking success. Fidelity is the level of similarity between an original and watermarked image. Following watermarking, it should not be detectable that the image has been processed. Fidelity is calculated as the PSNR, as in Equation (1) in which Rounded Mean Square Error (RMSE) is given by Equation (2). I is the original image, I^* is the watermarked image and i, j are the pixel coordinates.

$$PSNR = 20\ log_{10}(255 / RMSE) \quad (1)$$

$$RMSE = sqrt((\sum_{i,j} (I^*_{ij} - I_{ij})^2) / (N \times N)) \quad (2)$$

Robustness is another criterion for measuring the success of watermarking methods that aim to prove ownership. After an original image is watermarked, it can be modified by normal image processing such as cropping, blurring and sharpening. Such modifications

can be performed to remove or destroy a watermark. Watermarking must be performed in such a way that the watermark can be extracted from the watermarked image despite the image having been modified. Furthermore, modifications to a watermarked image should not be obvious.

A watermarking algorithm is called non-blind if the original image is needed for extracting the watermark, whereas blind type if original image is not needed.

In early watermarking studies, watermarks were embedded in the Least Significant Bits (LSB) [22]. When the LSBs of a grayscale image were changed the overall appearance of the image did not change obviously. LSB watermarking was eventually replaced by transforming the cover image to another domain, adding a watermark to transform domain values and then applying inverse transform. Cox *et al*. [4] transformed a cover image into a Discrete Cosine Transform (DCT) domain and added a PRNS with mean zero as watermark to the highest K coefficients, except that the DC component where K is the PRNS length.

Piva *et al*. [15] also used DCT-based watermarking to embed a watermark PRNS again in the highest M coefficients by skipping the L number of coefficients, so as to show that the watermarking method does not suffer from high pass filters, gamma correction, etc.

The correlation between the original and the extracted watermark can be computed as in Equation (3) where *W* is the original watermark, *W** is the extracted watermark and *M* is the watermark size.

$$Z = (W \times W^*) / M = \frac{1}{M}\sum_{i=1}^{M} W_i . W^*_i \qquad (3)$$

Piva *et al*. [15] calculated threshold Tz as shown in Equation (4), where $\alpha = 0.2$ in the paper. If $Z > Tz$, then it was decided that a watermark was present.

$$Tz = \frac{\alpha}{3M}\sum_{i=1}^{M} |W^*_i| \,, \alpha = 0.2 \qquad (4)$$

Tao and Eskicioğlu [21] embedded a binary image logo in 4 of the bands (LL, LH, HL, HH) of DWT decomposition of the cover image. Their embedding and extraction values are given in Equation (5) and Equation (6) respectively, where original image is size 2N×2N, k is the band number in {1, 2, 3, 4}, W is N×N binary logo watermark image, V^k_{ij} are DWT coefficients in the band k, $V^k{}'_{ij}$ are watermarked DWT coefficients, V^{*k}_{ij} are watermarked-and-probably-changed-image's DWT coefficients in band k, and *W** is an extracted watermark.

$$V^k{}'_{ij} = V^k_{ij} + \alpha_k W_{ij} \qquad (5)$$

$$W^*_{ij} = (V^{k*}_{ij} - V^k_{ij}) / \alpha_k \qquad (6)$$

The watermark strength constant α is 2 in bands LH, HL and HH, whereas in LL band α is used as value 8, as LL band coefficients are larger in magnitude. Larger magnitude coefficients provide greater watermark holding capacity.

Other DWT-based watermarking methods were subsequently proposed [2, 5, 9, 12, 14, 16, 21]. Lai and Tsai [12] applied DWT to an image, and then embedded a watermark into the singular values of the Singular Value Decomposition (SVD) of bands LL, LH, HL, and HH.

This paper contains two studies that have common parts and also have different aims and contributions. Section 2 has the purpose of clarifying the impact of dividing the host image into blocks in DWT domain. How the block size affects the success of watermarking is analysed. In section 3, a method is developed to embed a vector image as watermark to a host image. Vector images have RGB colour info which is superior to black-white binary images. Vector images also do not lose quality when resized as opposed to pixel based binary images.

