A Novel Reversible Watermarking Scheme For Providing Five Layer Security

V.Belmer Gladson,
Research Scholar,
Dept of Computer Science and Engg
M.S University,
Thirunelveli, Tamil Nadu, India

Dr. R.Balasubramanian.
Professor,
Dept of Computer Science and Engg
M.S University,
Thirunelveli, Tamil Nadu, India

Abstract— **The main objective of this paper is to develop a novel approach for reversible watermarking process. Reversible image watermarking can restore the original image without any distortion after the hidden data is extracted. Reversible watermarking has gained importance due to increased involvement of digital media. So this paper developed new scheme which utilizes Discrete Wavelet Transform (DWT), Arnold Transform (AT), Chaotic Based Encryption (CBE), Rationale of Prediction Using a Rhombus Pattern and Sorting the Prediction Error algorithms. So this novel scheme provide Five Layer Security for reversible watermarking process. First the input image is pre-processed by using Arnold transform and then DWT is applied onto the pre-processed image. And then the DWT transformed image is divided into 4 sub blocks. After dividing the sub blocks, the next step is to shuffle the sub blocks to differentiate between the homogenous and non homogenous blocks. After shuffling the next step is to apply the prediction and find the prediction error. Then the prediction error value is sorted to generate the places for embedding the watermark bit. At this stage, the watermark image is get from the user and these images are also pre-processed and encrypted. These encrypted watermark image pixels are embedded with the sorted prediction error values. Finally, the homogenous and non homogenous blocks are combined and the four sub blocks are also combined to generate the single image. After that inverse DWT and Arnold transform is applied to produce the watermarked image. To analyse the performance of novel method, several performance metrics are used. This paper utilizes Peak Signal-to-Noise Ratio (PSNR), Recall Rate, Normalized Cross-Correlation (NC) and Bit Error Ratio (BER) to analyses the performance. From the experimental results it is shown that the Proposed Approach performs better than the other method.**

Keywords— ***Reversible Watermarking, Discrete Wavelet Transform (DWT), Arnold Transform (AT), Chaotic Based Encryption (CBE), Rationale of Prediction Using a Rhombus Pattern and Sorting the Prediction Error.***

I. INTRODUCTION

In recent years, the growth of the Internet and media has given rise to the need to protect copyrights for various digital media (i.e. images, audio, video, etc.). Digital water marking is used to prevent digital media (images) from unauthorized access and unauthorized modifications. This is part of the cover-up of information that hides information about the property inside, the cover image in the form of corporate logos, property descriptions, and more. Water markers can be broadly classified into visible or invisible water markers [1]. In general, invisible water markers are used in digital multimedia communications systems. In water markers, markers are usually clearly visible after applying a common image. Visible watermarks transmit information about ownership directly to the media and can identify any copyright infringement attempts. Invisible Marks aims to embed invisible copyright information in organized media so that in the event of a copyright infringement, hidden information may be retrieved to identify the copyright owner. It is important that image IDs are resistant to normal image operations to ensure that hidden information is still retrieved after such changes. Embedding visible or invisible water labels damages the quality of the entire environment. A technique called reverse water signage allows legitimate users to remove embedded watermarks and restore original content if needed. A reversible water marking technique (RRW) is used to embed and retrieve water marks without damaging the channel, but also against unintentional attacks and extract as many water marks as possible. Done for the noise channel.

Shuang Yi, Yicong Zhou, and Zhongyun Hua [2] proposed an inverse method for hiding natural image data using block level prediction and error propagation. This method can embed confidential data into 2 × 2 blocks of images with extra pixels in each block. Extending this concept to encrypted domains, the authors propose an inverse method for hiding data in encrypted images using the Adaptive Level Prediction and Error Extraction (ABPEERDHEI). ABPEE-RDHEI encrypts the original image by rearranging the block to prevent data embedded asymmetry and applying a stream stream to the block that allowed the image to further enhance the security level. Thanks to the pixel-adapter and embedded selection, ABPEE-RDHEI can achieve high levels of

embeddedness and quality of encrypted images. The results and the original analysis show that ABPEE-RDHEI performed better than the current method. Yanchi and Lingling Liu [3] proposed a reversible water recognition algorithm with low color illusion. By analyzing the correlation of different color components in the color image and amplifying the pixels predicted by the adapter forecasting operator, data, water marking and retrieval of the original image. Experiments have shown that a given algorithm performs better than other classical color image algorithms and can retrieve the original image without loss. Digital water marking can be carried out in the domain or frequency range. The amplitude regression technique changes the pixel values in the image overlap with different algorithms. In the Territory is Chan Sykel. L.M. [4] Alternatives of the proposed LBS can be used to integrate the partitioned information into the cover. With the LB1 technique, a bit of text is replaced by unremarkable pixel images. The LB technique is simple and has low computational complexity. A Hello-based LB method for hiding messages in images was proposed by Mark Hassan et al. In [5]. They used the LDS replacement technique to embed water markers and water markers based on the spiral replacement algorithm. J. Feng, I. Lin, C. Tsai, Y. Chu [6] Over the years, there has been some research work on reverse water markers. Reverse water marking is an early type of water marking scheme. Not only can it increase the ownership of the original media, but it can also restore the original media entirely from the printer. This feature is suitable for a number of major media outlets such as medical and military imaging, as these types of media do not allow loss. The purpose of this document is to identify the purpose of reverse water markers that reflect current progress and to provide some scientific issues for the future. Liu and Jay Ying [7] proposed a color-blind, color-blind adaptive image resolution algorithm. The proposed algorithm increases the ability to attack the hidden features of the image, improves the detection security, and has greater resistance to noise, cut, and attack of JPEG compression. Khumb Biphavar, Bavana Palai and Drs. Sardana K. Mishra [8] proposed a method of water marking. This method uses conditioned local forecasts and produces better results. The general idea of digital water marking is to incorporate data into the media to ensure data security. The water filtration technique that meets these requirements is known as reverse water filtration.

In technique, the overlapping image frequency domain is transformed into the frequency domain by changes such as discrete cosine transform (DCT) and integral wavelength (IWT). Subsequently, the conversion coefficients of the lower-limb of the human visual system (HVS) change to include the secret message. D. Hung [9] proposed a water-marking algorithm based on the TCS 4c 4A technique. The localization of the tamper is done by means of variable predictions. Changes in AC ratio (1, 1) at all 4x4 pixel blocks of 3x3 quarters. The formula for quantifying AC (1, 1) has been modified. 4x4 dc integrator reduces artifact blocking caused by 8x8 DCT. The PSNR for the Lina images was found to be db42.98. The AC prediction value is used as the error control code. Jinal H. Mehta and Vishakha Kelkar [10] present a comparison between classical forecast error extension based on inverse water injection and a forecast error extension scheme considering areas of interest for gray scale medical imaging. . In the classical forecast error extension, the predicted error value addition is used for data embedding. The proposed scheme uses the extension of the forecast error by protecting the area of interest. Both schemes focus on concealing interchangeable data, where the original image can be easily improved after the load is extracted.

This article is to compare the effectiveness of the three methods used to image brain tumors. Brain tumor classification is an important procedure for early tumor diagnosis and radiotherapy planning. The distribution of brain tumors in brain tumor imaging is mainly performed manually in clinical practice. In addition to being time consuming, manually identifying brain tumors is difficult and depends on each operator. This article analyzes three approaches and compares them to find the best method for high-grade brain tumor distribution. Initial methods segmented images of brain tumors using LIPC-based classification. The second method uses waveforms and self-organizing maps (SOM). Various performance indicators are used to analyze the effectiveness of this method. This document uses the precision, speed of call, measurement of F, sensitivity and precision for the analysis of results. Experimental results show that the WaveWave-based method works better than other methods.

According to the study, concealed methods of known loss of data can be divided into two types: the Histogram (HR) circuit and the Distributed Histogram (HDC) scheme. The HR-based scheme was proposed by De Vleeschouwer et al. In [11], which is considered to be the start of unreliable data encryption. Using a circular interpretation of the digital variation, their circuits can achieve high convergence and strength against high-quality JPEG compression. At [11] the image of the machine was split into non-overlapping blocks. Then, two regions designated A and B are randomly selected from each block and their histograms are plotted. The vector directed from the center of the circle to the center of mass of regions A and B rotates, respectively, clockwise and counterclockwise. By controlling the direction of rotation, the "0" or "1" watermark is embedded in the machine image. In the process of embedding data, one may experience overload and overload. To prevent pixelation and overflow, modular operation -266 was used, resulting in a “salt and pepper” sound in the water image. To solve this problem, Zo [12], Ni [13], 14 have developed a mathematical model for embedded HDC in the field of amplitude and wavelength change, respectively. The HDC scheme embeds a watermark by changing the statistical properties of the cover image according to the histogram distribution of the image block. Ni et al. Calculates the arithmetic mean of each block. Bit 1 is incorporated by changing the arithmetic mean, multiplied by 0. If only a small number were fitted, the block remained unchanged. Zou et al. Calculating the Integer Wave Changes of Cover Images. After calculating the average of block coefficients HL1 or LH1, bit 1 is fitted with an average change

of 0 by the amount of displacement. If only a small number were fitted, the block remained unchanged. To control runoff and runoff, some labels have been changed from 1 to 0. These intentional errors were corrected by the correction code (ECC) for water retrieval. Therefore, the capability of the HDC scheme is greatly reduced. Hui-Yu Huang [15] presented a data loss method based on differential digital waveforms (SDs) in the frequency domain for embedding confidential messages. They use the quantified coefficients for the 9/7 wavelength filter in the DT and combine the secret data into the continuous zero coefficients of the high-frequency components. In, [16], Zhao et al. In 2011, he proposed a method that uses a multimedia histogram modification technique to create an integer pointer called "embedding level" to indicate the bin. First, the right-hand pointer has moved the EL + 1 threshold to the right, and the left-hand pointer has been moved to the left EL to the left to create an existing space that uses hierarchical concepts to embed level-level confidential data. Thus, the larger ELL suggests that more sensitive data may be embedded. In the above histogram displacement method, despite the high partial capacity, this algorithm results in a significant reduction in image quality. Reversal, strength and invisibility are required to provide the RRW framework.

