

Chapter 3

Watermarking Scheme for Image Authentication

In this chapter, two image watermarking schemes RWS-WM and RWS-CA have been developed for image authentication. In RWS-WM, a weighted matrix (WM) based reversible watermarking scheme (RWS) has been developed for image authentication. The cover image is decomposed into three different color blocks and then entry-wise-multiplication operation has been performed. Repetition of embedding procedure with modified WM has been performed to achieve high payload and security. In RWS-CA, a CA based watermarking scheme has been proposed in sub-sampled image. The sustainability against some standard geometric attacks have been analyzed for both the schemes. It is observed that the schemes are robust against such attacks. The proposed schemes can also detect tampering in extracted watermark in some cases.

3.1 RWS-WM: Weighted Matrix Based RWS ¹

High capacity, secure and reversible watermarking scheme for image authentication and tamper detection is still an important area of research. In this investigation, a weighted matrix based RWS has been proposed using color image. This RWS provides image authentication and tamper detection. At first, the original image (CI) is partitioned into (3×3) pixel blocks. Then, these blocks are decomposed into R, G, B color components. After that, the sum of entry-wise-multiplication operation has been performed using a modified weighted matrix (WM) to embed the watermark (W). The watermark embedding locations are stored in an index file (InF) to enhance security, increase embedding capacity, maintain good visual quality, achieve reversibility and confirm authenticity. The proposed watermarking scheme not only perform authentication and tamper detection but also improves both data embedding capacity up to 8.00 (bpp) as well as increase visual quality measured by PSNR 50.03 (dB). Finally, the scheme is compared with other existing state-of-the-art methods and gives a reasonably better performance in terms of visual quality and embedding capacity. RWS-WM scheme has been evaluated through various steganographic analysis and it has been observed that the scheme is secure and robust against various geometric attacks.

The watermark embedding and extraction of RWS-WM scheme have been described in two subsections (3.1.1 and 3.1.2).

¹Published in **Multimedia Tools and Application**, Springer: **Impact Factor: 1.541** September 2018, Volume 77, Issue 18, pp 23073-23098, with title *Weighted matrix based reversible watermarking scheme using color image*.

3.1.1 Watermark Embedding Phase

In this investigation, a reversible image watermarking scheme has been proposed for color image using modified weighted matrix. At first, the color original image (CI) is partitioned into (3×3) pixel blocks. Then Red (R), Green (G) and Blue (B) color components are decomposed from the original (3×3) pixel blocks and the respective values are stored in three separate matrices CI_R , CI_G and CI_B respectively. The watermark image (W) is considered as the secret message $M = m_1|m_2|m_3|m_4|\dots|m_n$, where m_i corresponds to 4 bits secret information of W. The schematic diagram of RWS-WM embedding process is shown in Fig. 3.1. A (3×3)

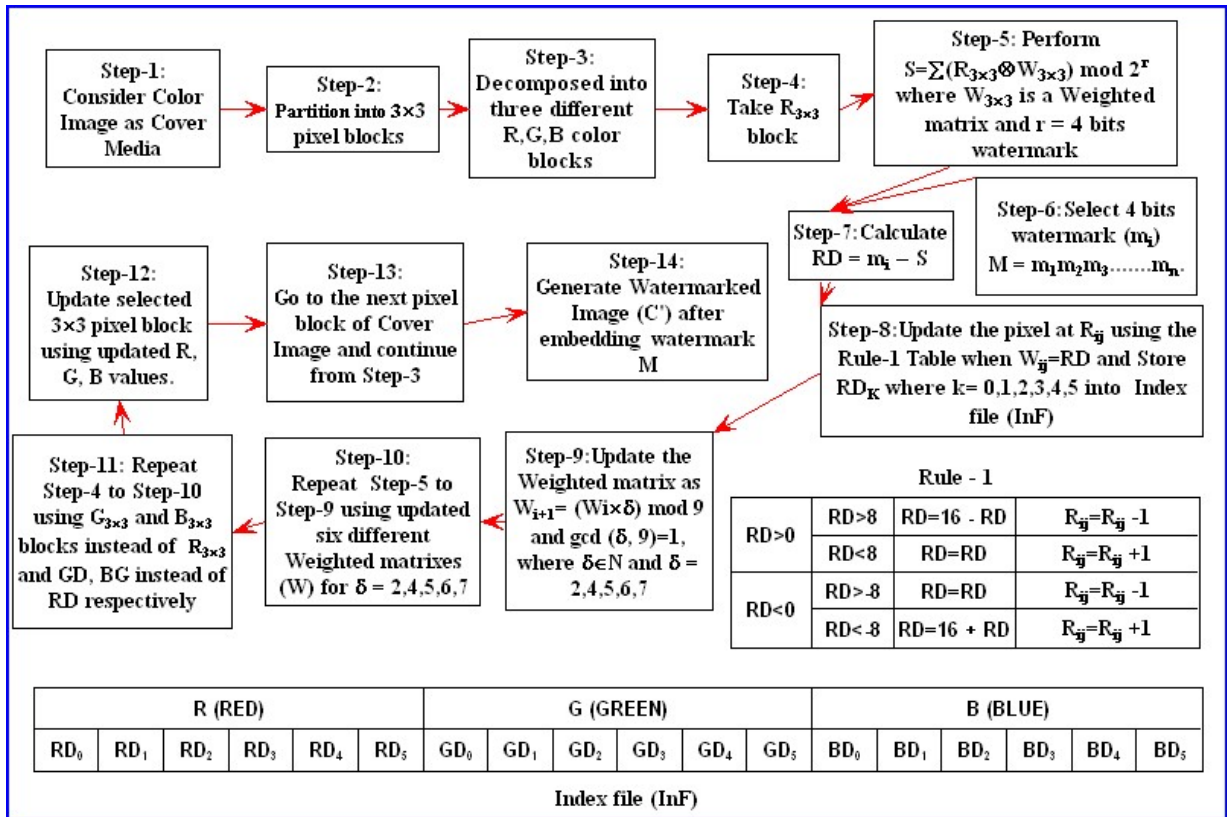


Figure 3.1: Schematic diagram of watermark embedding phase in RWS-WM

integer WM has been shared between sender and receiver before data communication. The criterion of forming WM is that each element of the matrix is arbitrarily allotted a value from the combination $(1, 2, \dots, 2^{r-1})$ and each element appears at least once in WM. Here, r denotes the number of secret bits those will be embedded per operation of each color block. In this process, r data bits $(b_1 b_2 \dots b_r)$ are embedded into image block CI_R using the equation (3.1).

$$d = dec((b_1 b_2 \dots b_r)_2) - (SUM(CI_R \otimes WM)) \pmod{2^r} \quad (3.1)$$

Algorithm 3.1: RWS-WM: Watermark Embedding Algorithm

Input : Cover Color Image (CI), Watermark (W), $M = m_1 m_2 m_3 m_4 \dots Length(M/4)$, Where $m_i = 4$ -bit secret message.
Output: Watermarked Image (WI), Index file (InF)

```

1  Algorithm Embedding():
2  for (y = 0; y < n/3; y++) do
3  for (x = 0; x < m/3; x++) do
4  for (r = 0; r < 6; r++) do StoreDataBitsInPixelBlock(P_RED,r,x,y);
5  for r = 0; r < 6; r++ do StoreDataBitsInPixelBlock(P_GREEN,r,x,y);
6  for r = 0; r < 6; r++ do StoreDataBitsInPixelBlock(P_BLUE,r,x,y);
7  end
8  end
9  GenerateWatermark(P_RED, P_GREEN, P_BLUE);

10 Function StoreDataBitsInPixelBlock (P,r,x,y):
11  recalculate = false;
12  Consider WM = Wr;
13  do
14  for (i = 0; i < 3; i++) do
15  for (j = 0; j < 3; j++) do sum = sum + (P[3 * y + i][3 * x + j] * WM[i][j]);
16  end
17  S = Mod(sum, 16);
18  d = Binary2Decimal(Mk) - S;
19  if d > 8 then d = 16 - d;
20  if d < -8 then d = 16 + d;
21  modD = Abs(d);
22  for (i = 0; i <= 2; i++) do
23  for (j = 0; j <= 2; j++) do
24  if (WM[i][j] == modD) then
25  if (d > 0) then
26  if (P[3 * y + i][3 * x + j] == 255) then
27  P[3 * y + i][3 * x + j] = 254;
28  recalculate = true;
29  WriteIndexFile("+", j, i); // store position of i and j in InF
30  continue;
31  end
32  P[3 * y + i][3 * x + j]++;
33  WriteIndexFile("-", j, i); // store position of i and j in InF
34  continue;
35  end
36  if (d < 0) then
37  if (P[3 * y + i][3 * x + j] == 0) then
38  P[3 * y + i][3 * x + j] = 1;
39  recalculate = true;
40  WriteIndexFile("-", j, i); // store position of i and j in InF
41  continue;
42  end
43  P[3 * y + i][3 * x + j]--;
44  WriteIndexFile("+", j, i); // store position of i and j in InF
45  continue;
46  end
47  end
48  end
49  end
50  while (recalculate==true);
51  WMr+1 = (WMr × μr) mod 9, where μr = 2, 4, 5, 7, 8 // gcd(μr, 9)

```

where \otimes denotes entry-wise multiplication operator. The $SUM()$ function represents the modular summation of all the entries of matrix $(CI_R \otimes WM)$. If d is equal to zero then CI_R remains

unchanged; otherwise, CI_R is modified to CI'_R which satisfy the equation (3.2).

$$SUM(CI'_R \otimes WM) = dec(b_1 b_2 \dots b_r) \text{ mod } 2^r \quad (3.2)$$

In this scheme, color image blocks are used and above operations are performed in three different color blocks separately. Finally, three modified blocks CI'_R , CI'_G and CI'_B are combined to get modified image block CI' . The final watermarked image (WI) can be achieved by combining all these blocks. Before each entry-wise multiplication operation, WM is updated using the equation (3.3).

$$WM_{i+1} = (WM_i \times \mu) \text{ mod } 9 \quad (3.3)$$

where $i = 0, 1, 2, \dots, 2^r$ and $gcd(\mu, 9) = 1$. The initial weighted matrix WM and μ are shared with the receiver during data communication. The R, G, B matrix value is modified at the d^{th}