2. Block Size Analysis for Block Discrete Wavelet Watermarking

For this part of the study the Tao and Eskicioğlu's [21] method was used, except for blocking. The cover image and binary watermark image were split into blocks before DWT, and each cover image block was transformed to DWT separately. Each block was watermarked with a corresponding watermark block. Watermark blocks were 25% of the size of the cover blocks because of the nature of DWT. The algorithm was run by no-blocks [21], at block sizes of 64×64, 32×32, 16×16 and 8×8 [19]. PSNR values for watermarked images were calculated using Equation (1) and Similarity Ratio (SR) values between embedded and extracted watermarks were calculated using Equation (7). *S* is the number of matching pixels between the embedded and extracted watermark, and *D* is the number of different pixels between them.

$$SR = S / (S + D) \qquad (7)$$

2.1. Experiments

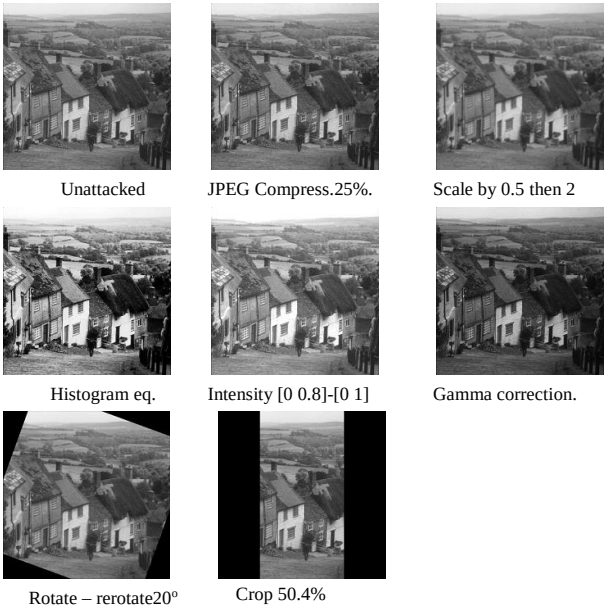
Image block sizes of 512×512 (full image), 64×64, 32×32, 16×16 and 8×8 were included in the experiment. The original image, the binary watermark image and second watermark image used for re-watermark attack are shown in Figure 1. The watermarked and attacked images are shown in Figure 2 in the case of an original image that was divided into 64×64 image blocks for some attack types. PSNR values are given for each case. Common attacks were applied to watermarked images.

Extracted watermarks from each DWT band and corresponding SR values for each 'attack type'-'block size' combination are given in Tables 1-3. The numbers given under extracted watermarks are SR values. Extracted watermark quality and, hence, the SR value increased as block size decreased. SR values obtained in the present study for LL band are shown in Table 4

with a comparison to those reported by [21]. Only 8x8 block size results are listed for LH, HL, HH bands. As parameters for some attack types were not reported by Tao *et al*. [21] such as the scaling factor or cropping ratio, direct comparison is not possible. It is clearly seen that block-DWT watermarking yielded better results than the non-block condition.

a) Original image. b Binary img watermark. c) 2nd watermark.

Figure 1. The original image & watermark images used in the study.



Unattacked JPEG Compress.25%. Scale by 0.5 then 2

Histogram eq. Intensity [0 0.8]-[0 1] Gamma correction.

Rotate – rerotate20° Crop 50.4%

Figure 2. Un-attacked Watermarked and attacked images for 8 type of attacks.s

Table 1. Extracted watermarks for various block sizes in lossy JPEG compress attacks and Histogram Eq.

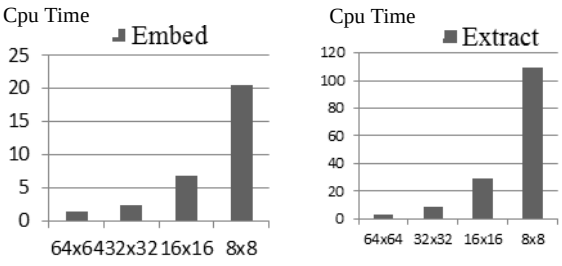
	JPEG 75% Q		JPEG 25% Q		Histogram Eq.	
	LL	LH	HL	HH	LL	LH
No Blocks	0.8976	0.4534	0.4540	0.4138	0.6269	0.6355
64×64 Blocks	0.8976	0.4534	0.4540	0.4138	0.6269	0.6355
32×32 Blocks	0.9048	0.4810	0.4847	0.4411	0.6639	0.6637
16×16 Blocks	0.9119	0.5063	0.5129	0.4679	0.6996	0.6916
8×8 Blocks	0.9156	0.5204	0.5287	0.4828	0.7177	0.7059

Table 2. Extracted watermarks for intensity adjustment, gamma correction and rotate attacks.