The rest of the document is organized as follows: Section 2 presents an overview of the first method. Section 3 specifically describes the second method, including design concepts and practical approaches. Section IV provides an overview of the third method. Section V compares the results of the three methods. Finally, the conclusions are drawn in Section VI.

II. BASIC CONCEPTS

This work uses the wavelet transform (DWT), the Arnold transform (ET), the chaos-based encoding (CIO), the reasoning prediction using the robotic model, and the sorting of prediction error algorithms. The following subsections give a brief explanation of each algorithm. This section describes the general base concepts used in the proposed water marking scheme.

- DWT transform is used to insert the watermark in imperceptible manner. The watermark bits are inserted in the significant coefficients sub-bands by considering the human visual system (HVS) characteristics.
- CBE technique is used to encode the character text before embedding it in the image.
- Arnold Transform is used to make the watermark more secure and protect the embedded data.
- Prediction and prediction error sorting algorithm are used to find the best position to embed the watermark data increase the quality of watermarking

A. Discrete wavelet transform

Digital wavelet transform (SD) is a high-resolution mathematical tool that destroys hierarchical images and can be implemented effectively with the help of various digital filters. An image can be transmitted through high and low pass filters that are separated into several sub-lines with different resolutions. The implementation of SD images is decomposed into four components, namely, LMSA, HMA, which corresponds to the vertical, horizontal, and diagonal features, as shown in Ft. 1. The subframe L is about half of the original image. Whereas the subdivisions of L and H have horizontal or vertical edges or images. Hm. 2 provides an example of a 1-D SD image breaker displaying the L, L, MA, and HMA bundles.

LL1	LH1
HL1	HH1

Fig. 1. The principle of 1-level DWT.

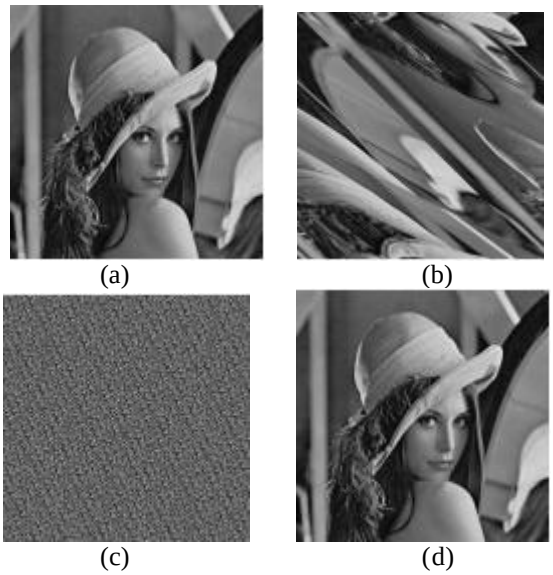
Fig. 2. 1-level DWT of Lena.

B. Arnold Transform

Arnold's transformation is a 2D chaotic map that is used to randomize a water symbol before embedding it in a cover image. Although there are several ways to mix them, in this document we will only discuss the conversion of Aran [17] to enhance the robustness and improve the safety of the water marking scheme. Arnold's transformation is the process of relocating pixels. Suppose that the original image is a N×N array and the coordinate of the pixel is x, y ∈ {0, 1, …, N − 1}. The generalized two dimension (2D) Arnold transform is defined as:

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & k \\ l & kl+1 \end{bmatrix} \begin{bmatrix} x_{n-1} \\ y_{n-1} \end{bmatrix} \bmod N \quad (1)$$

where xn and yn are the transformed coordinates corresponding to $xn - 1$ and $yn - 1$ after n iterations respectively, k and l are positive integers, and N represents the width or height of the square image processed. Arnold transform is a periodic process, so the original position of (x; y) coordinates gets back after T iterations. The factor T is called the transform period and depends on parameters k; l and n. These parameters will be used as secret keys in this paper. To recover the original image, periodicity is required. If the scrambling has performed n iterations; then the original image can be obtained by performing $T - n$ iterations. Fig. 3 illustrates an example of Arnold Transform into an image with different iterations.



(a) (b) (c) (d)

Fig. 3. Arnold transform with k = l = 1 (a) Lena 512 x 512. (b) AT with one iteration (c) AT with 12 iterations (d) AT with 190 iterations.

C. Chaotic Based Encryption

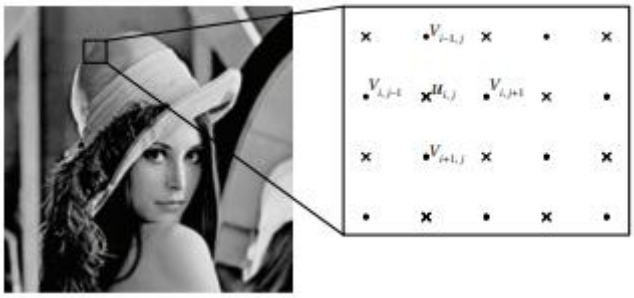
Chaotic encryption algorithms are an effective method of data encryption. Chaos signal has undeniable qualities and dynamic behavior. The chaotic system has high sensitivity to the original parameters. The first chaotic sequence is similar to white noise with abusive behavior with better correlation and complexity, and is defined as reported in [18] and [19] and given by:

$$Cn+1 = \mu \times Cn \times (1 - Cn) \quad (2)$$

where $0< \mu < 4$ typically μ is set to value 3.9 in order to achieve highest randomness and $0 < Cn < 1$ is the nth value generated from Eqn. 2. Different values of Cn could be obtained by varying the value of n from 0 to L-1. Here, L is the maximum number of chaotic values. By setting the initial values of μ and C0, we can get the required chaotic signal

D. Rationale of Prediction Using a Rhombus Pattern

The JPEG-LS prediction scheme is not suitable for sorting of the prediction errors. Therefore, a new prediction scheme is necessary that permits sorting. For this reason we propose a rhombus pattern prediction scheme. In order to predict the pixel value of position ui,j in Fig. 1, four neighboring pixels (i.e., vi,j−1, vi+1,j , vi,j+1, and vi−1,j) are used. The five pixels including ui,j comprise a cell which is used to hide one bit of data. All pixels of the image are divided into two sets: the “Cross” set and “Dot” set (see Fig. 4).



.

Fig. 4. Prediction pattern. The pixel value u of the Cross set can be predicted by using the four neighboring pixel values of the Dot set and expanded to hide one bit of data

The Cross set is used for embedding data and Dot set for computing predictors. Henceforth, this scheme will be called the Cross embedding scheme. The encoder of the Cross embedding scheme for a single cell is as follows. Center pixel ui,j of the cell can be predicted from the four neighboring pixels vi,j−1, vi+1,j , vi,j+1, and vi−1,j . The predicted value u i,j is computed as follows:

$$u'_{i,j} = \left\lfloor \frac{v_{i,j-1} + v_{i+1,j} + v_{i,j+1} + v_{i-1,j}}{4} \right\rfloor. \quad (3)$$

Based on the predicted value u i,j and original value ui,j , the prediction error di,j is computed as

$$d_{i,j} = u_{i,j} - u'_{i,j}. \quad (4)$$

This prediction error can be expanded to hide information like the difference expansion algorithm proposed by Tian [20] as follows:

$$D_{i,j} = 2d_{i,j} + b \quad (5)$$

where Di,j is the prediction error after expansion called modified prediction error. The value b is one bit of the hidden message. After data hiding, the original pixel value ui,j is changed to Ui,j as

$$U_{i,j} = D_{i,j} + u'_{i,j}. \quad (6)$$

The decoding procedure for the Cross embedding scheme for a single cell is an inverse of the encoding scheme. During data hiding, pixels from the Dot set are not modified, so the predicted values u i,j are also not changed. Using the predicted

value u i,j and the modified pixel value Ui,j , the decoder can exactly recover the embedded bit and original pixel value. The modified prediction error is computed as

$$D_{i,j} = U_{i,j} - u'_{i,j}. \quad (7)$$

The embedded bit value is computed as

$$b = D_{i,j} \bmod 2. \quad (8)$$

The original prediction error is computed as

$$d_{i,j} = \left\lfloor \frac{D_{i,j}}{2} \right\rfloor \quad (9)$$

The original pixel's value is computed as

$$u_{i,j} = u'_{i,j} + d_{i,j} \quad (10)$$

E. Sorting the Prediction Error

Kamstra and Heijmans' [21] use of sorting introduced a significant performance advantage over previous methods. However, sorting is possible only when cells are independent. In other words, embedding data into one cell should not affect the other cells. However, Thodi and Rodriguez's method [22] produces dependent cells, where embedding data to one cell changes prediction errors of other cells. Note that the Dot and Cross sets of the rhombus scheme are independent each other. In order to hide more data with less visual degradation, the order to hide data into the cells needs to be changed. Thus, the cells can be rearranged by sorting according to the correlation of neighboring pixels. Local variance μi,j for each cell can be computed from the neighboring pixels vi,j−1, vi+1,j , vi,j+1, and vi−1,j (see Fig. 4) as follows:

$$\mu_{i,j} = \frac{1}{4}\sum_{k=1}^{4} (\Delta v_k - \Delta \bar{v}_k)^2 \quad (11)$$