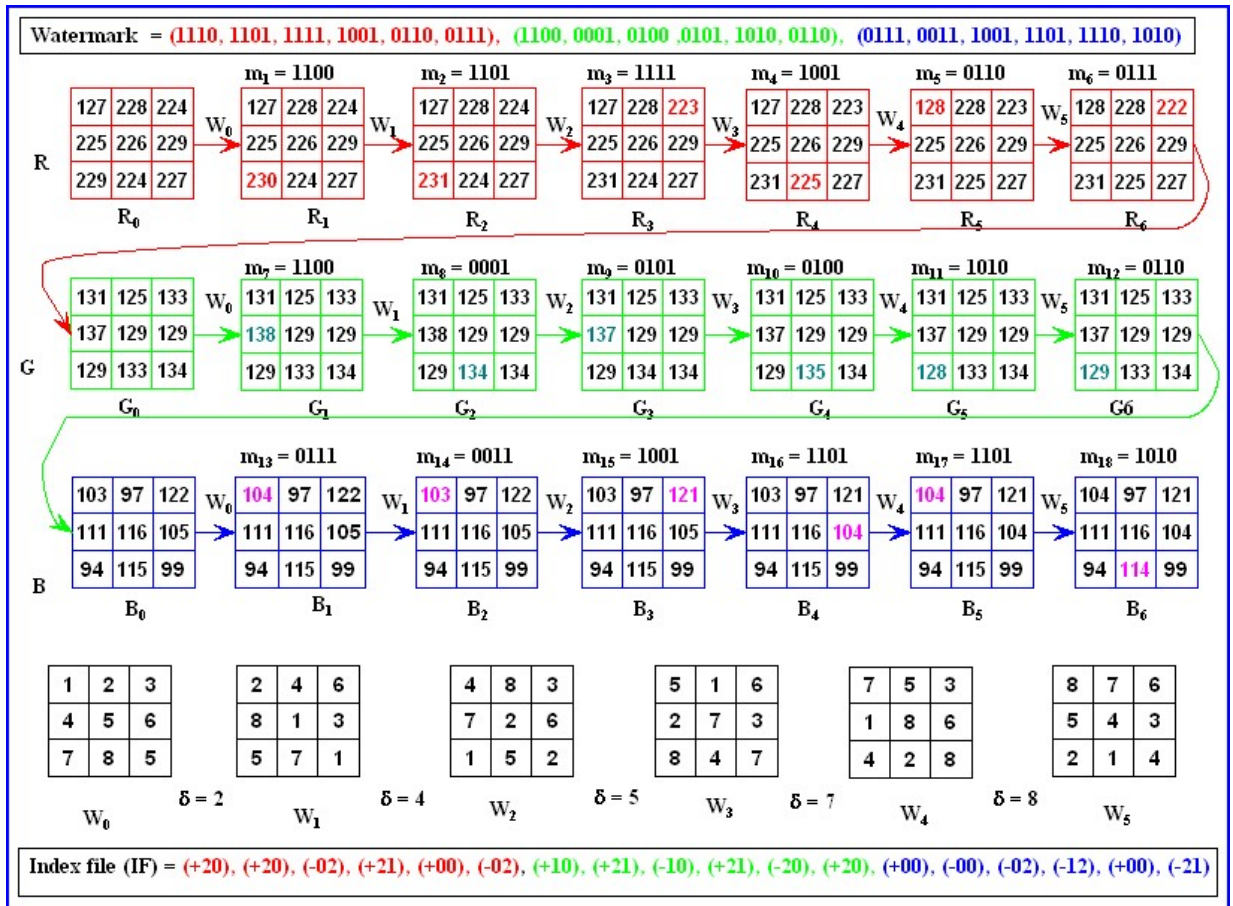


Figure 3.2: Numerical illustration of watermark embedding phase in RWS-WM

position of the weighted matrix, by increasing or decreasing pixel value. That means, if CI'_R increases by one then the modular sum $SUM(CI_R \otimes WM)$ is increased by WM and if CI_R

decreases by one then the modular sum $SUM(CI_R \otimes WM)$ is decreased by WM . In extraction phase, the receiver only needs to calculate $SUM(CI'_R \otimes WM) \pmod{2^r}$ to derive b_1, b_2, \dots, b_r .

The embedding algorithm of RWS-WM is presented in Algorithm 3.1 containing two procedures: *Embedding()* and *StoreDataBitsInPixelBlock()*. The detail process is described as follows:

Embedding() : Here, *StoreDataBitsInPixelBlock()* function is called thrice for three different color pixel blocks. Then, three updated color blocks are combined by function *GenerateWatermark()* to form the color watermarked image (WI).

StoreDataBitsInPixelBlock(): Here the sum of entry-wise multiplication of WM with Pixel block (PB) are stored in the variable *sum*. Then $\text{mod}(D)$ location is found in WM and d values are checked for negative or positive sign. After that, increment or decrement has been performed accordingly and call *WriteIndexFile()* to store the position of the m and n with sign in InF. Unique weighted matrices are derived with different μ values, when $\text{gcd}(\mu, 9) = 1$.

The numerical illustration has been shown in Fig. 3.2. The R, G, B blocks are updated from lower number (0) to higher number (6) after embedding 4-bit secret data in each operation using specific WM. The weighted matrices are updated from lower number (0) to higher number (6) using equation (3.3). The μ as 2, 4, 5, 7 and 8 are chosen to derived unique WM. Seventy two bits watermark information are embedded within a (3×3) pixel block where each R, G, B component separately contain twenty four bits. As a result, a (3×3) color image pixel block contain 72 bits watermark information.

3.1.2 Watermark Extraction and Recovery Phase

At the time of watermark extraction and original image reconstruction, watermarked image is partitioned into (3×3) blocks. Then, the blocks are decomposed into R, G, B color components. The schematic diagram of watermark extraction and image recovery process is depicted in Fig. 3.3. The extraction process started from the CI_B block. Repeat the extraction process six times within each color block using six different WM. The extraction and image recovery algorithm are presented in Algorithm 3.2.

The extraction of RWS-WM has *Extraction()*, *ExtractDataBitsFromPixelBlock()* and *GetPreviousPixelMatrixUsingIndexFile()*.

The *GetPreviousPixelMatrixUsingIndexFile()* function simply performs sum of entry-

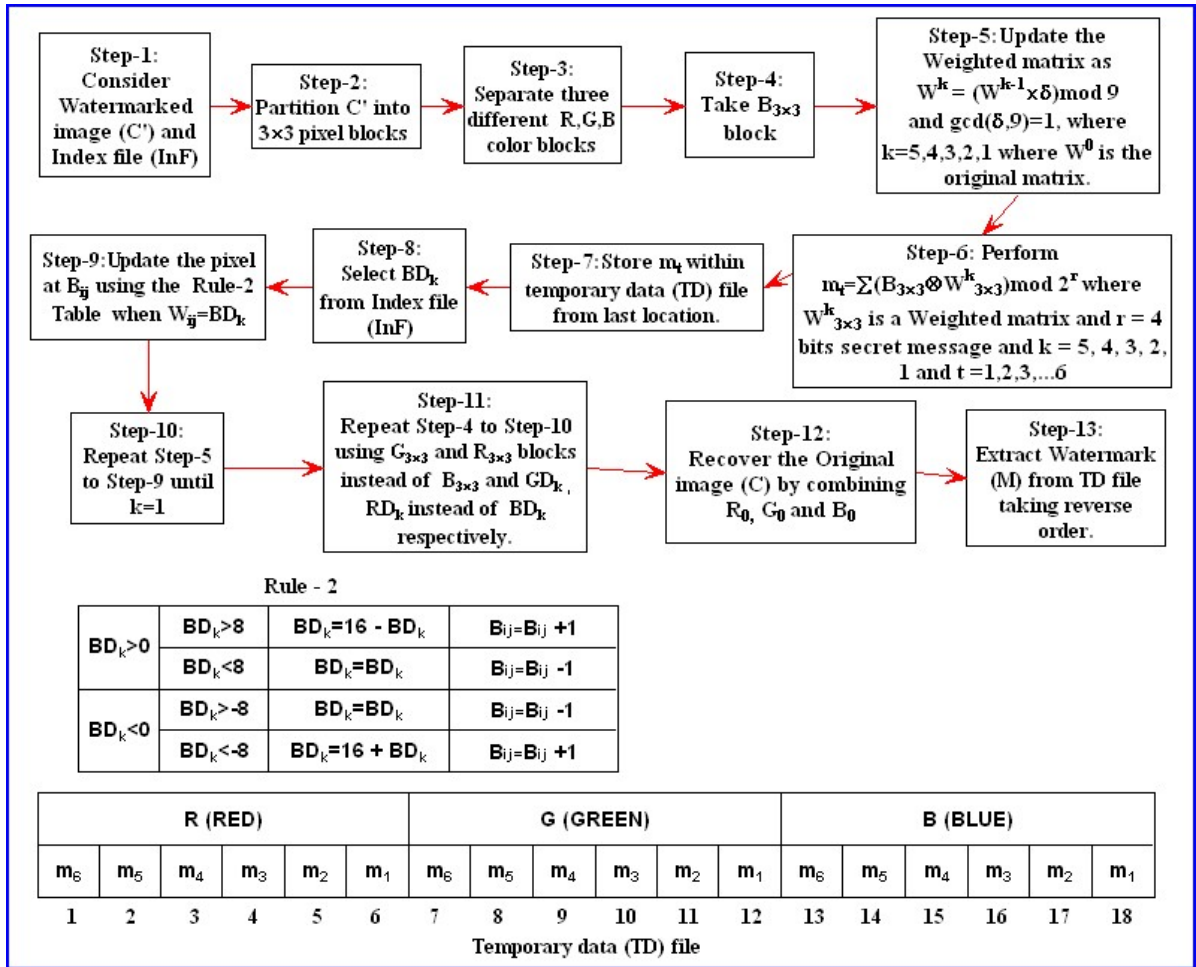


Figure 3.3: Schematic diagram of watermark extraction phase in RWS-WM

wise multiplication with pixel block (CI_B) and weighted matrix ($WM = WM_r$) where $r =$ starts from 5 to 0 and store extracted data in a temporary file (TD). After that, the *GetPreviousPixelMatrixUsingIndexFile()* function is called to get the previous pixel value by using InF. Then WM_r has been updated using $WM_{(r-1)} = (WM_r \times \mu_r) \bmod 9$, where $\mu_r = 8, 7, 5, 4, 2$. After performing extraction in blue pixel block, the same extraction procedure has been performed in green and in red pixel blocks. Cover image is reconstructed using *GenerateCoverImage()* function by combining each color blocks. Finally, secret information are appended in reverse order from TD file after completion of data extraction from one block and original secret information are obtained by concatenating one after another. Numerical illustration of the watermark extraction and the recovery of cover image are presented in Fig. 3.4.