	Intensity A.		Gamma correction		Rotate 20°	
	HL	HH	LL	LH	LL	LH
No Blocks	0.7549	0.8391	0.1991	0.8269	0.7660	0.4576
64×64 Blocks	0.7549	0.8391	0.1991	0.8269	0.7660	0.4576
32×32 Blocks	0.7736	0.8522	0.1991	0.8363	0.7764	0.4823
16×16 Blocks	0.7549	0.8391	0.1991	0.8477	0.7838	0.5060
8×8 Blocks	0.7980	0.8721	0.1991	0.8536	0.7893	0.5197

Table 3. Extracted watermarks for crop and re-watermark attacks.

	Crop 50.40%		Re-watermark		Re-watermark	
	LL	LH	LL	LH	HL	HH
No Blocks	0.5859	0.6996	0.8645	0.8269	0.8645	0.8645
64×64 Blocks	0.5860	0.6996	0.8645	0.8269	0.8645	0.8645
32×32 Blocks	0.5859	0.7167	0.9151	0.8363	0.9151	0.9151
16×16 Blocks	0.5859	0.7245	0.9516	0.8477	0.9516	0.9516
8×8 Blocks	0.5860	0.7309	0.9770	0.8536	0.9770	0.9770



a) Embed CPU time. b) Extract CPU time.

Figure 3. Block size versus CPU time graphs for a. embedding b.extraction phases.

Block-DWT watermarking does not negatively affect image fidelity because PSNR values are the same as the non-block counterparts.

The downside of the block-DWT algorithm is that it requires more CPU time for embedding and extraction phases than non-block-DWT watermarking. The CPU

time required for embedding and extraction for each block size is given in Figure 3. As block size decreased, CPU time increased. If extra CPU time is not a problem, block-DWT yields better results. It must be noted that only first-level DWT decomposition is possible for a block size of 8×8, which prohibits watermarking in further level DWT decompositions.

Table 4. SR values for extracted watermarks from LL band for each type of attack.

	LL						LH	HL	HH
	512×512	64×64	32×32	16×16	8×8	Tao&Esk	8×8	8×8	8×8
Filter	0.772	0.772	0.79	0.805	0.815	0.822	0.442	0.482	0.501
Gauss	0.685	0.688	0.709	0.727	0.737	0.717	0.62	0.622	0.625
Scale	0.75	0.75	0.773	0.795	0.807	0.7795	0.496	0.523	0.48
JPEG 75	0.898	0.898	0.905	0.912	0.916	0.92	0.52	0.551	0.481
Intens.Adj.	0.801	0.801	0.867	0.932	0.966	0.197	0.795	0.798	0.872
Hist.Eq.	0.627	0.627	0.664	0.7	0.718	0.421	0.706	0.708	0.754
Crop	0.586	0.586	0.586	0.586	0.586	0.996	0.731	0.761	0.734
Gamma	0.199	0.199	0.199	0.199	0.199	0.803	0.854	0.865	0.9
Rotate	0.766	0.766	0.776	0.784	0.789	0.91	0.52	0.558	0.484
Rewatermark	0.864	0.864	0.915	0.952	0.977	0.905	0.977	0.977	0.977
TOTAL	7.878	7.881	8.149	8.39	**8.526**	**8.0615**	**6,661**	**6.845**	**6.808**

3. Block DWT-Based Vector Image Watermarking

A color vector image was inserted as a watermark into a grayscale host image [20]. This DWT-based study was, robust and non-blind. To the best of our knowledge it is the first study to insert a vector image as a watermark. Adding a color vector image as a watermark facilitates proving ownership, without hesitation, in case the watermark is fully recovered. As vector images do not lose quality after resizing, the study is valuable for steganography.

Vector image formats were investigated to determine which format best suited the research. The SVG format was chosen because it was editable via a text editor and it stored objects in the XML format. The two vector SVG images used as watermarks and a section of one of the SVG images source code are shown in Figure 4.The same host image shown in Figure 1.a was used.