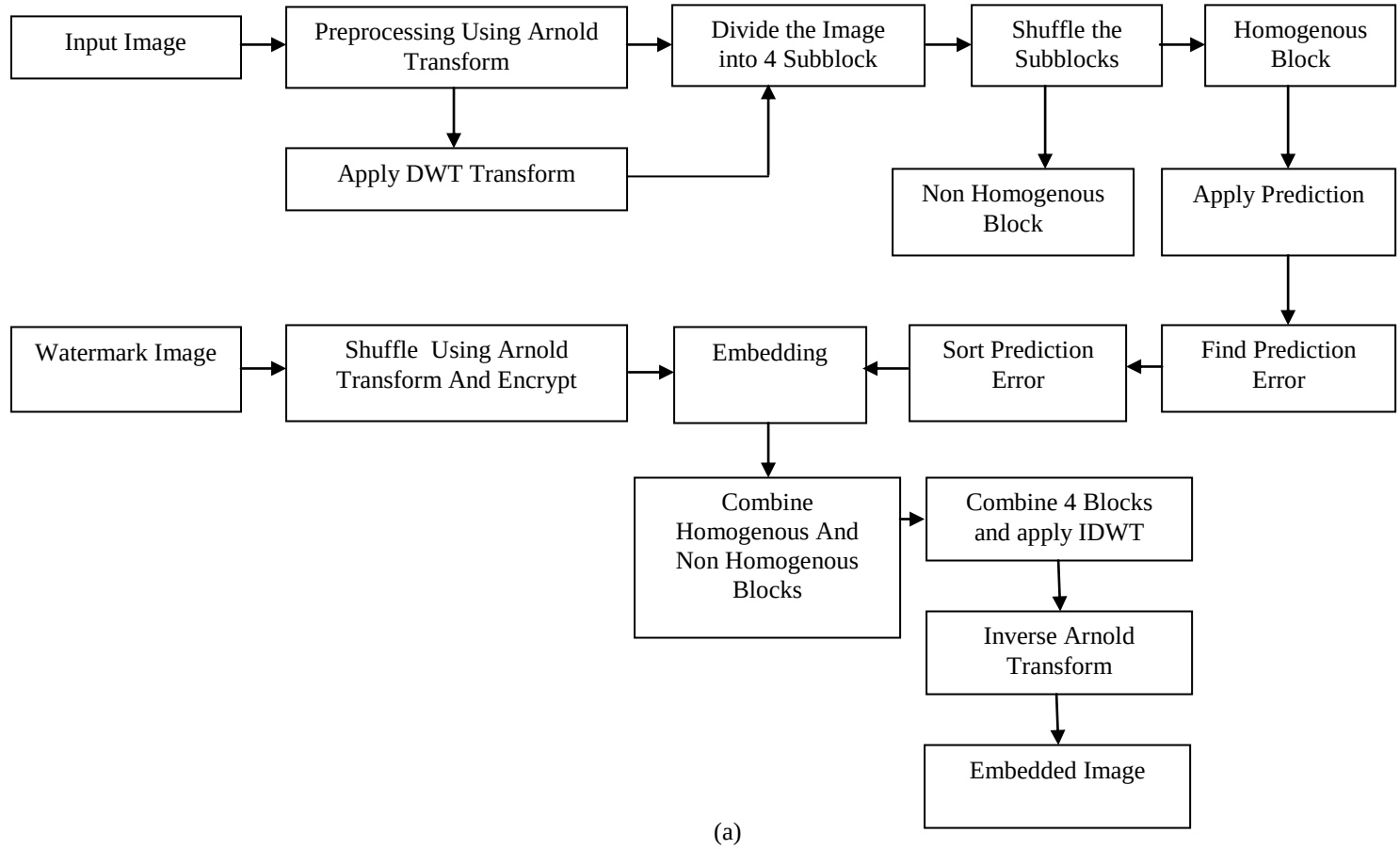
where v1 = |vi,j−1 − vi−1,j|, v2 = |vi−1,j − vi,j+1|, v3 = |vi,j+1 − vi+1,j|, v4 = |vi+1,j − vi,j−1|, v¯k = (v1 + v2 + v3 + v4)/4. Local variances μ calculated by using (11) achieve the most appropriate sorting for improving performance of data hiding. The local variance μi,j has several features. First, this value remains unchanged after data hiding. Second, this value is proportional to the magnitude of prediction error di,j of the cell. For example, a small variance indicates a small magnitude of prediction errors, and vice versa.

Assume that dsort is the sorted row of all di,j . The histogram shift method embeds data with the thresholds Tn and Tp. Cells are sorted in ascending order of the local variance values. Cells with smaller variance values are better for data hiding. Thus, the embedding process starts from the cell with the smallest variance value in the sorted row, and moves on to the next cells until the last bit of data is embedded. The histogram shift method is applied to the cells with small variance values which are comprised of sets E and S where P = |E|. Of course, the payload P also depends on threshold values Tn and Tp. Needless to say, for extracting the embedded data and recovering the original image, a few bits of information about the position of the last changed cell (or embedding capacity) and threshold values are necessary.