Algorithm 3.2: RWS-WM: Watermark Extraction Algorithm

```

Input : Watermarked Color Image (WI), Index File (InF);
Output: Watermark (W); Original Cover Image (WI)

1 Algorithm Extraction-main():
2   for ( $y = 0; y < n/3; y ++$ ) do
3     for ( $x = 0; x < m/3; x ++$ ) do
4       for ( $r = 5; r >= 0; r --$ ) do
5         | ExtractDataBitsFromPixelBlock( $P_{BLUE}, x, y$ );
6       end
7       for ( $r = 5; r >= 0; r --$ ) do
8         | ExtractDataBitsFromPixelBlock( $P_{GREEN}, x, y$ );
9       end
10      for ( $r = 5; r >= 0; r --$ ) do
11        | ExtractDataBitsFromPixelBlock( $P_{RED}, x, y$ );
12      end
13    end
14  end
15  GenerateCoverImage( $P_{RED}, P_{GREEN}, P_{BLUE}$ ); //Original color image extraction

16 Procedure ExtractDataBitsFromPixelBlock ( $P, x, y$ ):
17   Consider  $WM = W_r$ ;
18   // Elementary Matrix Multiplication of  $(3 \times 3)$  pixel matrix with  $(3 \times 3)$  weighted matrix
19   for ( $i = 0; i < 3; i ++$ ) do
20     | for ( $j = 0; j < 3; j ++$ ) do
21       |  $sum = sum + (P[3 * y + i][3 * x + j] * WM[i][j])$ ;
22     | end
23   end
24    $S = Mod(sum, 16)$ ;
25    $M = Decimal2Binary(S)$ ;
26   TD = StoreInDataFile(M); // Extracted data stored in TD file
27   ;
28   GetPreviousPixelMatrixUsingIndexFile( $P, x, y$ );
29    $WM_{r-1} = (WM_r \times \mu_r) \bmod 9$ , where  $\mu_r = 8, 7, 5, 4, 2$ ;
30   // End Procedure

29 Procedure GetPreviousPixelMatrixUsingIndexFile ( $x, y$ ):
30   // Get the pixel matrix in previous stage using Index File F
31   posX = Get Position of x from index file (InF);
32   posY = Get Position of y from index file (InF);
33   sign = Get +1 or -1 sign from index file (InF);
34    $P[y * 3 + posY][x * 3 + posX] += sign$ ;
35   // End Procedure

```

3.1.3 Experimental Results and Comparison

In this section, RWS-WM have been tested using standard benchmark images shown in Fig. 2.3 and evaluated using various evaluation metrics. The experimental results with comparison are given below:

3.1.3.1 Quality Measurement and Payload Analysis

In this section, some subjective and objective quality evaluation are performed. The color watermark images with different size, used for watermark embedding are shown in Fig. 3.5. Here, the distortion is measured by means of two parameters namely, Mean Square Error (MSE) and Peak Signal to Noise Ratio (PSNR). The MSE is calculated using equation (2.5). The differ-

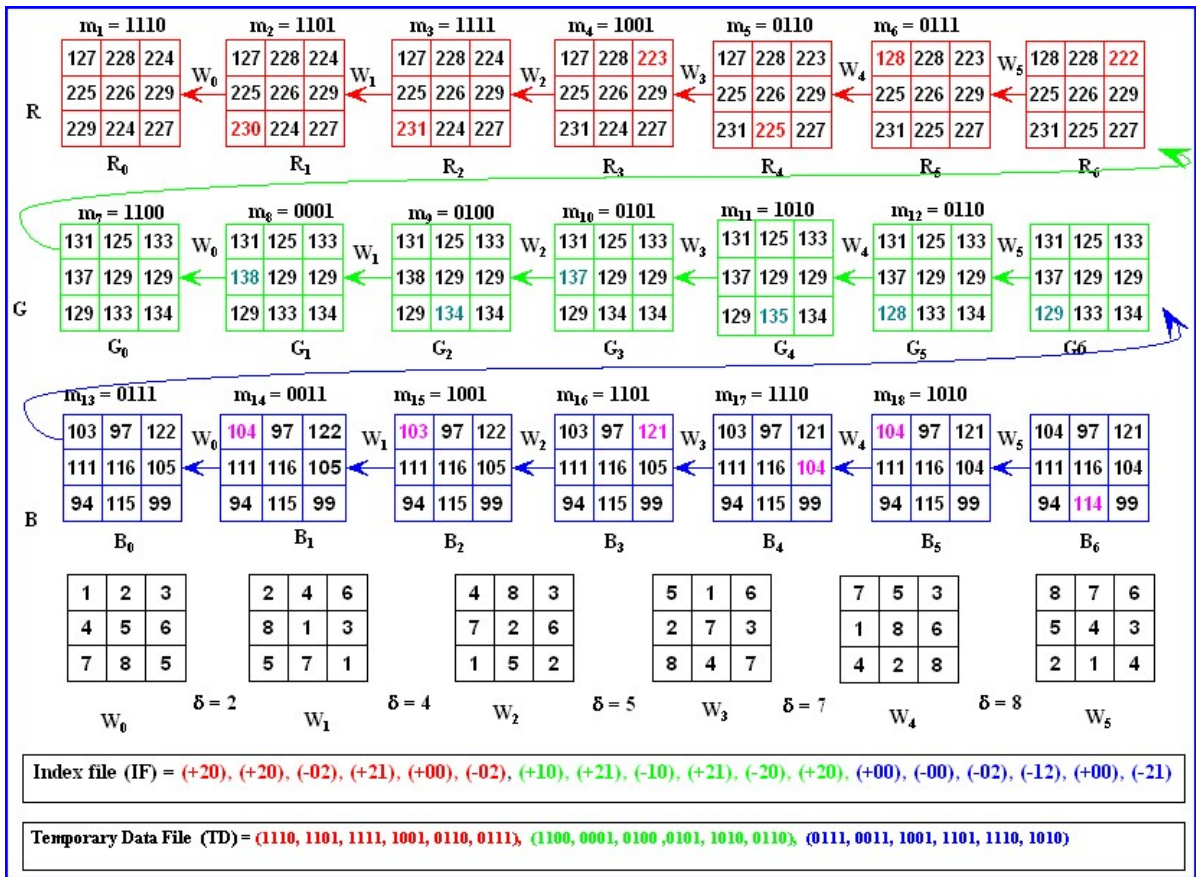


Figure 3.4: Numerical illustration of watermark extraction phase in RWS-WM.

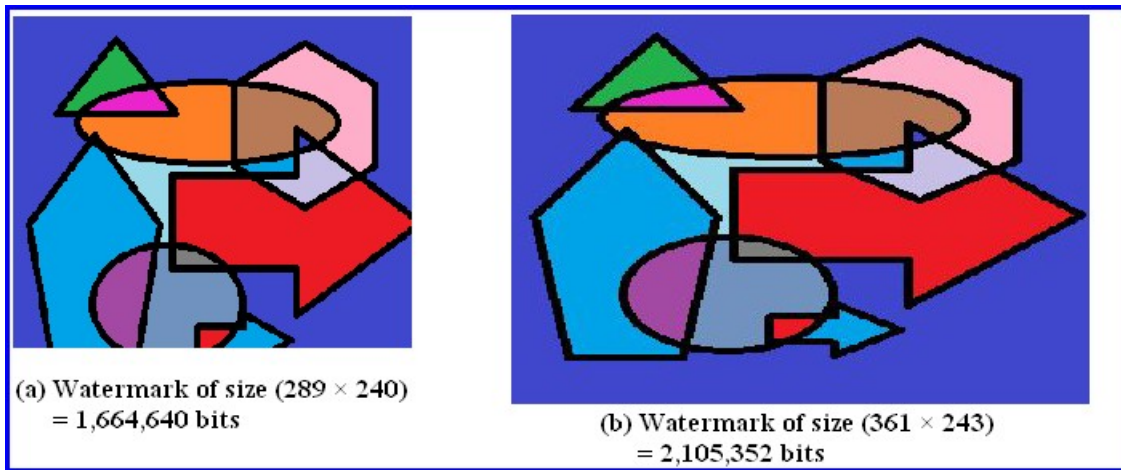


Figure 3.5: Color watermark image (logo) with different size used in RWS-WM

ence between the CI and WI are assessed by the Peak Signal to Noise Ratio (PSNR) using equation (2.6). Higher the values of PSNR between two images indicates better the quality of the watermarked image and very similar to the cover image where as low PSNR demonstrates the opposite.

The experimental results are tabulated in Table 3.1. It shows PSNR, Payload and statistical

Table 3.1: PSNR (dB), Payload (bpp), SD and CC results of different datasets in RWS-WM

Cover Image (CI)	Watermark (W) (bits)	PSNR (dB)	Payload (bpp)	SD (σ) of		CC (ρ) of CI & WI
				CI	WI	
Lena	1,664,640	51.17	6.35	129.1583	129.1627	1.0000
	2,105,352	50.21	8.03		129.1628	0.9999
Baboon	1,664,640	51.28	6.35	129.0957	129.0991	1.0000
	2,105,352	50.22	8.03		129.0981	0.9999
Barbara	1,664,640	51.03	6.35	144.6319	144.6295	1.0000
	2,105,352	50.21	8.03		144.6404	1.0000
Aeroplane	1,664,640	50.07	6.35	124.4987	124.5064	0.9998
	2,105,352	50.07	8.03		124.5084	0.9999
Parrot	1,664,640	51.68	6.35	161.0205	161.0282	1.0000
	2,105,352	50.20	8.03		161.0408	1.0000
Boats	1,664,640	48.08	6.35	162.7456	162.5703	0.9819
	2,105,352	49.52	8.03		162.4684	0.9769
Monarch	1,664,640	49.64	6.35	129.3781	129.3889	0.9998
	2,105,352	48.13	8.03		129.3835	0.9998
Pens	1,664,640	50.07	6.35	169.8330	169.8393	1.0000
	2,105,352	49.22	8.03		169.8405	1.0000
Soccer	1,664,640	48.75	6.35	150.2585	150.2683	0.9999
	2,105,352	47.91	8.03		150.2720	0.9999
Yacht	1,664,640	48.68	6.35	158.1548	158.1575	0.9999
	2,105,352	47.26	8.03		158.1652	0.9998
Flower	1,664,640	51.68	6.35	187.2133	187.2084	1.0000
	2,105,352	50.21	8.03		187.2099	1.0000
Girl	1,664,640	50.27	6.35	146.7087	146.7061	1.0000
	2,105,352	49.21	8.03		146.7236	0.9999
ucid00148	1,664,640	51.76	6.35	123.6534	123.6534	0.9825
	2,105,352	50.34	8.03		123.6534	0.9782
ucid00354	1,664,640	51.68	6.35	117.8965	117.9832	0.9856
	2,105,352	50.48	8.03		117.9652	0.9734
ucid00617	1,664,640	50.12	6.35	172.8763	172.8937	1.0000
	2,105,352	49.56	8.03		172.8846	0.9987
im0381	1,664,640	47.36	6.35	104.5732	104.6356	0.9835
	2,105,352	46.18	8.03		104.6232	0.9742
im0015	1,664,640	47.31	6.35	106.5673	106.9510	0.9686
	2,105,352	46.13	8.03		107.0761	0.9532
im00051	1,664,640	48.26	6.35	108.3436	108.3428	1.0000
	2,105,352	47.04	8.03		108.4158	0.9984
bardowl	1,664,640	48.56	6.35	154.8662	154.9956	0.9642
	2,105,352	47.78	8.03		154.9754	0.9537
bluheron	1,664,640	49.36	6.35	149.5743	149.5700	0.9987
	2,105,352	48.49	8.03		142.4653	0.9783
jerusalem	1,664,640	49.46	6.35	148.4562	148.5749	0.9734
	2,105,352	48.75	8.03		148.5556	0.9685