3.1. Watermarking Pre-Processing

The vector image source was preprocessed before embedding a watermark into the host image. The image format SVG has some backward compatibility parts that do not affect vector image appearance. Backward compatibility parts were eliminated beforehand. ID numbers given to objects were also disposed of, because when the vector image is extracted it can be rendered without the ID numbers. The numbers that exist in an SVG file were fetched and loaded in an array, programmatically. The part of an SVG file that didn't contain numeric tokens was saved as an SVGNS file. The array containing numeric values was saved as an SVGNN file. Numeric values were used for the watermark embedding phase. Another array was produced that contained the number types in the number array. This array was saved as an SVGNG file. For each value in the SVGNN array, one of the attribute values {"integer", "RGB intensity value", "real value between (-1.0 and 1.0)", "real value"} was assigned in SVGNG array. This SVGNG array was used for the watermark embedding and extraction phases and affected how many transform domain values would be consumed for each type of numeric value.

Numeric values were analyzed when parsing the SVG file during the preprocessing phase. The maximum of two values, namely the absolute value of the whole part of numbers and the RGB band values in the SVGNN array was found. Then how many bits this max value can be stored in a binary number system, which is assigned to IBC variable, was calculated. If, for example, the maximum value is 384, then 9 is assigned to the IBC. For real values' whole parts, another maximum of absolute values was found and the number of bits required to store this maximum value was calculated, and an RIBC value was assigned. The absolute maximum fractional part of real numbers between (-1, 1) (-1 and 1 not included) in an SVGNN array was determined. This value was multiplied by 100 and the first 3 digits were taken, which shows how many bits can store this value, and an RFBC variable was assigned. The three calculated values, IBC, RIBC, and RFBC were added to SVGNG array and stored in a file. These numbers in the SVGNG file were used during the watermark extraction phase. The preprocessing phase is shown in Figure 5.

```
<svg> <linearGradient Id = linearGradient5520>
<stop style = stop-color: #2b71d9;stop-opacity:
0.91891891; offset = 0 Id = stop5522 />
```

Figure 4. Two vector images and part of the source code of one of them.

3.2. Watermark Embedding Phase

After the original image was loaded from the file system, the SVGNN and SVGNG files that were produced during the preprocessing phase were loaded from files to arrays. The last three values of the SVGNG array were assigned to IBC, RIBC, and RFBC variables.

The original image was divided into 8×8 blocks and each block was transformed separately into DWT. Block size was 8×8, but the algorithm was also run according to block sizes of 16×16, 32×32, and 64×64. The purpose of dividing the image into blocks and taking each block's DWT was to increase the algorithm's resistance to attacks. After DWT, a DWT values pool was formed, as shown in Figure 6. If the original image is M×N, then there are (M/2) × (N/2) values in each of the LL, LH, HL, and HH bands. As the values in an SVGNN array would be watermarked separately into each of the LL, LH, HL, and HH bands, each numeric value would be watermarked into a KSPN number of transform values, as in Equation (8). A description of how each numeric value is watermarked with KSPN transform values is presented later.

$$KSPN = \lfloor (\lfloor M/2 \rfloor * \lfloor N/2 \rfloor) / SzWn \rfloor \quad (8)$$

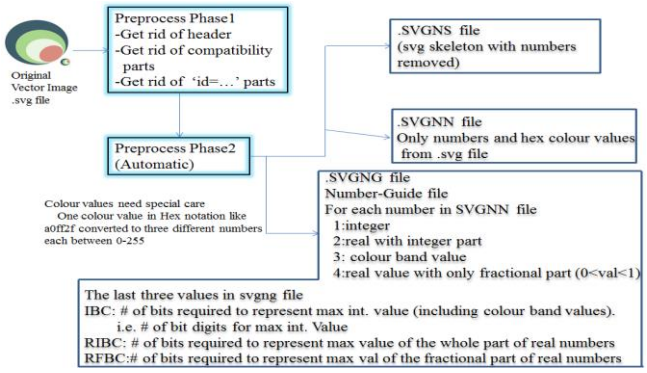


Figure 5. Vector image preprocessing.