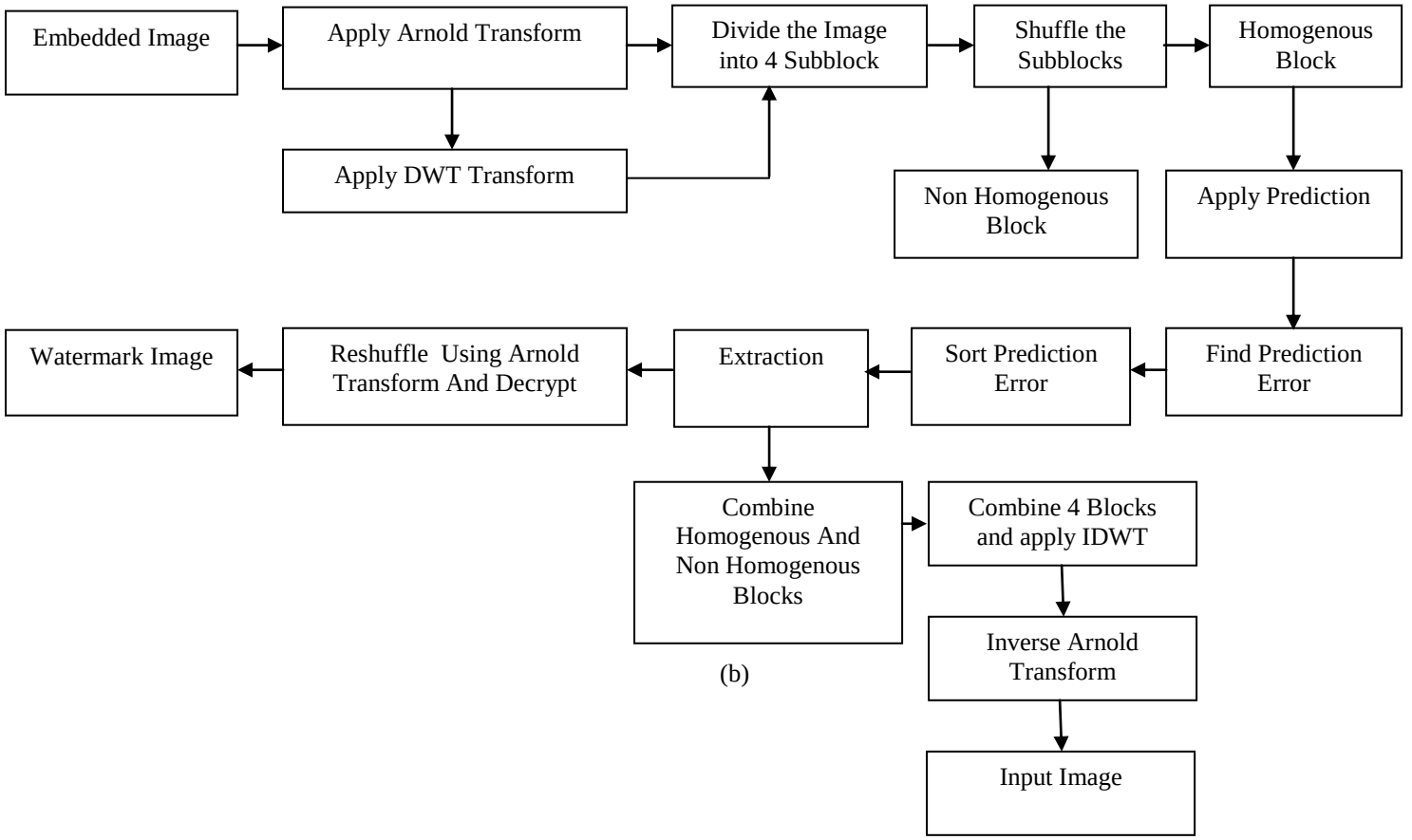
III. THE PROPOSED REVERSIBLE WATERMARKING APPROACH

The materials, brain MR image datasets, and methods used to The overall system of our proposed approach is illustrated in Fig. 5. Our proposed system consist of two phases such as watermark embedding phase and watermark extraction phase. In embedding phase, first the input image is pre-processed by using Arnold transform and then DWT is applied onto the pre-processed image. And then the DWT transformed image is divided into 4 sub blocks. After dividing the sub blocks, the next step is to shuffle the sub blocks to differentiate between the homogenous and non homogenous blocks. After shuffling the next step is to apply the prediction and find the prediction error. Then the prediction error value is sorted to generate the places for embedding the watermark bit. At this stage, the watermark image is get from the user and these images are also pre-processed and encrypted. These encrypted watermark image pixels are embedded with the sorted prediction error values. Finally, the homogenous and non homogenous blocks are combined and the four sub blocks are also combined to generate the single image. After that inverse DWT and Arnold transform is applied to produce the watermarked image.

. In extraction phase, first the watermarked image is pre-processed by using Arnold transform and then DWT is applied onto the pre-processed image. And then the DWT transformed image is divided into 4 sub blocks. After dividing the sub blocks, the next step is to shuffle the sub blocks to differentiate between the homogenous and non homogenous blocks. After shuffling the next step is to apply the prediction and find the prediction error. Then the prediction error value is sorted to get the places for extracting the watermark bit. At this stage, the watermark bit are extracted and then it is decrypted. These decrypted watermark image is considered as the output watermark image. Finally, the homogenous and non homogenous blocks are combined and the four sub blocks are also combined to generate the single image. After that inverse DWT and Arnold transform is applied to produce the original image.


Input Image
Preprocessing Using Arnold Transform
Divide the Image into 4 Subblock
Shuffle the Subblocks
Homogenous Block
Apply DWT Transform
Non Homogenous Block
Apply Prediction
Watermark Image
Shuffle Using Arnold Transform And Encrypt
Embedding
Sort Prediction Error
Find Prediction Error
Combine Homogenous And Non Homogenous Blocks
Combine 4 Blocks and apply IDWT
Inverse Arnold Transform
Embedded Image

(a)


Embedded Image
Apply Arnold Transform
Divide the Image into 4 Subblock
Shuffle the Subblocks
Homogenous Block
Apply DWT Transform
Non Homogenous Block
Apply Prediction
Watermark Image
Reshuffle Using Arnold Transform And Decrypt
Extraction
Sort Prediction Error
Find Prediction Error
Combine Homogenous And Non Homogenous Blocks
Combine 4 Blocks and apply IDWT
Inverse Arnold Transform
Input Image

(b)

A. Watermark Embedding Algorithm

1. The input image is pre-processed by using Arnold transform.
2. DWT is applied onto the pre-processed image.
3. DWT transformed image is divided into 4 sub blocks.
4. Shuffle the sub blocks to differentiate between the homogenous and non homogenous blocks.
5. Apply the prediction and find the prediction error on these homogenous and non homogenous blocks.

1. The prediction error value is sorted to generate the places for embedding the watermark bit.
2. The watermark image is get from the user and these images are also pre-processed and encrypted.
3. Encrypted watermark image pixels are embedded with the sorted prediction error values.
4. The homogenous and non homogenous blocks are combined and the four sub blocks are also combined to generate the single image.
5. inverse DWT and Arnold transform is applied to produce the embedded image

B. Watermark Extraction Algorithm

1. The watermarked image is pre-processed by using Arnold transform.
2. DWT is applied onto the pre-processed image.
3. DWT transformed image is divided into 4 sub blocks.
4. Shuffle the sub blocks to differentiate between the homogenous and non homogenous blocks.
5. Apply the prediction and find the prediction error on these homogenous and non homogenous blocks.
6. The prediction error value is sorted to generate the places for embedding the watermark bit.
7. Extract watermark image and then decrypted.
8. The homogenous and non homogenous blocks are combined and the four sub blocks are also combined to generate the single image.
9. inverse DWT and Arnold transform is applied to produce the original image

IV. PERFORMANCE ANALYSIS

A. Expiremental Images

Experiments were conducted on a group of standard ieee images to verify the effectiveness of the proposed scheme. All of these images are 512 × 512 images. These images are used as cover image. For watermarking purpose 256 × 256 size logo images are used. Some of the images are shown in Figure 7.

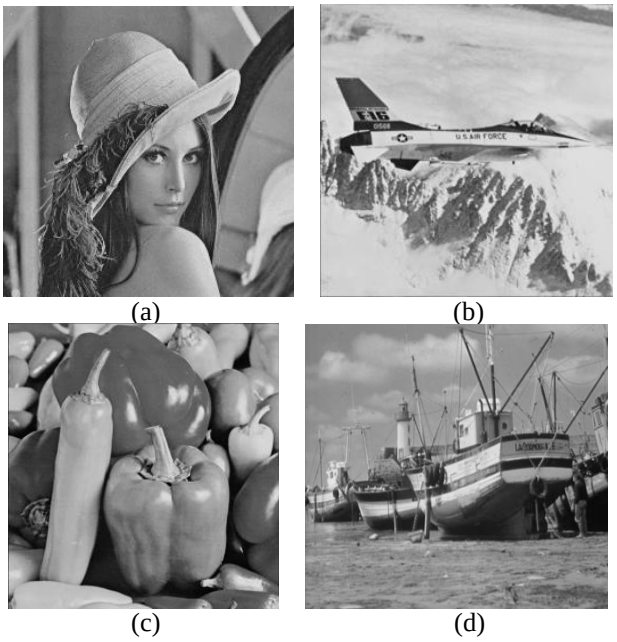
B. Performance Analysis

To evaluate the performance of the proposed method several performance metrics are available. This paper uses the Peak Signal-to-Noise Ratio (PSNR), Recall Rate, Normalized Cross-Correlation (NC) and Bit Error Ratio (BER) to analyses the performance.

1. Peak Signal-to-Noise-Ratio (PSNR):

The peak signal-to-noise ratio (PSNR) is used to evaluate the quality between the attacked image and the original image. The PSNR formula is defined as follows:

$$PSNR = 10 \times \log 10 \frac{255 \times 255}{\frac{1}{H \times W}\sum_{x=0}^{H-1}\sum_{y=0}^{W-1}[f(x,y) - g(x,y)]^2} dB$$



(a) (b) (c) (d)

Fig. 7. Expiremental Images

2. Structural Similarity Index

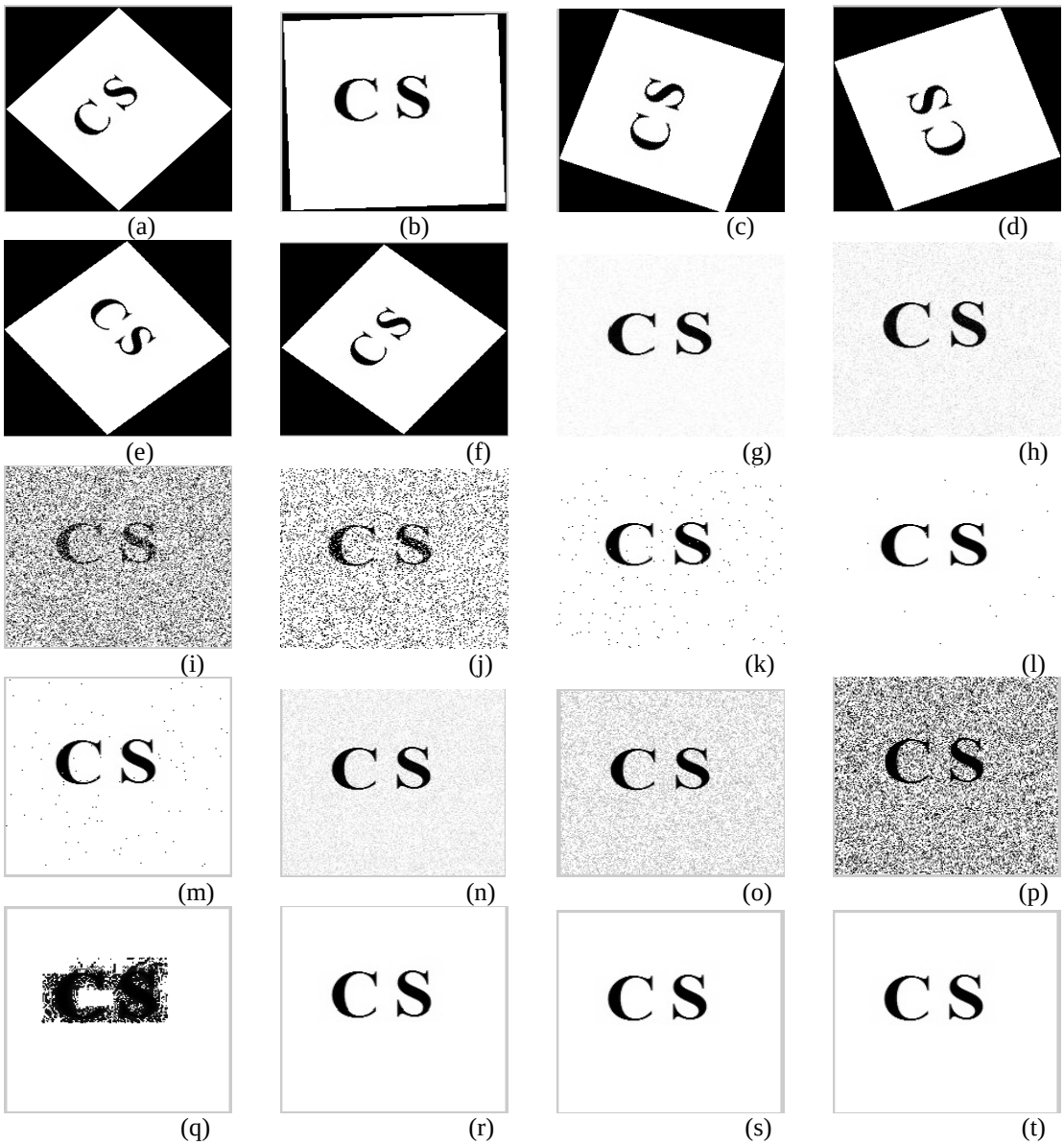
The structural similarity index is a method for measuring the similarity between the 3D image and the original 2D image.