analysis results of RWS-WM. It is observed that after hiding 2, 105, 352 bits (i.e., (361×243) pixels) as secret watermark within the original image of size (513×513) pixel, the PSNR is nearer to 50.07 dB for Lena image taken from SCI-SIPI [90] image database. The size of all input images are adjusted into (513×513) from (512×512) by repeating the value of the last row and column to get divisible by the block size. In general, for $(T \times T)$ block size, the cover image is adjusted in such a way that it can be divisible by T . The proposed scheme has been tested using different block size that is (2×2) , (3×3) , (4×4) , $(5 \times 5) \dots (T \times T)$ where, $T = 512$ shown in Table 3.2. It has been observed that the PSNR and the size of the InF in (3×3) block are reasonably better with respect to other block size. The scheme has been developed to accommodate any image block with same size WM. But (3×3) block performs better than other block size which motivates us to work with (3×3) block. This is not the state-of-the-art to use (3×3) blocks but experimental results encourage to use (3×3) block in color image watermarking. The graph in Fig. 3.6 between PSNR versus different block size of four benchmark datasets illustrated that (3×3) size is acceptable for this approach. In this experiment, 100 images are considered from each benchmark image databases [26] [90] [89] [61]. The average PSNR (dB) of each 20, 40 and 100 images are presented in Table 3.3 which is nearer to 50 dB for USC-SIPI [90] and UCID [61] image database. In HDR [26] and STARE [89] databases, the PSNR is around 48 dB and 46 dB respectively. From this experiment, it is observed that for uniform and smooth images PSNR (dB) will be decreased compared with non-uniform images when embedding capacity is increased. Figure 3.7 shows the graphical comparison of four different benchmark datasets with 100 images, in terms of visual quality measured by PSNR. It has been observed that USC-SIPI [90] dataset performs better results.

The embedding capacity or payload (P) is calculated in terms of bits per pixel (bpp) using equation (2.7). The maximum payload (P) of this RWS-WM scheme is

$$P = \frac{72 \times \text{no of blocks}}{(513 \times 513) \text{ pixel}} = \frac{72 \times \frac{513}{3} \times \frac{513}{3}}{(513 \times 513)} \text{ bpp} = \frac{72 \times 513 \times 513}{(3 \times 3 \times 513 \times 513)} \text{ bpp} = \frac{72}{9} \text{ bpp} = 8.0 \text{ bpp}$$

To measure the computational complexity, the cover image of size $(M \times N)$ is considered to measure the computational complexity and the watermarking process embeds seventy two bits watermark within a (3×3) pixel block, where in each color block is repeated 6 times. Time complexity of both embedding and extraction algorithms are $\mathcal{O}(MN)$. The execution time is minimized by applying threading concept in the coding section where only 3 seconds time are taken to embed (361×243) color image (i.e., 2, 105, 352 bits) as watermark within

Table 3.2: Comparison with different block size to measure PSNR (dB) in RWS-WM

Block Size	PSNR (dB)	Size of WM	No of repetition for different μ (Max)
2×2	48.89	2×2	3
3×3	50.03	3×3	8
4×4	48.84	4×4	15
5×5	48.25	5×5	24
6×6	47.73	6×6	35
7×7	46.89	7×7	48
...
$M \times M$...	$M \times M$	$M^2 - 1$

Table 3.3: Comparison of PSNR (dB) with different benchmark image datasets in RWS-WM

Database	Size of Image	Number of Image	Average PSNR
USC-SIPI [90]	512×512	20	50.19
		40	49.88
		100	50.11
UCID [61]	512×512	20	49.52
		40	48.89
		100	46.44
STARE [89]	512×512	20	34.87
		40	28.23
		100	26.89
HDR [26]	512×512	20	38.08
		40	36.01
		100	34.88

(513×513) original color image (6, 291, 456 bits) during embedding and 2 seconds are taken during extraction in a standard 64-bit Core i3 machine with 4 GB RAM.

The comparison in terms of PSNR (dB) and embedding capacity (bpp) with Ni et al. [60], Hwang et al. [33], Hu et al. [32], Luo et al. [58], Abadi et al. [1], Yang et al. [100], Jung et al. [40], Jana B. [35], Kuo et al. [45] and Soleymani & Taherinia [77] are presented in Table 3.4. It is observed that RWS-WM embeds 2, 10, 352 bits of watermark within (513×513) image which is higher than all other existing watermarking schemes, 553, 460 bits higher than

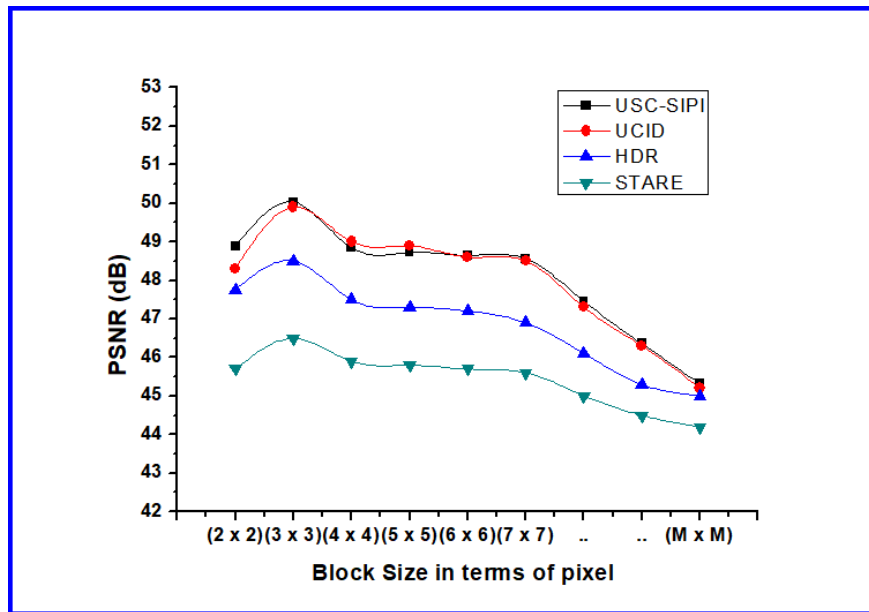


Figure 3.6: Graphical representation of PSNR (dB) versus block size in RWS-WM

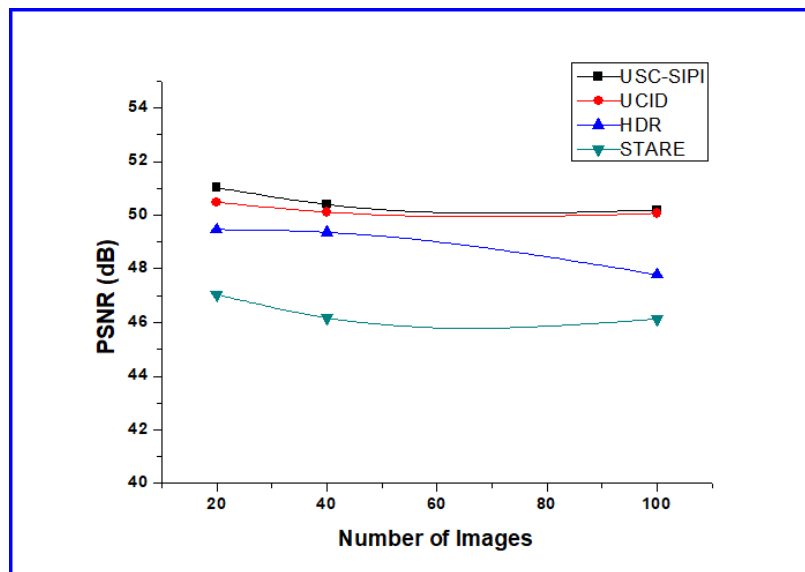


Figure 3.7: Graphical representation of PSNR (dB) versus number of images in RWS-WM

Soleymani & Taherinia [77] and 1, 329, 128 bits higher than Jana's [35]. In RWS-WM scheme, PSNR(dB) is greater than other existing schemes when USC-SIPI database is used. It is 13.63 dB higher than Soleymani & Taherinia [77] and 14.41 dB higher than Jana's [35] schemes. From this experiment, it is found that RWS-WM achieves high embedding capacity with good visual quality which is essential in the medical and military application.

Fig. 3.8 and Fig. 3.9 shows the comparison graph with respect to visual quality measured by PSNR (dB) and watermark embedding capacity measured by payload (bpp) of RWS-WM with other existing schemes. It has been observed that in both cases RWS-WM achieves better result

Table 3.4: Comparison with existing RWS in terms of average PSNR (dB) and Payload (bpp) in RWS-WM

Scheme	Lena (512 × 512)		Baboon (512 × 512)		Airplane (512 × 512)		Boats (512 × 512)	
	W (bits)	PSNR (dB)	W (bits)	PSNR (dB)	W (bits)	PSNR (dB)	W (bits)	PSNR (dB)
Ni et al. [60]	5460	48.20	5421	48.20	16171	48.30	7301	48.20
Hwang et al. [33]	5336	48.22	5208	48.20	15300	48.40	7501	48.25
Hu et al. [32]	60241	48.69	21411	48.34	77254	48.86	28259	48.40
Luo et al. [58]	71674	48.82	22696	48.36	84050	48.94	38734	48.50
Abadi et al. [1]	73207	48.78	45043	48.52	86964	48.92	43420	48.51
Yang et al. [100]	393660	45.26	536192	41.88	396369	43.96	440371	43.05
Jung et al. [40]	519180	48.16	519180	48.18	519180	48.18	519180	48.18
Jana B. [35]	776,224	35.80	776,224	36.12	776,224	35.78	776,224	35.42
Kuo et al. [45]	851,958	37.24	851,958	37.25	851,958	37.25	851,958	37.25
Soleymani & Taherinia [77]	1,551,892	36.56	1,551,892	36.90	1,551,892	35.05	1,551,892	35.90
RWS-WM	2,105,352	50.21	2,105,352	50.22	2,105,352	50.07	2,105,352	49.52

than other existing schemes.

3.1.3.2 Robustness Analysis

The robustness of RWS-WM has been evaluated through some standard analysis and attacks like Standard Deviation (SD), Correlation Coefficient(CC) and Brute force attack (BFA).

The statistical distortion is assessed by some parameters like SD (σ) and CC (ρ) using equation (2.9) and (2.10) respectively, to check the impact on the image after embed a good amount of watermark. The evaluation results on SD and CC of original and watermarked image are depicted in Fig. 3.10. Minimizing the parameter difference is one of the primary aims in order to get rid of statistical attacks. From Fig. 3.10 it is found that there is no substantial divergence in SD between the CI and WI.

The SD of the CI and WI is 128.37 and 128.38 respectively, and their difference is 0.01 for Lena image after embedding 2,105,352 bits watermark. The CC between the CI and WI is 0.99 for Lena image, which implies the change in the original image will predict a change in the same direction in the WI. So it is hard to locate the embedding position in the WI. This study shows that the magnitude of change in WI based on image parameters, is small from the original image. Since the image parameters have not been changed much, the method offers a good concealment of watermark and reduces the chance of watermark detection. Thus, it indicates a absolutely secure and robust watermarking scheme.