Numeric values were categorized as integer value, RGB brightness value, real value with no integer part, real value with an integer value of 0, and those categories of each numeric values in the SVGNN array were loaded into the SVGNG array from the SVGNG file. Numeric values were considered bit sequences of 0 and 1. If the bit value of a numeric value was 1, a Pseudo Random Sequence (PRNS) was added to the corresponding DWT values, and if the bit value was 0, the same PRNS was subtracted from the corresponding DWT values.

Watermarking success was dependent on how close in proximity the numeric values extracted from the watermarked (and possibly attacked) image were to the original embedded numeric values; therefore, digit bit significance was taken into consideration. The number of values from the transformation value pool of the original host necessary to embed a 1 or 0 values for this digit was calculated. The calculated number is the PRNS length that can watermark a bit value of 1 or 0. The array for holding the value range count for the digits of integer values was PRNS_BC.

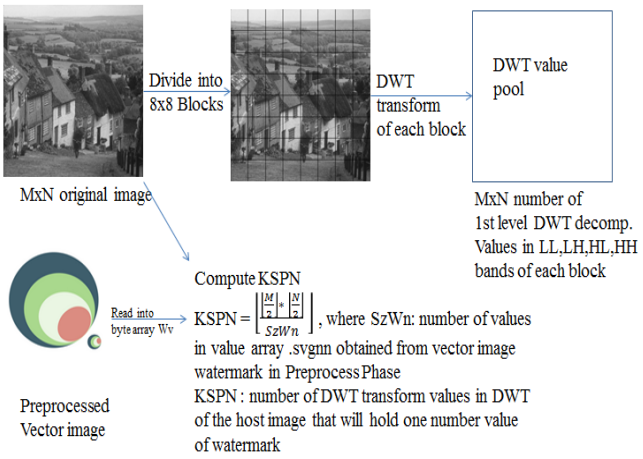


Figure 6. Formation of the DWT value pool.

The size of the PRNS_BC array was IBC. PRNS_BC (1) holds the number of transformation values (PRNS bit length) that hold the watermark bit value for the first, i.e., the LSB of the integer value. PRNS_BC (IBC) holds the PRNS length for the most significant bit of the integer value. As PRNS length increases resistance to attacks increases; as such, so we want the most significant bits to be more resistant to attacks than less significant bits. The contribution of digit bits to the overall numeric value increases in powers of 2, as the position of the digit increases in significance. Firstly, 20 values were given to each digit including the LSB. Next, significant bits were watermarked with longer PRNS as the length of PRNS was calculated using Equation (10) where *y* is the digit position to be embedded. The PRNS_R_BC array held the PRNS length of each bit for the whole number of real numeric values, which have a whole number greater than 0. PRNS_R_BC values were calculated according to Equation (13). PRNS length for the fractional part of real numbers between -1 and 1 were held in array PRNS_F_BC and calculated using Equation (15). If for example the KSPN value is 1969, and IBC is 8, then PRNS_BC array was calculated as in Equation (10) and given in Table 5. Whole calculated PRNS_R_BC and PRNS_F_BC are calculated as in Equation (13) and Equation (15).

$$kspn\_r = kspn - 20 \times ibc \quad (9)$$

$$prns\_bc(y) = 20 + floor(2^{(y-1)} / 2^{ibc}) \times kspn\_r \quad (10)$$

$$rbc = ribc + rfbc \quad (11)$$

$$kspn\_r = kspn - 20 \times rbc \quad (12)$$

$$prns\_r\_bc(y) = 20 + floor(2^{(y-1)} / 2^{rbc}) \times \quad (13)$$

$$kspn = kspn - 20 \times rfbc \quad (14)$$

$$prns\_f\_bc(y) = 20 + floor(2^{(y-1)} / 2^{rfbc}) \times \quad (15)$$

After *prns* length values were calculated and put in arrays, the actual pseudo random sequences of those lengths were produced during the image watermarking phase. All PRNS length values were put in the M_PRNS matrix and written to a file to be used during the watermark extraction phase. Figure 7 shows how each bit of the watermark was embedded.

Table 5. Examples of PRNS_BC values calculated for KSPN=1969 and IBC = 8.