$$SSIM\ (y, \hat{y}) = \frac{\left(2_{\mu_y\mu_{\hat{y}}} + c_1\right)(2\sigma_{y\hat{y}} + c_2)}{(\mu_y^2 + \mu_{\hat{y}}^2 + c_1)(\sigma_y^2 + \sigma_{\hat{y}}^2 + c_2)}$$

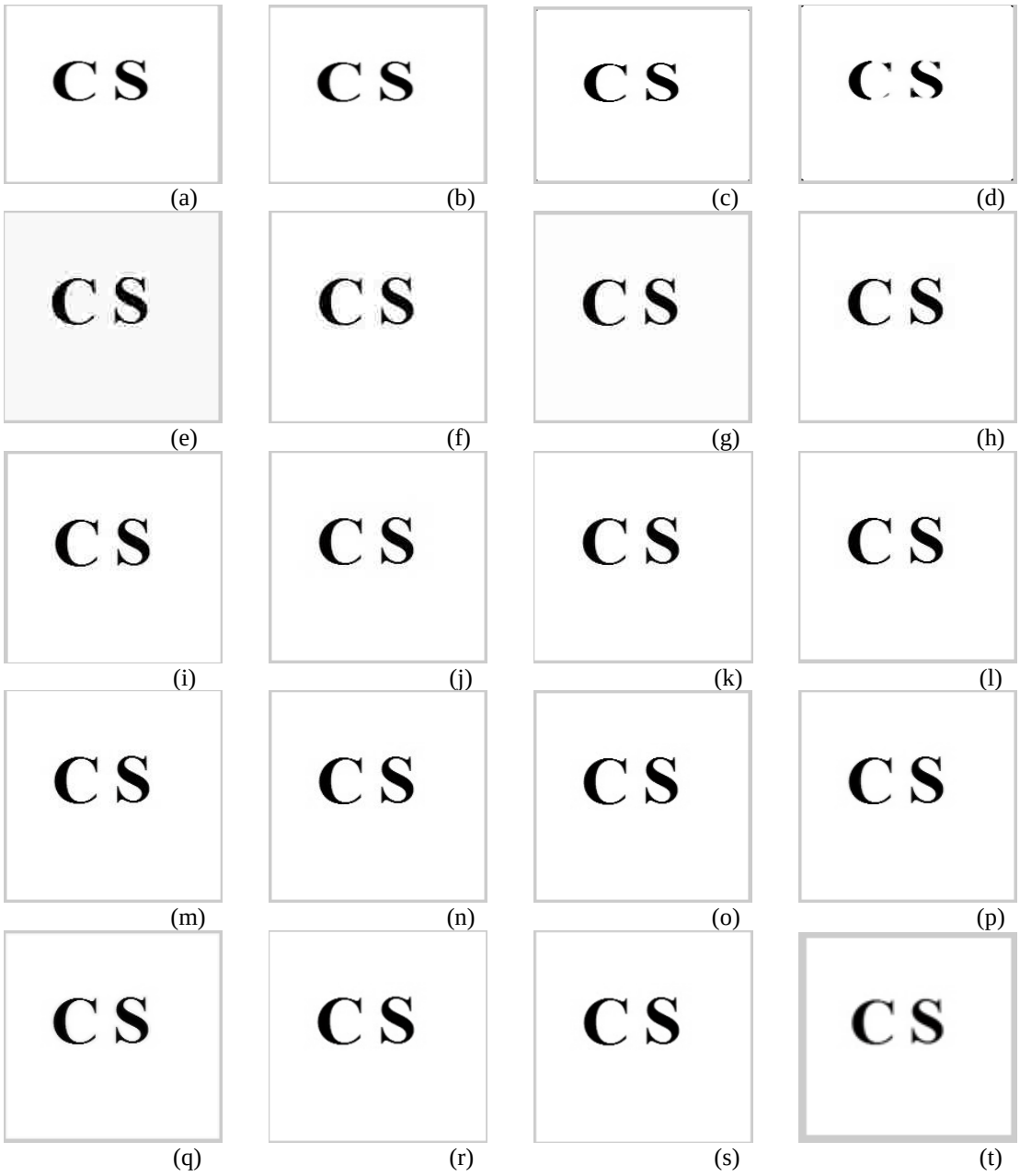
where, $\hat{Y}$ is the 3D image, the Y is the original 2D image, μ is the mean and the is the variance.

3.Normalised Cross-Correlation (NCC)

This is one of the important parameter for calculating the robustness. The robustness is evaluated by using Normalized Cross-Correlation (NC) is shown in below eq



(a) Extracted image under Rotation (45^o) (b) Extracted image under Rotation (2^o) (c) Extracted image under Rotation (70^o) (d) Extracted image under Rotation (110^o) (e) Extracted image under Rotation (-50^o) (f) Extracted image under Rotation (50^o) (g) Extracted image under Gaussian noise (0.001) (h) Extracted image under Gaussian noise (0.005) (i) Extracted image under Gaussian noise (0.05) (j) Extracted image under Salt & Pepper (0.3) (k) Extracted image under Salt & Pepper (0.01) (l) Extracted image under Salt & Pepper (0.001) (m) Extracted image under Salt & Pepper (0.005) (n) Extracted image under Speckle noise (0.01) (o) Extracted image under Speckle noise (0.04) (p) Extracted image under Speckle noise (0.5) (q) Extracted image under Histogram Equalization (r) Extracted image under Crop (10) (s) Extracted image under Crop (20) (t) Extracted image under Crop (30)



(a) Extracted image under Gaussian filter (3x3) (b) Extracted image under Gaussian filter (5x5) (c) Extracted image under Median filter (3x3) (d) Extracted image under Median filter (5x5) (e) Extracted image under JPEG compression (5) (f) Extracted image under JPEG compression (10) (g) Extracted image under JPEG compression (20) (h) Extracted image under JPEG compression (30) (i) Extracted image under JPEG compression (40) (j) Extracted image under JPEG compression (50) (k) Extracted image under JPEG compression (60) (l) Extracted image under JPEG compression (70) (m) Extracted image under JPEG compression (80) (n) Extracted image under JPEG compression (90) (o) Extracted image under JPEG compression (100) (p) Extracted image under Wiener filter (3x3) (q) Extracted image under Wiener filter (5x5) (r) Extracted image under Scale (2) (s) Extracted image under Scale (4) (t) Extracted image under Scale (0.5)