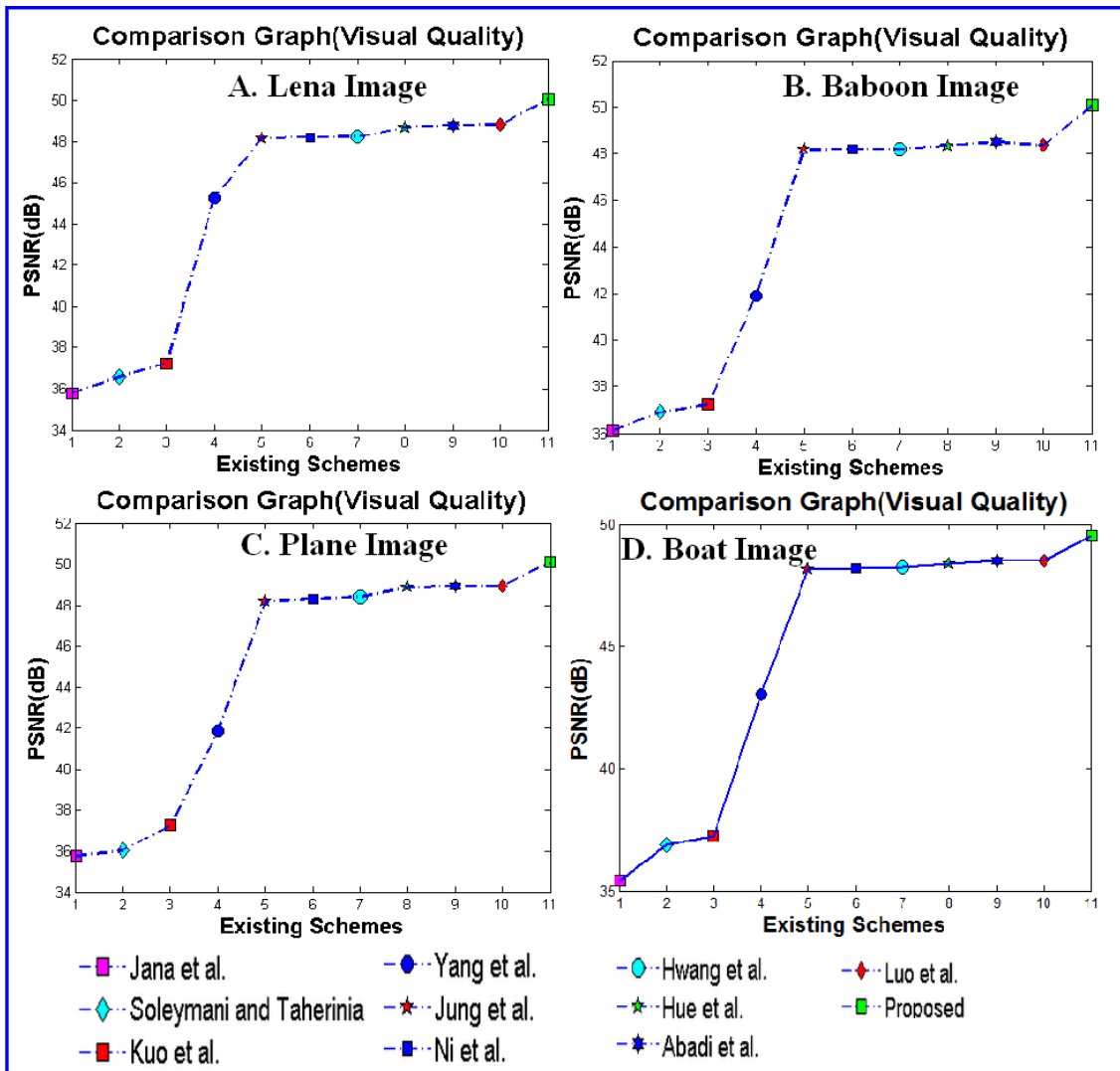


Figure 3.8: Comparison graph in terms PSNR (dB) with existing schemes in RWS-WM

3.1.3.3 Tamper Detection and Recovery

Some interesting experimental results using various benchmark image datasets [26] [90] [89] [61] during watermark extraction from tampered watermarked image is depicted in Fig. 3.11, 3.12, 3.13 and 3.14 depicted. Three kinds of tampered watermarked images are used for this experiment to measure the presence of watermark within tampered watermarked image. In first sample, 20% tampering is done through salt and pepper noise. After performing watermark extraction, it is observed that the presence of watermark within the tampered watermarked image, can successfully detected. The second case, 10% tampering is done through opaque. It is also shown that the hidden watermark has been recovered using extraction algorithm and tampered cover image is extracted. The difference of SD between CI and WI is 2.58 which is negligible and CC between CI and WI is 0.97 which also represent negligible changes. But the presence of

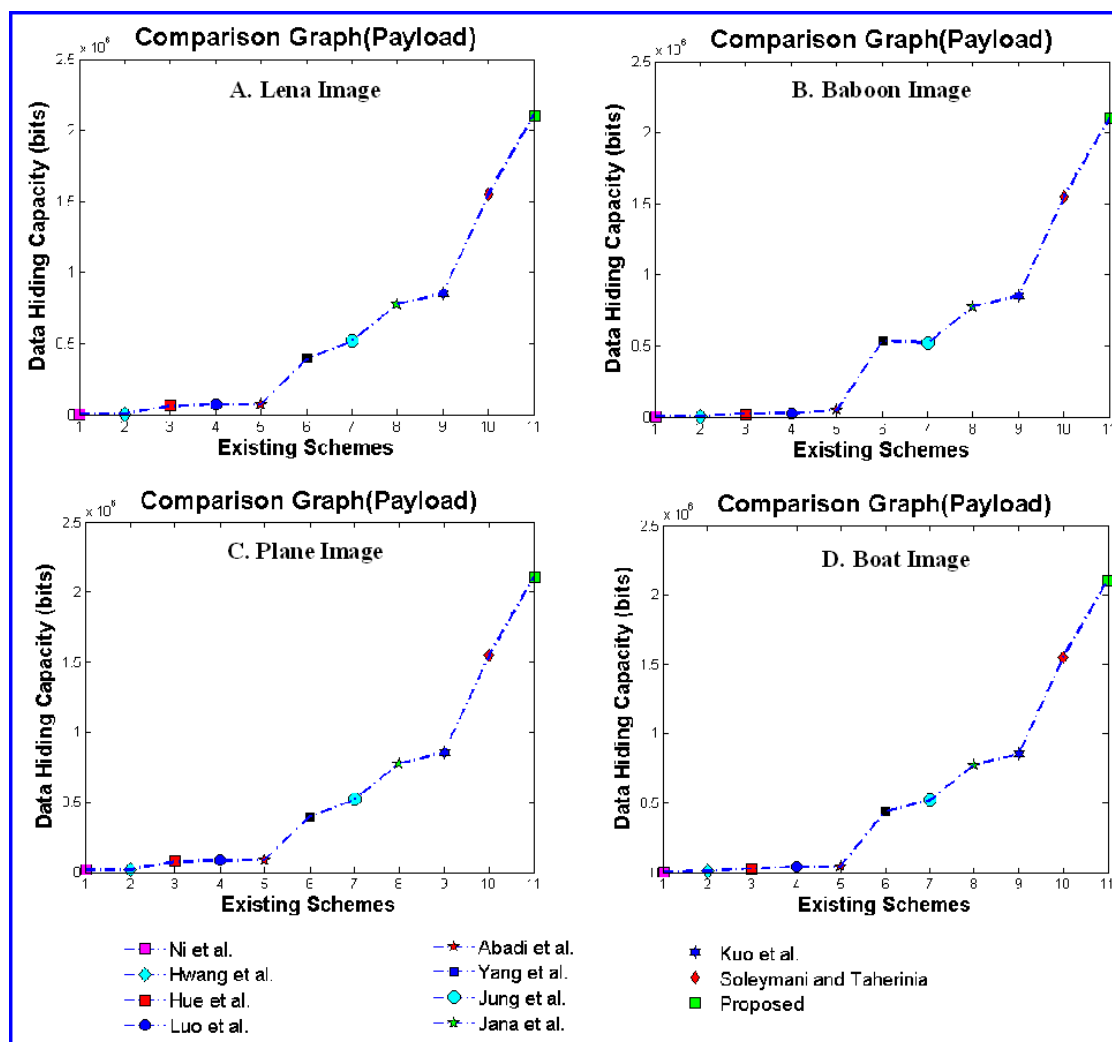


Figure 3.9: Comparison graph in terms of payload (bpp) with existing schemes in RWS-WM watermark have been found in all attacking condition from tampered watermarked image used on USC-SIPI [90] image dataset. The last one is used tampered image through 20% cropping. Here the hidden watermark and cover image are successfully identified with some distortion. The various tamper detection and authentication experiments from tampered image of different image datasets [26] [90] [89] [61] are presented in Fig. 3.11, 3.12, 3.13 and 3.14. From this experiment, it is observed that the adversary can not remove the watermark from watermarked image through various tampering approaches such as modification, filling, cropping, opaque etc. But the authorized person can easily identify the hidden watermark within the tampered watermarked image. So this scheme will be used to solve image authentication, ownership identification, copyright protection and tamper detection problems. In this approach, only one-bit increment or decrement is happened during data embedding, but original data were not embedded. This enhances the security of RWS-WM and the scheme is robust against vari-

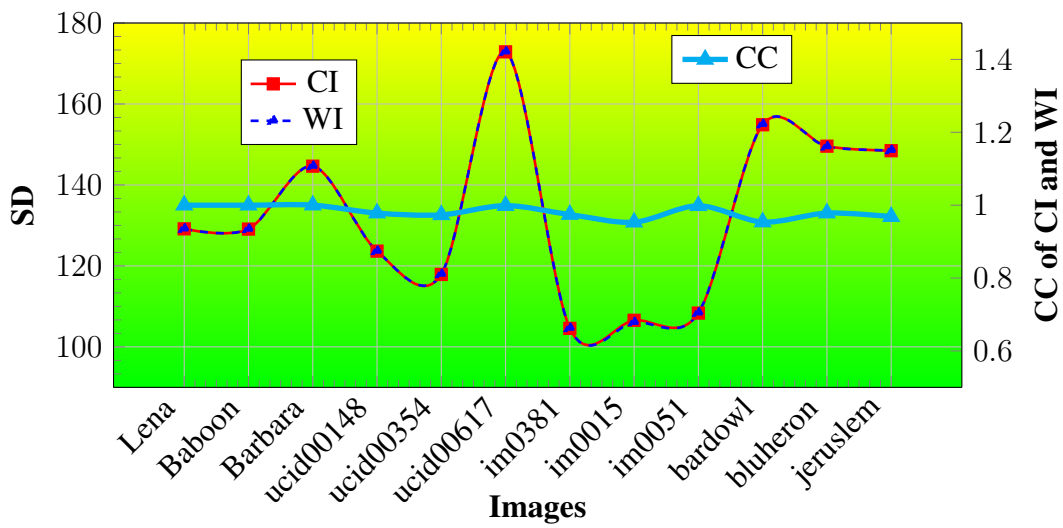


Figure 3.10: Graphical representation of various images with respect to SD and CC in RWS-WM

Original Image (512×512) (C)	Watermark (361 × 243)	Watermarked Image (C')	Tampered Watermarked Image	Recovered Watermark	Recovered Cover Image	Statistical Analysis
						Difference of SD between C & C' = 154.93-128.37 = 26.56 CC between C & C' = 0.75
Lena	Logo Image	PSNR=50.07 (dB)	Salt & pepper(20%)	PSNR=9.58 (dB)	PSNR=15.61 (dB)	
						Difference of SD between C & C' = 128.37-125.79 = 2.58 CC between C & C' = 0.97
Lena	Logo Image	PSNR=50.07 (dB)	Opaque(10%)	PSNR=20.24 (dB)	PSNR=28.72 (dB)	
						Difference of SD between C & C' = 184.43-128.37 = 57.06 CC between C & C' = 0.66
Lena	Logo Image	PSNR=50.07 (dB)	Crop(20%)	PSNR=14.87 (dB)	PSNR=13.51 (dB)	

Figure 3.11: Tampered results on USC-SIPI image database [90] in RWS-WM

ous attacks. Also, the scheme is blind and fragile watermarking scheme which can successfully detect the tampered and authenticate the ownership of images.