Bit digit position	7	6	5	4	3	2	1	0
PRNS length	924	472	246	133	76	48	34	27

3.3. Watermark Extraction Phase

The original and watermarked (and possibly-attacked) image was transformed into the DWT domain, taking into consideration block size (64, 32, 16, and 8). The DWT values of any watermarked image were subtracted from the DWT values of the original image and difference of the DWT values pool was obtained. The SVGNG file that was formed and saved during the image-embedding phase was loaded from the file. The values PRNS_BC, PRNS_R_BC, and PRNS_F_BC were assigned values from the SVGNG array. For each of numeric value in an SVGNG array the type of numeric value determines the bit count of the value to be extracted. For each bit of the numeric value to be extracted the number of DWT values from the DWT difference value pool that would be used was calculated. Correlations between the PRNS values and the difference in DWT values were calculated. Correlations between the PRNS values and the negative difference in DWT values were also calculated. The extracted bit value was 1 or 0 according to comparison of the two computed correlation values. The extraction phase algorithm is shown in Figure 8. The vector watermarks extracted from watermarked and attacked images are shown in Table 9.

3.4. Evaluation of Extracted Vector Image Watermarks

The PSNR value between the watermarked and original image was 35.301, which is lower than previously reported by Huang and Fang [6], but they only used 80% JPEG quality compression and 3×3 median filter attacks. As such, the degree of robustness needed for their study was much lower than required for the present study. The watermarking method presented herein resisted JPEG compression that reduced image quality by 50% or 75%, 20 degree rotation, scaling and cropping which all distort an image to a greater extent than the two attacks applied by [6]. General comparison with [6] can be seen in Table 6.

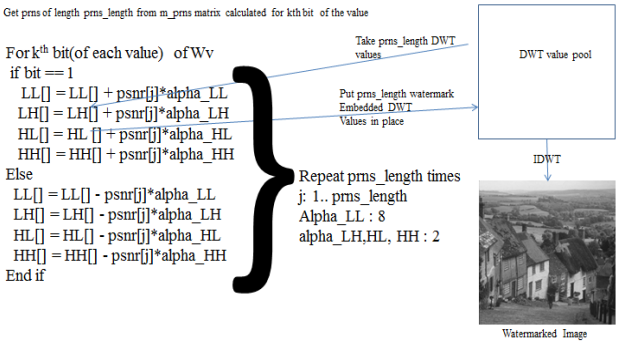


Figure 7. Watermarking one bit.

When vector image logo is used as a watermark, the success of the algorithm can be measured in two ways, as described in Table 7.

The SR and RMSE values for measuring the algorithm's robustness against various types of attacks can be seen in Table 8.

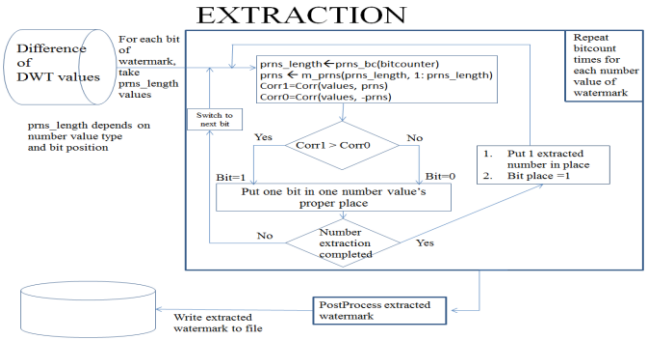


Figure 8. Watermark extraction.

Table 6. Results Given in Comparison to Huang and Fang [6].

	Transform Domain	**Watermark Type**	**Attacks Resisted**	**PSNR (AVG)**
Proposed Method	DWT	Vector Image Logo	JPEG %50 Q JPEG %25 Q Blurring Histogram Eq. Gaussian Noise Gamma Corre.	35.301
Huang & Fang [6]	DCT	EXIF MetaData	JPEG %80 Quality Filter 3x3	46.2

Table 7. Methods for measuring the success of the vector image watermark algorithm.

Method 1	1.1	Convert the original vector image and extracted vector image into pixel based images
Method 1	1.2	Convert pixel-based images to YUV format and compute SR values between two using Eq. (7). Usually the SR value is computed between two binary images but here we have two grayscale images. If the brightness value (0-255) difference between the extracted watermark pixel and original watermark pixel > 10, they are considered equal while calculating the SR value. As SR value increases, the algorithm is more effective.
Method 2	2.1	Calculate RMSE (2) between extracted numeric values and embedded original numeric values.
	2.2	As the RMSE value decreases, the algorithm is more effective

Table 8. SR and RMSE values between original and extracted values.