Table 1
The NC Of The Extracted Watermark Under Different Attacks

Attack	LWT		Proposed		DWT		SWT	
	DB1	DB3	DB1	DB3	DB1	DB3	DB1	DB3
None								
Pepper & salt noise (density 0.3)	0.8816	0.9435	0.8546	0.9568	0.8741	0.9389	0.7526	0.8564
Pepper & salt noise (density 0.01)	0.8856	0.9518	0.8947	0.9618	0.6856	0.9456	0.7212	0.6879
Pepper & salt noise (density 0.001)	0.8845	0.9518	0.8994	0.9618	0.6845	0.9475	0.7989	0.7548
Pepper & salt noise (density 0.005)	0.8856	0.9534	0.8974	0.9634	0.7856	0.9489	0.7848	0.7956
Speckle noise (var = 0.01)	0.8879	0.9377	0.8952	0.9477	0.8879	0.9124	0.6703	0.8457
Speckle noise (var = 0.04)	0.8712	0.9204	0.8873	0.9104	0.9712	0.9025	0.7588	0.8459
Speckle noise (var = 0.4)	0.8623	0.9808	0.8704	0.9908	0.6623	0.9546	0.7480	0.7451
Gaussian noise (M = 0, var = 0.001)	0.8856	0.9766	0.8979	0.9666	0.6856	0.9578	0.7253	0.8344
Gaussian noise (M = 0, var = 0.005)	0.8853	0.9517	0.8925	0.9617	0.6853	0.9345	0.7656	0.7259
Gaussian noise (M = 0, var = 0.5)	0.8521	0.9769	0.8636	0.9869	0.7521	0.9589	0.7435	0.8456
Rotation (angle 45)	0.8894	0.9623	0.8983	0.9823	0.7894	0.9478	0.7915	0.8589
Rotation (angle 2)	0.8845	0.9865	0.8981	0.9981	0.6845	0.9687	0.7427	0.7756
Rotation (angle 70)	0.8897	0.9598	0.8981	0.9668	0.7897	0.9356	0.7536	0.8653
Rotation (angle 110)	0.8812	0.9578	0.8981	0.9686	0.6812	0.9389	0.7602	0.7532
Rotation (angle −50)	0.8854	0.9648	0.8984	0.9751	0.7854	0.9245	0.7476	0.8485
Rotation (angle 50)	0.8843	0.9748	0.8985	0.9850	0.6843	0.9564	0.7224	0.7786
Histogram Equalization	0.8894	0.9715	0.8990	0.9865	0.7894	0.9521	0.7848	0.8875
JPEG compression Q = 5	0.8898	0.9584	0.8952	0.9666	0.6898	0.9358	0.7590	0.7425
JPEG compression Q = 10	0.8825	0.9426	0.8972	0.9557	0.7825	0.9278	0.7628	0.8954
JPEG compression Q = 20	0.8843	0.9056	0.8983	0.9120	0.6843	0.8459	0.7229	0.7678
JPEG compression Q = 30	0.8873	0.9289	0.8987	0.9327	0.6873	0.9097	0.8623	0.7456
JPEG compression Q = 40	0.8854	0.9348	0.8988	0.9411	0.8854	0.9105	0.7155	0.7389
JPEG compression Q = 50	0.8854	0.9378	0.8990	0.9487	0.6854	0.9156	0.7670	0.8824
JPEG compression Q = 60	0.8894	0.9456	0.8992	0.9564	0.6894	0.9254	0.7978	0.8486
JPEG compression Q = 70	0.8854	0.9378	0.8994	0.9644	0.6854	0.9142	0.7270	0.8245
JPEG compression Q = 80	0.8856	0.9689	0.8995	0.9746	0.8856	0.9436	0.7186	0.7489
JPEG compression Q = 90	0.8857	0.9748	0.8997	0.9887	0.8857	0.9548	0.7996	0.8389
JPEG compression Q = 100	0.8895	0.9859	0.8998	0.9922	0.87895	0.9589	0.7834	0.7856
Median filtering (3 ×3)	0.8835	0.9412	0.8982	0.9581	0.7835	0.9241	0.7563	0.8942
Median filtering (5 ×5)	0.8857	0.9056	0.8950	0.9421	0.7857	0.8456	0.7468	0.8735
Wiener filtering (3 ×3)	0.8865	0.9456	0.8984	0.9509	0.7865	0.9274	0.7609	0.8821
Wiener filtering (5 ×5)	0.8898	0.9312	0.8974	0.9404	0.7898	0.9138	0.7261	0.8942
Gaussian filtering (3 ×3)	0.8875	0.9548	0.8987	0.9652	0.7875	0.9256	0.7108	0.8546
Gaussian filtering (5 ×5)	0.8854	0.9546	0.8987	0.9649	0.8854	0.9245	0.7806	0.7975
Scaling (zoomout = 0.5, zoomin = 2)	0.8856	0.9045	0.8948	0.9356	0.8856	0.8012	0.7467	0.7835
Scaling (zoomout = 0.25, zoomin = 4)	0.8876	0.9423	0.8788	0.9588	0.7876	0.9249	0.7637	0.8935
Scaling (zoomout = 2, zoomin = 0.5)	0.8845	0.9689	0.8992	0.9772	0.7845	0.9458	0.7816	0.8724
Crop 10	0.8876	0.9745	0.8994	0.9833	0.7876	0.9549	0.7228	0.8976
Crop 20	0.8872	0.9712	0.8989	0.9820	0.7872	0.9513	0.7100	0.7245
Crop 30	0.8873	0.9623	0.8986	0.9780	0.7873	0.9458	0.7465	0.7831

Table 2
The PSNR of the embedded Watermark Under Different Attacks

Attack	LWT		Proposed		DWT		SWT	
	DB1	DB3	DB1	DB3	DB1	DB3	DB1	DB3
None								
Pepper & salt noise (density 0.3)	34.26	38.56	40.15	44.23	28.45	30.85	24.56	25.59
Pepper & salt noise (density 0.01)	35.12	39.45	41.46	44.56	28.36	31.26	22.96	26.45
Pepper & salt noise (density 0.001)	35.82	36.12	40.56	44.89	28.59	32.48	23.59	26.85
Pepper & salt noise (density 0.005)	35.62	37.45	40.84	45.23	29.14	32.95	24.78	25.48
Speckle noise (var = 0.01)	35.23	36.45	40.26	43.12	29.58	32.18	22.94	27.41
Speckle noise (var = 0.04)	35.89	37.59	40.16	44.89	28.67	31.28	23.82	27.92
Speckle noise (var = 0.4)	34.21	38.15	40.45	44.78	28.45	30.85	24.61	25.38
Gaussian noise (M = 0, var = 0.001)	34.59	39.57	40.21	44.72	29.38	30.57	24.93	25.10
Gaussian noise (M = 0, var = 0.005)	35.85	38.95	40.19	43.52	29.84	31.75	22.48	25.46
Gaussian noise (M = 0, var = 0.5)	34.20	39.56	40.24	43.89	28.39	31.27	23.94	26.48
Rotation (angle 45°)	35.46	39.25	40.16	44.28	29.86	30.86	24.86	25.43
Rotation (angle 2°)	34.42	38.16	40.12	44.21	28.54	30.68	24.71	26.86
Rotation (angle 70°)	35.85	37.16	41.67	43.89	29.43	31.26	23.31	25.49
Rotation (angle 110°)	34.95	36.48	41.21	44.27	28.16	30.29	23.51	26.38
Rotation (angle −50°)	33.57	36.16	40.56	43.56	29.46	32.64	22.16	26.92
Rotation (angle 50°)	33.41	36.45	40.58	43.68	29.87	32.54	24.59	27.81
Histogram Equalization	33.21	37.56	40.56	42.56	29.92	31.62	24.67	25.18
JPEG compression Q = 5	34.59	38.92	41.28	42.01	28.49	32.74	23.05	26.94
JPEG compression Q = 10	35.92	37.58	40.89	43.67	28.57	32.76	24.94	27.86
JPEG compression Q = 20	33.21	38.62	41.57	42.15	28.10	31.68	23.16	26.84
JPEG compression Q = 30	34.26	36.21	40.69	42.59	28.36	30.29	22.49	25.93
JPEG compression Q = 40	34.49	37.56	40.16	44.67	28.15	30.45	23.82	26.53
JPEG compression Q = 50	35.64	36.99	40.58	44.28	29.64	30.96	23.76	26.48
JPEG compression Q = 60	33.59	38.64	40.28	44.75	28.35	31.92	22.64	27.94
JPEG compression Q = 70	33.48	39.56	40.21	43.12	29.43	32.52	24.16	27.56
JPEG compression Q = 80	34.75	38.64	40.76	44.52	29.51	32.84	24.53	27.64
JPEG compression Q = 90	35.62	37.30	41.94	43.01	28.38	31.29	23.90	25.63
JPEG compression Q = 100	34.21	38.64	41.64	43.82	28.92	30.16	22.46	25.97
Median filtering (3 × 3)	35.28	37.41	40.57	44.28	28.16	31.28	22.72	25.84
Median filtering (5 × 5)	34.65	39.06	40.64	43.67	29.38	32.94	24.76	26.83
Wiener filtering (3 × 3)	35.20	38.46	40.27	44.27	29.84	31.28	22.67	26.81
Wiener filtering (5 × 5)	35.28	39.24	41.68	44.19	28.96	32.61	22.89	25.08
Gaussian filtering (3 × 3)	33.61	38.62	41.56	43.17	29.82	30.69	22.16	27.09
Gaussian filtering (5 × 5)	33.84	37.45	40.28	43.89	28.74	30.49	23.84	26.18
Scaling (zoomout = 0.5, zoomin = 2)	34.20	36.26	40.96	44.68	29.68	31.23	23.64	26.19
Scaling (zoomout = 0.25, zoomin = 4)	34.39	38.20	40.82	42.95	29.38	32.82	23.61	27.83
Scaling (zoomout = 2, zoomin = 0.5)	35.64	39.54	41.20	43.61	28.49	31.27	24.53	27.49
Crop 10	33.29	37.12	41.26	44.38	28.81	31.06	22.86	26.35
Crop 20	33.82	38.64	41.85	42.61	28.64	32.08	24.06	26.39
Crop 30	34.61	39.41	41.24	44.50	29.38	32.82	23.94	25.18

Table 3
The BER of the extracted watermarks under different attacks.