The RWS-WM protects multimedia documents by embedding watermark using WM. Here, the position values of the watermark are stored within index file, instead of the original watermark information. The μ has been used to update WM during each operation on every R, G, B block. The scheme is secured to prevent possible malicious attacks. The Fig. 3.15 shows the revelation



















Original Image (512×512) (C)	Watermark (361 × 243)	Watermarked Image (C')	Tampered Watermarked Image	Recovered Watermark	Recovered Cover Image	Statistical Analysis
						Difference of SD between C & C' = 172.87-185.52 = 12.65 CC between C & C' = 0.98
ucid00617	Logo Image	PSNR=49.56(dB)	Salt & pepper(20%)	PSNR=12.64(dB)	PSNR=15.36(dB)	
						Difference of SD between C & C' = 172.87-188.4 = 15.53 CC between C & C' = 0.91
ucid00617	Logo Image	PSNR=49.56(dB)	Opaque(10%)	PSNR=25.28(dB)	PSNR=28.72(dB)	
						Difference of SD between C & C' = 172.87-195.17 = 22.03 CC between C & C' = 0.84
ucid00617	Logo Image	PSNR=49.56(dB)	Crop(20%)	PSNR=18.95(dB)	PSNR=13.51(dB)	

Figure 3.12: Tampered results on UCID image database [61] in RWS-WM










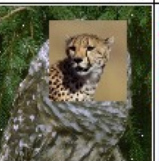

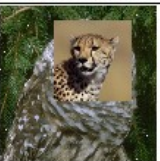






Original Image (512×512) (C)	Watermark (361 × 243)	Watermarked Image (C')	Tampered Watermarked Image	Recovered Watermark	Recovered Cover Image	Statistical Analysis
						Difference of SD between C & C' = 154.86-155.32 = 0.46 CC between C & C' = 0.99
Bardowl	Logo Image	PSNR=47.78(dB)	Salt & pepper(20%)	PSNR=12.64(dB)	PSNR=15.36(dB)	
						Difference of SD between C & C' = 154.86-164.47 = 9.61 CC between C & C' = 0.94
Bardowl	Logo Image	PSNR=47.78(dB)	Opaque(10%)	PSNR=25.28(dB)	PSNR=28.72(dB)	
						Difference of SD between C & C' = 154.86-168.79 = 13.93 CC between C & C' = 0.87
Bardowl	Logo Image	PSNR=47.78(dB)	Crop(20%)	PSNR=14.95(dB)	PSNR=13.51(dB)	

Figure 3.13: Tampered results on HDR image database [26] in RWS-WM

example where wrong weighted matrix and wrong index file are used to reveal the watermark image. If the malicious attacker holds the original image and watermarked image and is fully aware of the proposed scheme, the watermark still cannot be correctly revealed without knowing the correct index file and correct WM. For example, Fig. 3.15 shows original image, watermark,




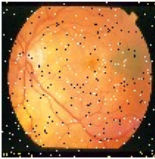

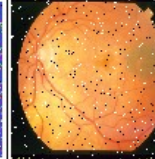











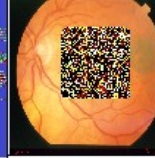
Original Image (512×512) (C)	Watermark (361 × 243)	Watermarked Image (C')	Tampered Watermarked Image	Recovered Watermark	Recovered Cover Image	Statistical Analysis
						Difference of SD between C & C' =104.57-145.63 =41.06 CC between C & C'=0.83
im0381	Logo Image	PSNR=46.18(dB)	Salt & pepper(20%)	PSNR=12.63(dB)	PSNR=15.36(dB)	
						Difference of SD between C & C' =104.57-126.63 =22.06 CC between C & C'=0.87
im0381	Logo Image	PSNR=46.18(dB)	Opaque(10%)	PSNR=19.89(dB)	PSNR=28.72(dB)	
						Difference of SD between C & C' =104.57-146.63 =42.06 CC between C & C'=0.79
im0381	Logo Image	PSNR=46.18(dB)	Crop(20%)	PSNR=13.88(dB)	PSNR=13.51(dB)	

Figure 3.14: Tampered results on STARE image database [89] in RWS-WM

attack with unknown weighted matrix and index file. The result indicates that attacker only acquires noise like images when applying wrong weighted matrix and/or index file to reveal the watermark image. But all the cases extracted cover image is as similar to the original image with good visual quality. In this context, it is mentioned that the RWS-WM is highly robust and only authorized person can extract the watermark from WI. It protects the copyright for the image owner. Furthermore, attacker may employ BFA, that tries all possible permutation to reveal watermark, which is computationally infeasible for current computers. So the RWS-WM achieves stronger robustness against several attacks. Again, the watermark can be extracted without encountering any loss of data and original image can be retrieved successfully from the WI using valid WM and Index file.

3.2 RWS-CA: Cellular Automata based RWS²

One dimensional Cellular Automata (CA) or Elementary Cellular Automata (ECA) have many applications in coding theory and in cryptography. Here, CA is applied in image watermarking scheme due to its simplicity and low hardware complexity. A special type of periodic boundary

²Submitted in **IET Image Processing**, IEEE: ISSN 1751-9659, **Impact Factor: 1.401** with title *Robust Watermarking Scheme for Tamper Detection and Authentication Exploiting Cellular Automata*













(A) Brute force attack with unknown weighted matrix													
Original Image	Watermark	Brute force attack	Extracted Watermark	Extracted cover									
		<table border="1"> <tr><td>2</td><td>1</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td></tr> </table>	2	1	3	4	5	6	7	8	9		
2	1	3											
4	5	6											
7	8	9											
Lena (512×512)	Logo (300×243)	weighted matrix	PSNR= 7.11(dB)	PSNR= 43.58(dB)									
(B) Brute force attack with unknown index file													
Original Image	Watermark	Brute force attack	Extracted Watermark	Extracted cover									
		<pre>(+01.),(-01.),(-21.), (-02.),(-21.),(+21.), (-11.),(-01.),(+01.), (-20.),(-21.),(-11.), (-00.),(+00.),(+01.), (-20.),(-00.),(-12.)</pre>											
Lena (512×512)	Logo (300×243)	index file	PSNR= 7.22(dB)	PSNR= 43.71(dB)									
(C) Brute force attack with unknown both index file and weighted matrix													
Original Image	Watermark	Brute force attack	Extracted Watermark	Extracted cover									
		<table border="1"> <tr><td>2</td><td>1</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td></tr> </table> <pre>(+01.),(-01.),(-21.), (-02.),(-21.),(+21.), (-11.),(-01.),(+01.),</pre>	2	1	3	4	5	6	7	8	9		
2	1	3											
4	5	6											
7	8	9											
Lena (512×512)	Logo (300×243)	Both unknown	PSNR= 7.12(dB)	PSNR= 43.71(dB)									

Figure 3.15: Experimental results under brute force attack in RWS-WM

CA (CA attractor) is a very powerful tool which helps to achieve image authentication and tamper detection. CA attractor (CAA) is combined with a shared secret key (μ) introduced in this work to improve security level. Authentication Code (AC) is generated from the watermark image by employing Secure Hash Algorithm (SHA-512). This approach embeds watermark and AC within each sample images. The experimental results of RWS-CA are compared with the result of existing schemes to establish the strength and effectiveness of the scheme. Some standard NIST recommended steganalysis and a series of attacks are conducted to measure robustness and imperceptibility. It has been observed that RWS-CA is robust, secure and it can detect tamper made by unintentional or intentional attacks.

In this section, a color image based watermarking method, using elementary cellular automata has been introduced for authentication and tamper detection. The watermark bits are not

embedded within the cover media directly to keep the unvarying security level. It is changed using Elementary Cellular Automata Attractor (ECAA) before embedding into the cover media. This works as a secret key in proposed algorithm. In embedding phase, the tamper detection bits are generated by employing CAA in coloured cover image and the authentication bits (AC) are computed by employing cryptographic hash function SHA-512 on watermark. Then this AC information are embedded within the sub-sample of color images. In the detection phase, authentication bits are extracted and authenticity is verified. In this investigation, our aim is to formulate a reversible watermarking algorithm using cellular automata for practical applications. The proposed scheme provides good visual characteristics to the watermarked image. The RWS-CA has been described in two subsection (3.2.1 and 3.2.2).

3.2.1 Watermark Embedding Phase

The schematic framework of embedding, extraction and authentication phase of RWS-CA has been depicted in Fig. 3.16(a), 3.16(b) and 3.16(c) respectively.

A shared secret key μ of length 128-bit is considered for watermark embedding. Now watermark bit stream (M) is taken from the watermark image (W) and 512-bit AC is generated using SHA-512 algorithm. After that, a sequence of numbers between 0 to 8 have been generated using pseudo random number generator (PRNG) scheme with the seed value μ and then stored in the array $NS[]$. Choose four secret vectors $\{\xi_1, \xi_2, \xi_3, \xi_4\}$ from $NS[]$ sequentially, such that the sum of all four secret vector is τ (i.e. $\tau = 8$). Hence, any τ -block CA attractor (δ_i for $i = 1$ to τ) has to be considered for watermark embedding. Here, CA rule-42 is applied to choose attractor block with initial state 53. For example, $(3 - 2 - 2 - 1)$ has been collected from $NS[]$ as four secret vector (ξ_i for $i = 1, 2, 3, 4$) and $(53 - 154 - 77 - 166 - 83 - 169 - 212 - 106)$ is considered as 8-block CA attractor (δ_i for $i = 1$ to 8). For the next block, μ is updated using equation (3.4) to get a new sequence of four secret vector (ξ_i for $i = 1, 2, 3, 4$) from updated $NS[]$.

$$\mu_{i+1} = (\mu_i \times \chi) \text{ mod } 256 \quad (3.4)$$

where, χ is the average of 16 pixels of the corresponding block of cover image.