	No attack	Jpeg %50	Jpeg %25	Filter	Gaussian N.	Scale	Histogram Eq.	Brightness Adjust	Gamma C.	Rotate	Crop
SR	0.982	0.982	0.794	0.95	0.982	0.546	0.982	0.471	0.982	0.725	0.598
RMSE	0.000	0.000	0.001	0.017	0.000	0.044	0.000	0.148	0.000	0.140	0.069

Table 9. Vector image watermarks extracted from images attacked using various attacks for 2 different embedded vector watermarks.

		
JPEG %50	JPEG %25	Blurring
		
Scale-rescale	Histogram E	Adjust Brightn
		
Crop	Gaussian N	Gamma C
		
Rotation(2,1st Vector)		Rotation(2nd Vector)
		
JPEG %50	JPEG %25	Blurring
		
Scale-rescale	Histogram E	Adjust Brightn
		
Crop	Gaussian N	Gamma C

4. Conclusions

The present findings clearly show that dividing an image into blocks, taking the DWT of each block separately and then watermarking each block with a corresponding watermark block increased the resistance of watermarking against many types of the attacks. As block size decreased, the robustness of the algorithm increased. The only drawback of block-based watermarking is that the CPU time required increases as block size decreases. If there are no time constraints for the watermarking and extraction phases, then block-based watermarking increases robustness considerably.

The present study is the first to use a vector image as a watermark. Block-based DWT watermarking was used as a general approach. The SVG vector image format was used, and was preprocessed before watermarking. The numbers in the SVG file were categorized and watermarked on a bit-by-bit basis, taking into consideration the significance of each bit in numeric form. Various attacks were applied to the watermarked image showing that the algorithm was robust against many types of attacks. Nonetheless, the algorithm described herein was relatively weak against scale, rotation, and crop attacks.

As of future work, embedding vector image as watermark may be investigated in different transform domains and different methods can be developed to overcome the weakness of proposed algorithm to scale, crop, rotation attacks. Vector image can be hidden in a host image or file using a similar method as a steganography study.

References

[1] Amira-biad S., Bouden T., Nibouche M., and Elbaşı E., "A Bi-Dimensional Empirical Mode Decomposition Based Watermarking Scheme," *The International Arab Journal of Information Technology*, vol. 12, no. 1, pp. 24-31, 2015.

[2] Chamlawi R., Khan A., Idris A., and Munir Z., "A Secure Semi-Fragile Watermarking Scheme for Authentication and Recovery of Images based on Wavelet Transform," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 2, no. 11, pp. 727-7319, 2008.

[3] Chen B., Coatrieux G., Chen G., Sun X., Louis J., and Shu H., "Full 4-D Quaternion Discrete Fourier Transform Based Watermarking for Color Images," *Digital Signal Processing*, vol. 28, pp. 106-119, 2014.

[4] Cox I., Kilian J., Leighton F., and Shamoon T., "Secure Spread Spectrum Watermarking for Multimedia, in Image Processing," *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1673-1687, 1997.

[5] Dharwadkar N. and Amberker B., "An Efficient Non-blind Watermarking Scheme for Color Images using Discrete Wavelet Transformation," *International Journal of Computer Applications*, vol. 2, no. 3, pp. 60-66, 2010.

[6] Huang H. and Fang W., "Metadata-Based Image Watermarking for Copyright Protection," *Simulation Modelling Practice and Theory*, vol. 18, no. 4, pp. 436-445, 2010.

[7] Inamdar V. and Rege P., "Dual Watermarking Technique with Multiple Biometric Watermarks," *Sadhana*, vol. 39, no. 1, pp. 3-26, 2014.

[8] Jane O., İlk H., and Elbaşı E., "A Robust Transform Domain Watermarking Technique by Triangular and Diagonal Factorization," *in Proceedings of 36th International Conference on Telecommunications and Signal Processing*, Roma, pp. 867-871, 2013.

[9] Jane O. and Elbaşı E., "A New Approach in Non-Blind Watermarking Method Based on DWT and SVD Via LU Decomposition," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 22, no. 5, pp. 1354-1366, 2014.