Attack	LWT		Proposed		DWT		SWT	
	DB1	DB3	DB1	DB3	DB1	DB3	DB1	DB3
None								
Pepper & salt noise (density 0.3)	10.65	9.18	8.56	7.24	12.46	11.49	14.26	13.62
Pepper & salt noise (density 0.01)	10.67	9.64	8.49	7.59	12.49	11.34	14.59	13.49
Pepper & salt noise (density 0.001)	9.67	8.13	7.48	6.49	11.67	10.49	13.49	12.68
Pepper & salt noise (density 0.005)	9.48	8.27	7.35	6.57	11.62	10.49	13.47	12.74
Speckle noise (var = 0.01)	10.37	9.37	8.15	7.29	12.76	11.57	14.46	13.26
Speckle noise (var = 0.04)	10.48	9.28	8.69	7.34	12.96	11.49	14.60	13.58
Speckle noise (var = 0.4)	9.64	8.64	7.20	6.89	11.06	10.59	13.76	12.49
Gaussian noise (M = 0, var = 0.001)	9.78	8.19	7.56	6.24	11.45	10.64	13.49	12.48
Gaussian noise (M = 0, var = 0.005)	10.38	9.37	8.24	7.49	12.49	11.57	14.76	13.59
Gaussian noise (M = 0, var = 0.5)	9.67	8.49	7.34	6.35	11.67	10.64	13.47	12.39
Rotation (angle 45˚)	10.28	9.67	8.94	7.54	12.57	11.34	14.59	13.28
Rotation (angle 2˚)	9.91	8.27	7.09	6.29	11.08	10.49	13.46	12.68
Rotation (angle 70˚)	10.37	9.86	8.24	7.24	12.95	11.27	14.59	13.69
Rotation (angle 110˚)	10.64	9.75	8.43	7.29	12.48	11.34	14.76	13.48
Rotation (angle −50˚)	10.27	9.05	8.24	7.34	12.67	11.29	14.59	13.67
Rotation (angle 50˚)	10.94	9.38	8.59	7.26	12.46	11.34	14.29	13.05
Histogram Equalization	9.18	8.37	7.24	6.19	11.24	10.49	13.92	12.67
JPEG compression Q = 5	9.67	8.37	7.24	6.34	11.60	10.57	13.86	12.49
JPEG compression Q = 10	9.17	8.37	7.29	6.27	11.49	10.67	13.76	12.93
JPEG compression Q = 20	9.46	8.43	7.25	6.29	11.37	10.34	13.09	12.67
JPEG compression Q = 30	9.73	8.09	7.34	6.37	11.06	10.49	13.43	12.46
JPEG compression Q = 40	10.84	9.37	8.64	7.29	12.76	11.64	14.96	13.05
JPEG compression Q = 50	10.94	9.37	8.29	7.24	12.94	11.49	14.56	13.49
JPEG compression Q = 60	9.18	8.46	7.16	6.34	11.46	10.46	13.76	12.67
JPEG compression Q = 70	10.67	9.37	8.34	7.29	12.49	11.67	14.56	13.06
JPEG compression Q = 80	10.64	9.72	8.34	7.59	12.67	11.57	14.76	13.75
JPEG compression Q = 90	9.18	8.92	7.94	6.38	11.34	10.94	13.08	12.49
JPEG compression Q = 100	10.34	9.34	8.82	7.54	12.64	11.46	14.63	13.94
Median filtering (3 × 3)	9.48	8.37	7.20	6.35	11.94	10.67	13.43	12.37
Median filtering (5 × 5)	10.34	9.45	7.24	6.98	12.67	11.67	14.28	13.76
Wiener filtering (3 × 3)	9.46	8.51	7.34	6.45	11.43	10.46	13.37	12.46
Wiener filtering (5 × 5)	10.49	9.67	8.64	7.15	12.94	11.37	14.08	13.68
Gaussian filtering (3 × 3)	9.57	8.01	7.15	6.68	12.93	10.67	14.59	13.47
Gaussian filtering (5 × 5)	9.48	8.27	7.95	6.57	11.80	10.37	13.94	12.74
Scaling (zoomout = 0.5, zoomin = 2)	10.37	9.35	8.64	7.49	12.37	11.64	14.65	13.48
Scaling (zoomout = 0.25, zoomin = 4)	9.18	8.35	7.03	6.29	11.60	10.68	13.82	12.49
Scaling (zoomout = 2, zoomin = 0.5)	9.46	8.34	7.90	6.58	11.43	10.48	13.91	12.67
Crop 10	10.38	9.46	8.15	7.26	11.94	10.34	14.84	13.28
Crop 20	10.49	9.26	8.64	7.05	12.34	11.67	14.67	13.49
Crop 30	9.28	8.49	7.59	6.34	11.46	10.34	13.50	12.49

Table 4
The SSIM of the embedded watermark various attacks.

Attack	LWT		Proposed		DWT		SWT	
	DB1	DB3	DB1	DB3	DB1	DB3	DB1	DB3
None								
Pepper & salt noise (density 0.3)	34.26	38.56	40.15	44.23	28.45	30.85	24.56	25.59
Pepper & salt noise (density 0.01)	35.12	39.45	41.46	44.56	28.36	31.26	22.96	26.45
Pepper & salt noise (density 0.001)	35.82	36.12	40.56	44.89	28.59	32.48	23.59	26.85
Pepper & salt noise (density 0.005)	35.62	37.45	40.84	45.23	29.14	32.95	24.78	25.48
Speckle noise (var = 0.01)	35.23	36.45	40.26	43.12	29.58	32.18	22.94	27.41
Speckle noise (var = 0.04)	35.89	37.59	40.16	44.89	28.67	31.28	23.82	27.92
Speckle noise (var = 0.4)	34.21	38.15	40.45	44.78	28.45	30.85	24.61	25.38
Gaussian noise (M = 0, var = 0.001)	34.59	39.57	40.21	44.72	29.38	30.57	24.93	25.10
Gaussian noise (M = 0, var = 0.005)	35.85	38.95	40.19	43.52	29.84	31.75	22.48	25.46
Gaussian noise (M = 0, var = 0.5)	34.20	39.56	40.24	43.89	28.39	31.27	23.94	26.48
Rotation (angle 45°)	35.46	39.25	40.16	44.28	29.86	30.86	24.86	25.43
Rotation (angle 2°)	34.42	38.16	40.12	44.21	28.54	30.68	24.71	26.86
Rotation (angle 70°)	35.85	37.16	41.67	43.89	29.43	31.26	23.31	25.49
Rotation (angle 110°)	34.95	36.48	41.21	44.27	28.16	30.29	23.51	26.38
Rotation (angle −50°)	33.57	36.16	40.56	43.56	29.46	32.64	22.16	26.92
Rotation (angle 50°)	33.41	36.45	40.58	43.68	29.87	32.54	24.59	27.81
Histogram Equalization	33.21	37.56	40.56	42.56	29.92	31.62	24.67	25.18
JPEG compression Q = 5	34.59	38.92	41.28	42.01	28.49	32.74	23.05	26.94
JPEG compression Q = 10	35.92	37.58	40.89	43.67	28.57	32.76	24.94	27.86
JPEG compression Q = 20	33.21	38.62	41.57	42.15	28.10	31.68	23.16	26.84
JPEG compression Q = 30	34.26	36.21	40.69	42.59	28.36	30.29	22.49	25.93
JPEG compression Q = 40	34.49	37.56	40.16	44.67	28.15	30.45	23.82	26.53
JPEG compression Q = 50	35.64	36.99	40.58	44.28	29.64	30.96	23.76	26.48
JPEG compression Q = 60	33.59	38.64	40.28	44.75	28.35	31.92	22.64	27.94
JPEG compression Q = 70	33.48	39.56	40.21	43.12	29.43	32.52	24.16	27.56
JPEG compression Q = 80	34.75	38.64	40.76	44.52	29.51	32.84	24.53	27.64
JPEG compression Q = 90	35.62	37.30	41.94	43.01	28.38	31.29	23.90	25.63
JPEG compression Q = 100	34.21	38.64	41.64	43.82	28.92	30.16	22.46	25.97
Median filtering (3 × 3)	35.28	37.41	40.57	44.28	28.16	31.28	22.72	25.84
Median filtering (5 × 5)	34.65	39.06	40.64	43.67	29.38	32.94	24.76	26.83
Wiener filtering (3 × 3)	35.20	38.46	40.27	44.27	29.84	31.28	22.67	26.81
Wiener filtering (5 × 5)	35.28	39.24	41.68	44.19	28.96	32.61	22.89	25.08
Gaussian filtering (3 × 3)	33.61	38.62	41.56	43.17	29.82	30.69	22.16	27.09
Gaussian filtering (5 × 5)	33.84	37.45	40.28	43.89	28.74	30.49	23.84	26.18
Scaling (zoomout = 0.5, zoomin = 2)	34.20	36.26	40.96	44.68	29.68	31.23	23.64	26.19
Scaling (zoomout = 0.25, zoomin = 4)	34.39	38.20	40.82	42.95	29.38	32.82	23.61	27.83
Scaling (zoomout = 2, zoomin = 0.5)	35.64	39.54	41.20	43.61	28.49	31.27	24.53	27.49
Crop 10	33.29	37.12	41.26	44.38	28.81	31.06	22.86	26.35
Crop 20	33.82	38.64	41.85	42.61	28.64	32.08	24.06	26.39
Crop 30	34.61	39.41	41.24	44.50	29.38	32.82	23.94	25.18

$$NC(k,\bar{k}) = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}[k(i,j) - \mu_k]\left[\bar{k}(i,j) - \mu_{\bar{k}}\right]}{\sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}[k(i,j) - \mu_k]^2}\sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}\left[\bar{k}(i,j) - \mu_{\bar{k}}\right]^2}}$$

where N and M represent the number of pixels in the watermark, $k, \bar{k}$ indicates to the original watermark and the extracted watermark, The correlation coefficient can be between −1 and 1. If the NC value is near +1, then the extracted watermark is strongly correlated.