In embedding phase, a color image CI is considered as cover image. Then, CI is separated into three RGB channels CI_R , CI_G , and CI_B . After that, first color block CI_R is considered

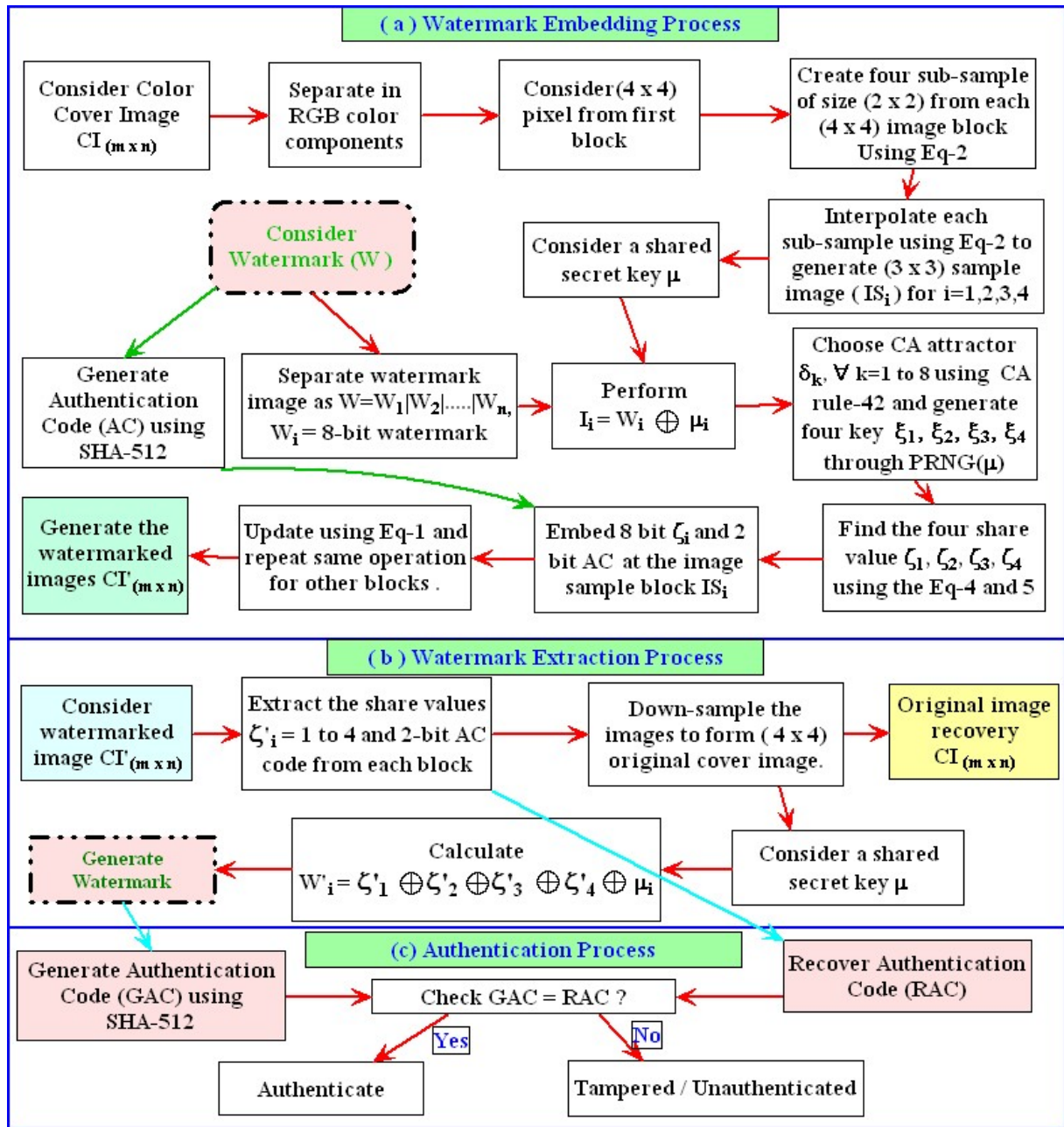


Figure 3.16: Block diagram of watermarking process in RWS-CA

and divided into (4×4) image blocks. Then, four (2×2) sub-samples (SSI) are produced from this (4×4) image blocks using equation (3.5). A sample generation and interpolation method are depicted in Fig. 3.17 (a). Four (3×3) interpolated image blocks ICI are constructed from each sub-sample using equation (3.6).

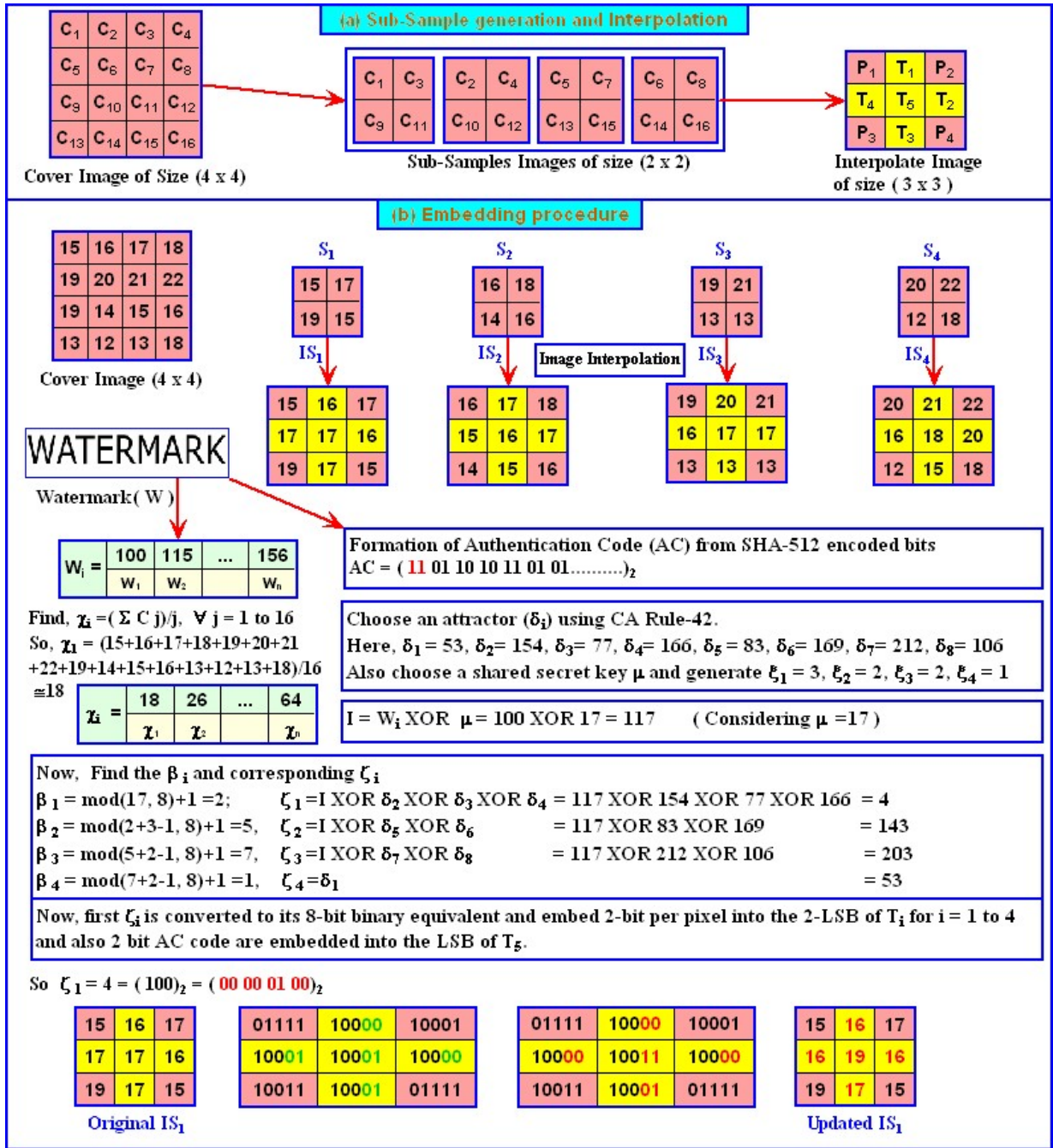


Figure 3.17: Numerical illustration of watermark embedding phase of RWS-CA

$$\left. \begin{aligned}
 SC_1(i, j) &= CI(2i - 1, 2j - 1) \\
 SC_2(i, j) &= CI(2i - 1, 2j) \\
 SC_3(i, j) &= CI(2i, 2j - 1) \\
 SC_4(i, j) &= CI(2i, 2j)
 \end{aligned} \right\} \forall i = 1 \text{ to } m; \text{ and } j = 1 \text{ to } n; \quad (3.5)$$

$$\left\{ \begin{array}{l} I_{min} = \min CI(i, j), CI(i + 2, j), CI(i, j + 2), CI(i + 2, j + 2) \\ I_{max} = \max CI(i, j), CI(i + 2, j), CI(i, j + 2), CI(i + 2, j + 2) \\ AD = \frac{3 \times I_{min} + 2 \times I_{max}}{5} \\ CI(i, j) = I(i, j) \\ CI(i, j + 1) = \frac{AD + (CI(i, j) + CI(i, j + 2))}{2} \\ CI(i + 1, j) = \frac{AD + (CI(i, j) + CI(i + 2, j))}{2} \\ CI(i + 1, j + 1) = \frac{(CI(i, j) + CI(i + 1, j) + CI(i, j + 1))}{3} \end{array} \right. \quad (3.6)$$

where $i = 2m, j = 2n, m, n = 0, 1, 2, \dots, k$. and m and n are considered as row and column of the CI. The interpolated image ICI is generated from the each sub-sampled image SSI which corresponds to CI individually presented in the equation (3.6). Here, a new interpolation scheme is proposed with a AD variable that provide better quality interpolated image.