[10] Jin R. and Kim J., "A Robust Watermarking Scheme for City Image," *International Journal of Security and Its Applications*, vol. 10, no. 1, pp. 303-314, 2016.

[11] Kusyk J. and Eskicioglu A., "A Semi-Blind Logo Watermarking Scheme for Color Images by Comparison and Modification of DFT Coefficients," *in Proceedings of SPIE-The International Society for Optical Engineering*, pp. 107-121, 2005.

[12] Lai C. and Tsai C., "Digital Image Watermarking Using Discrete Wavelet Transform and Singular Value Decomposition," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 11, pp. 3060-3063, 2010.

[13] Lang J. and Zhang Z., "Blind Digital Watermarking Method in the Fractional Fourier Transform Domain," *Optics and Lasers in Engineering*, vol. 53, pp. 112-121, 2014.

[14] Minamoto T. and Ohura R., "A Blind Digital Image Watermarking Method Based on the Dyadic Wavelet Transform and Interval Arithmetic," *Applied Mathematics and Computation*, vol. 226, pp. 306-319, 2014.

[15] Piva A., Barni M., Bartolini F., and Cappellini V., "DCT-Based Watermark Recovering without Resorting to the Uncorrupted Original Image," *in Proceedings of International Conference on Image Processing*, Santa Barbara, pp. 520-523, 1997.

[16] Ratakonda K., Dugad R., and Ahuja N., "Digital Image Watermarking: Issues in Resolving Rightful Ownership," *in Proceedings International Conference on Image Processing*, Chicago, pp. 414-418, 1998.

[17] Saryazdi S. and Nezamabadi-pour H., "A Blind Digital Watermark in Hadamard Domain," *World Academy of Science, Engineering and Technology*, pp. 126-129, 2005.

[18] Swanson M., Zhu B., and Tewfik A., "Transparent Robust Image Watermarking," *in Proceedings of 3rd IEEE International Conference on Image Processing*, Switzerland, pp. 211-214, 1996.

[19] Şenol A., Elbaşı E., Dinçer K., and Sever H., "A Block Size Analysis for Blocked Discrete Wavelet Watermarking," *in Proceedings of 7th International Conference on Information Security and Cryptology*, Istanbul, pp. 77-81, 2014.

[20] Şenol A., Elbaşı E., Dinçer K., and Sever H., "Bloklu Ayrık Dalgacık Dönüşümü ile Vektör Resim Damgalama," *in Proceedings of 23nd Signal Processing and Communications Applications Conference*, Malatya, 2015.

[21] Tao P. and Eskicioglu A., "A Robust Multiple Watermarking Scheme in the DWT Domain," *in Proceedings of Internet Multimedia Management Systems*, Philadelphia, pp. 133-144, 2004.

[22] Van Schyndel R., Tirkel A., and Osborne C., "A Digital Watermark," *IEEE International Conference Image Processing*, vol. 2, pp. 86-90, 1994.

[23] Wang X., Yang H., and Cui C., "An SVM-Based Robust Digital Image Watermarking Against Desynchronization Attacks," *Signal Processing*, vol. 88, no. 9, pp. 2193-2205, 2008.

Hayri Sever got his Bsc degree from Hacettepe University Computer Eng. Faculty, Ankara in 1986, MSc degree from Maine University, USA in 1991, Phd degree Louisiana University, Center of Advanced Computer Studies, in 1995. His research areas are Knowledge Discovery in Databases, Multimedia Retrieval Models and Systems, Multimedia Systems, Uncertainty Reasoning, Business Process Management, Machine Learning, and Speech Analysis.

Ahmet ŞENOL got his B.Sc. degree from Middle East Technical University(METU), Computer Engineering(CENG) Department in 1993, M.Sc. degree in 1993, METU CENG Department, Phd from Hacettepe University Computer Engineering Department. His current interest is on "Image Processing", "Image Watermarking", "Image Authentication", "Computer Forensics".

Ersin Elbasi is currently working for American University of the Middle East. He received MSc degree in computer science at Syracuse University; MPhil and PhD degrees in computer science at Graduate Center, The City University of New York. His research interests include multimedia security, event mining in video sequences and medical image processing