4. Bit Error Ratio

This is one of the parameter used for calculating the error value of the retrieved watermark image. The BER can be calculated as follows:

$$BER = \frac{1}{P}\sum_{i=1}^{N}|w_{out}(i) - w_{in}(i)|$$

Where w_{out} is the extracted watermark, w_{in} is the original watermark and P is the size of the watermark.

To analysis the performance of the proposed system, it is compared with various techniques by using the performance metrics which are mentioned above. This is shown in the above tables and graphs.

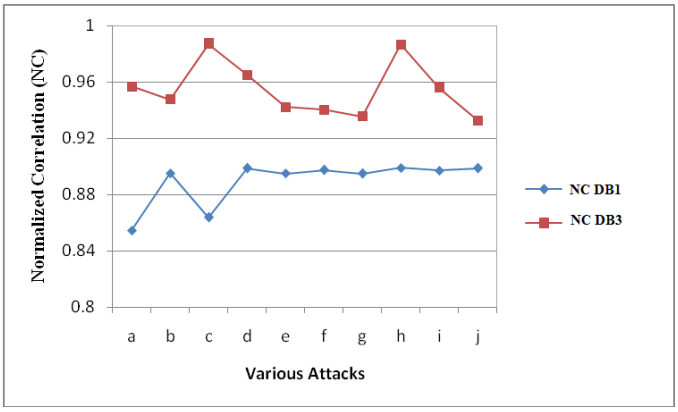


Fig. 8 Normalized Correlation on DB1 and DB3

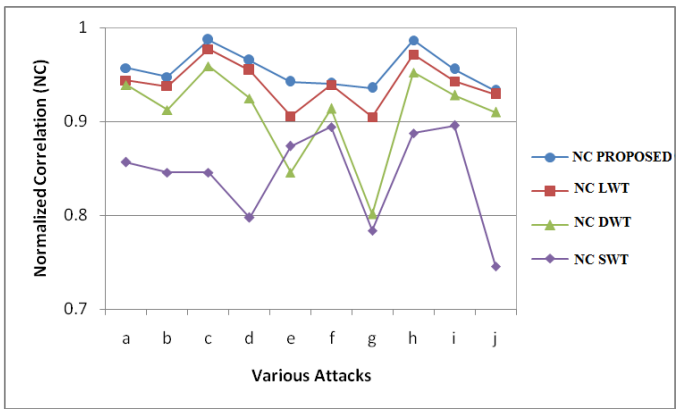


Fig. 2. Normalized Correlation on Various Wavelet Transform

V. CONCLUSION

In this paper, a new method for reversible watermarking is developed and used. This method is based on Discrete Wavelet Transform (DWT), Arnold Transform (AT), Chaotic Based Encryption (CBE), Rationale of Prediction Using a Rhombus Pattern and Sorting the Prediction Error algorithms. first the input image is pre-processed by using Arnold transform and then DWT is applied onto the pre-processed image. And then the DWT transformed image is divided into 4 sub blocks. After dividing the sub blocks, the next step is to shuffle the sub blocks to differentiate between the homogenous and non homogenous blocks. After shuffling the next step is to apply the prediction and find the prediction error. Then the prediction error value is sorted to generate the places for embedding the watermark bit. At this stage, the watermark image is get from the user and these images are also pre-processed and encrypted. These encrypted watermark image pixels are embedded with the sorted prediction error values. Finally, the homogenous and non homogenous blocks are combined and the four sub blocks are also combined to generate the single image. After that inverse DWT and Arnold transform is applied to produce the watermarked image. The performance of the proposed method is compared with the DWT,SWT and LWT algorithm. The experimental result shows that the proposed watermarking technique performs well than the DWT,SWT and LWT based watermarking approach. Various wavelet families such as DB1 and DB3 are used in this process. During this process DB3 performs well than DB1.

References

[1] Potdar Vidyasagar M, Song Han, Elizabeth Chang "A Survey of Digital Image Watermarking Techniques", 3rd IEEE International Conference on Industrial Informatics (INDIN) 2005, pp-709-716.

[2] Shuang Yi, Yicong Zhou and Zhongyun Hua, "Reversible Data Hiding in Encrypted Images using Adaptive Block-Level Prediction-Error Expansion", Journal of Signal Processing: Image Communication, pp. 1-26, 2018.

[3] Yan Qi & Liping Liu, "Reversible Watermarking Algorithm based on Prediction Error Expansion for Color Image", Proceedings of International Conference on Image Processing, pp. 102-106, 2017.
[4] Chan C.K, Cheng L.M. "Hiding data in images by simple LSB substitution", Pattern Recognition2004,pp469-474.
[5] Mathkour Hassan, Ghazy M.R., Abdulaziz Al Muharib, Ibrahim Kiady "A Novel Approach for Hiding Messages in Images", International Conference on Signal Acquisition and Processing, 2009, pp -89 – 93.
[6] J. Feng, I. Lin, C. Tsai, Y. Chu, "Reversible watermarking: current status and key issues", Int. Journal, Vol. 2, No. 3, pp. 161–170, 2006.
[7] Qing Liu & Jun Ying, "Gray scale Image Digital Watermarking Technology Based on Wavelet Analysis", IEEE Symposium on Electrical & Electronics Engineering, pp. 618-621, 2012.
[8] Khushboo Pawar, Bhawana Pillai, Dr. Sadhna K Mishra, "Conditional Local Prediction Based Image Watermarking", International Journal of Engineering Technology and Applied Science, Vol. 2 , No.3, March-2016.
9] Kuo-Ming Hung, "A Novel Robust Watermarking Technique Using IntDCT Based AC Prediction", WSEAS Transactions on Computers, Vol. 7, No. 1, pp. 16-24, 2008.
[10] Jinal H. Mehta & Vishakha Kelkar, "Comparison of Reversible Watermarking using Prediction Error Expansion and Prediction Error Expansion Considering Region of Interest for Medical images", IEEE 2nd International Conference for Convergence in Technology,2017.
[11] Vleeschouwer C. De, Delaigle J. F and Macq B., "Circular interpretation of bijective transformations in lossless watermarking for media asset management", IEEE Transaction on Multimedia, 5(1), 2003, pp.97-105.
[12] Zou D.K., Shi Y.Q., Ni Z.C., Su W, "A semi-fragile lossless digital watermarking scheme based on integer wavelet transform", IEEE Transactions on Circuits and Systems for Video Technology16(10) (2006) pp.1294-1300.
[13] Ni Z.C., Shi Y.Q., Ansari N., Su W.S, Sun Q.B., Lin X., "Robust Lossless Image Data Hiding," Multimedia and Expo, 2004, ICME '04, vol. 3, June 2004, pp. 2199–2202.
[14] Ni Z.C., Shi Y.Q., Ansari N., Su W.S, Sun Q.B., Lin X., "Robust lossless image data hiding designed for semi-fragile image authentication," IEEE Transactions on Circuits and Systems for Video Technology 18 (4) (2008)pp. 497–509.
[15] Huang Hui-Yu, Chang Shih-Hsu "A 9/7 wavelet-based lossless data hiding" 978-1- 4244-9915-1/11/$26.00 ©2011 IEEE.
[16] Zhao ZF, Luo H, Lu ZM, Pan JS. "Reversible data hiding based on multilevel histogram modification and sequential recovery". International Journal of Electronics and Communications (AEÜ) 2011; 65(10):pp.814–26.
[17] L. Wu, W. Deng, J. Zhang, and D. He. 2009. Arnold transformation algorithm and anti-Arnold transformation algorithm. Proc. of 1st International Conference on Information Science and Engineering (2009), 1164–1167.
[18] C.-T. Hsu and J.-L. Wu, ''Hidden digital watermarks in images,'' IEEE Trans. Image Process., vol. 8, no. 1, pp. 58–68, Jan. 1999.
[19] Y. K. Lin, ''A data hiding scheme based upon DCT coefficient modification,'' Comput. Standards Int., vol. 36, no. 56, pp. 855–862, 2014.
[20] J. Tian, "Reversible data embedding using a difference expansion," IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 8, pp. 890–896, Aug. 2003.
[21] L. H. J. Kamstra and A. M. Heijmans, "Reversible data embedding into images using wavelet techniques and sorting," IEEE Trans. Image Process., vol. 14, no. 12, pp. 2082–2090, Dec. 2005.
[22] D. M. Thodi and J. J. Rodriguez, "Expansion embedding techniques for reversible watermarking," IEEE Trans. Image Process., vol. 16, no. 3, pp. 721–730, Mar. 2007.