Now, the watermark bits are embedded into each ICI. First 8-bit data (W_i) is taken from the watermark bits (M). Then an XOR (\oplus) operation is performed between μ and W_i to get the value I_i . Now, W_i are transformed into four parts ζ_i (for $i = 1$ to 4) to embed in four different interpolated image samples. Also the randomness of the steps β_i can be achieve by choosing the number of δ_i embedded in the interpolated sample images. The β_i and ζ_i are calculated using the equation (3.7) and (3.8) respectively.

$$\left\{ \begin{array}{l} \beta_1 = \text{mod}(\mu, \tau) + 1 \\ \beta_2 = \text{mod}(\beta_1 + \xi_1 - 1, \tau) + 1 \\ \beta_3 = \text{mod}(\beta_2 + \xi_2 - 1, \tau) + 1 \\ \dots \\ \beta_k = \text{mod}(\beta_{k-1} + \xi_{k-1} - 1, \tau) + 1 \end{array} \right. \quad (3.7)$$

$$\left\{ \begin{array}{l} \zeta_1 = I \oplus S_{\text{mod}(\mu+0, \tau)} \oplus S_{\text{mod}(\mu+1, \tau)} \dots \oplus S_{\text{mod}(\mu+\xi_1-1, \tau)} \\ \zeta_2 = I \oplus S_{\text{mod}(\mu+\xi_1, \tau)} \oplus S_{\text{mod}(\mu+\xi_1+1, \tau)} \dots \oplus S_{\text{mod}(\mu+\xi_1+\xi_2-1, \tau)} \\ \zeta_3 = I \oplus S_{\text{mod}(\mu+\xi_1+\xi_2, \tau)} \oplus S_{\text{mod}(\mu+\xi_1+\xi_2+1, \tau)} \dots \oplus S_{\text{mod}(\mu+\xi_1+\xi_2+\xi_3-1, \tau)} \\ \dots \\ \zeta_k = I \oplus S_{\text{mod}(\mu+\xi_1+\xi_2+\dots+\xi_{k-1}, \tau)} \oplus S_{\text{mod}(\mu+\xi_1+\xi_2+\dots+\xi_{(k-1)}+1, \tau)} \dots \oplus S_{\text{mod}(\mu+\xi_1+\xi_2+\dots+\xi_k-1, \tau)} \end{array} \right. \quad (3.8)$$

Now ζ_i (for $i = 1$ to 4) are converted to 8-bit binary number. Each 2 bits from this number are embedded into 2 LSB of four different pixels (T_i) of the ICI. In this way all the bits of ζ_i are embedded into the first block of the corresponding shares ICI. The 2-bit AC is embedded into the middle pixel (T_5) of the first block of each shares. Before going to the next block, the secret key μ is updated using the equation (3.4). This way, other watermark bits and authentication bits are encoded within the rest of pixel blocks of watermark shares (WS).

3.2.2 Watermark Extraction and Recovery Phase

The watermark extraction phase has been clearly described using a numerical example illustrated in Fig. 3.18 and an algorithmic presentation is shown in Algorithm 3.4. The (4×4) pixel

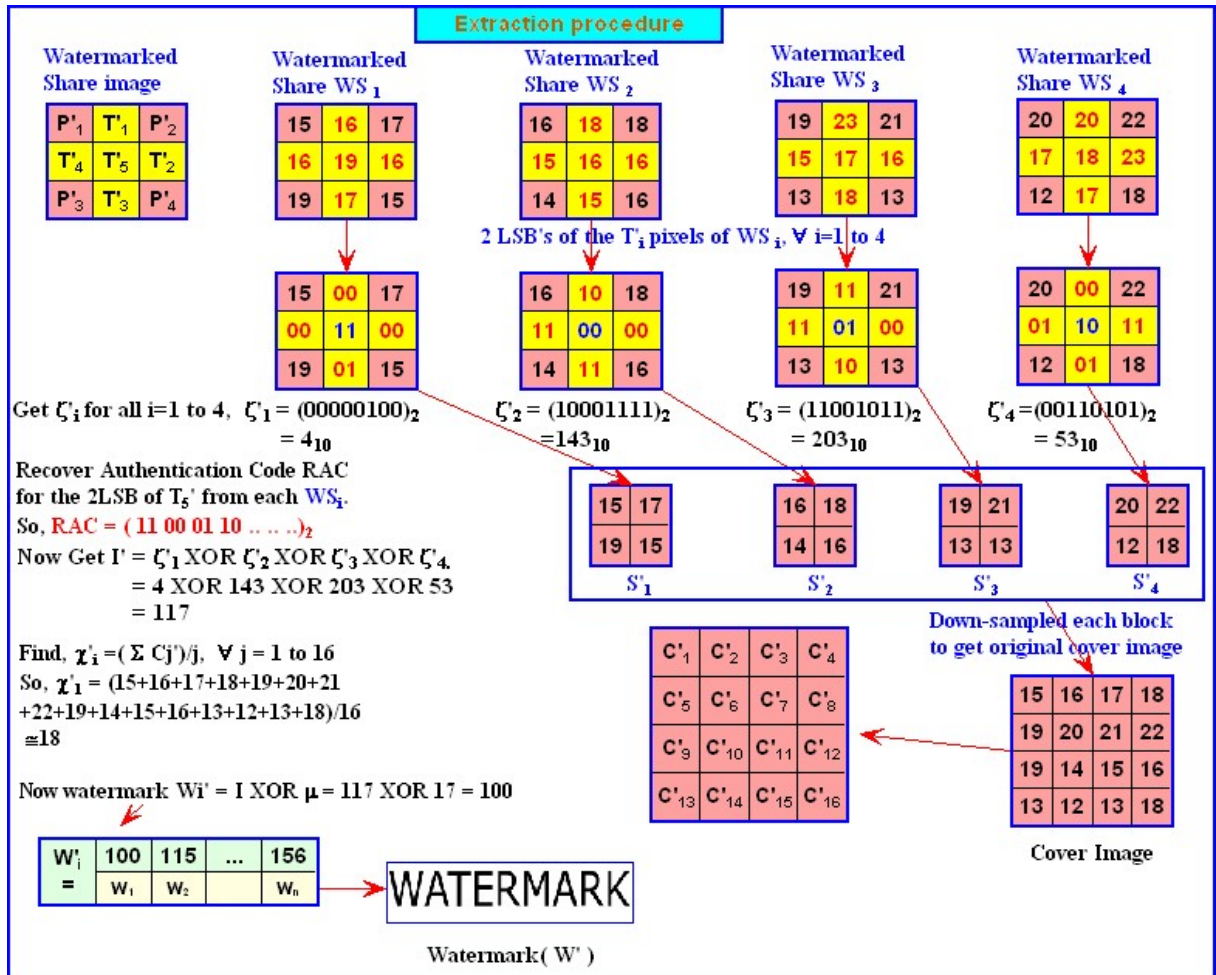


Figure 3.18: Numerical illustration of watermark extraction phase of RWS-CA

block of original cover image is constructed from the unaffected pixels P_1, P_2, P_3 and P_4 from each (3×3) pixel blocks of all the shares (WS). In this way the whole cover image is constructed

Algorithm 3.3: RWS-CA: Watermark Embedding Algorithm**Input** : Original Cover Image ($CI_{m \times n}$), Watermark (W), Secret Key (μ), CA attractor (δ_i)**Output**: Sub-sampled Watermarked Images WS'_1, WS'_2, WS'_3 and WS'_4

```

1  Algorithm Embedding () :
    // Convert cover image to an 3D array, for RGB pixels
2  coverImage=readImage(CoverFile);
3  subSampleArray[]=createSubSamplesArray();
4  interpolatedArray[]=createInterpolatedArray(subSampleArray[]);
5  secretDataBits=readImage(WatermarkImageFile);
6  attractor[]={53,154,77,166,83,169,212,106};
7   $\xi$ []={3,2,2,1};  coverY=0;
8  for (int y=0;y<imageHeight;y=y+3) do
9      coverX=0;
10     for (int x=0;x < imageWidth;x=3) do
        // For Red=0, Green=1 and Blue=2
11     for (int color=0; color<=2; color++) do
12         watermarkPixel=binaryToDecimal(get8BitsSecretData(secretDataBits));
13         avg = getAvgFromCover(coverX, coverY, color);
14         XorValue = (watermarkPixel  $\oplus$  avg);
        // Process 1st interpolated image
15         p = Mod(avg, 8) + 1;  sh = XorValue;
16         for ( i=p; i<=p +  $\xi$ [1] -1; i++) do
17             |  sh = sh  $\oplus$  attractor[i];
18         end
19         EmbedBitsInImageBlock(interpolatedArray[0], color, x, y, sh);
        // Process 2nd interpolated image
20         p = Mod(p +  $\xi$ [1] -1, 8) + 1;  sh = XorValue;
21         for ( i=p; i<=p +  $\xi$ [2] -1; i++) do
22             |  sh = sh  $\oplus$  attractor[i];
23         end
24         EmbedBitsInImageBlock(interpolatedArray[1], color, x, y, sh);
        // Process 3rd interpolated image
25         p = Mod(p +  $\xi$ [2] -1, 8) + 1;  sh = XorValue;
26         for ( i=p; i<=p +  $\xi$ [3] -1; i++) do
27             |  sh = sh  $\oplus$  attractor[i];
28         end
29         EmbedBitsInImageBlock(interpolatedArray[2], color, x, y, sh);
        // Process 4th interpolated image
30         p = Mod(p +  $\xi$ [3] -1, 8) + 1;  sh = 0;
31         for ( i=p; i<=p +  $\xi$ [4] -1; i++) do
32             |  sh = sh  $\oplus$  attractor[i];
33         end
34         EmbedBitsInImageBlock(interpolatedArray[3], color, x, y, sh);
35     end
36     coverX= coverX+4;
37 end
38 coverY=coverY+4;
39 end

40 Function EmbedBitsInImageBlock (pixelArray, color, x, y, data) :
    // Convert the data to 8 bit binary number
41 stringData=convertTo8BitBinary(data);
    // Split this 8 bit data into four 2 bit data and convert them to decimal
42 first2Bits = binaryToDecimal(stringData.substring(0,2));
43 second2Bits = binaryToDecimal (stringData.substring(2,4));
44 third2Bits = binaryToDecimal (stringData.substring(4,6));
45 fourth2Bits = binaryToDecimal (stringData.substring(6,8));
    // Embed each 2 bit decimal data in LSB 2 of 4 pixels
46 pixelArray[x+1][y+0][color] = (pixelArray[x+1][y+0][color] & 252) + first2Bits;
47 pixelArray[x+0][y+1][color] = (pixelArray[x+0][y+1][color] & 252) + second2Bits;
48 pixelArray[x+1][y+2][color] = (pixelArray[x+1][y+2][color] & 252) + third2Bits;
49 pixelArray[x+2][y+1][color] = (pixelArray[x+2][y+1][color] & 252) + fourth2Bits;

```

from 4 shares. Now, 2 LSBs are collected from the pixels T_1, T_2, T_3 and T_4 of the first block of WS, to form 8-bit binary number. This number is converted to its decimal equivalent and stored into corresponding ζ'_i . Also, the LSB of T_5 is collected from each share and appended to form a string (RAC). Now, a shared secret key μ is considered and an XOR operation is performed with ζ'_i to get I'_i . This I'_i is converted to 8-bit binary string and appended with W' . Then the average (χ'_i) of 16 pixels is calculated from the first (4×4) block of the reconstructed cover image. After that, this χ'_i is used to modify μ using equation (3.4). Then the above process is continued for the rest of the (3×3) pixel blocks of all the shares and recovered I'_i from each iteration is appended with W' . So, W' contains all the extracted watermarked bits in binary format. The watermarked image is then reconstructed from W' . The authentication code (GAC) is generated from this watermark using SHA-512 algorithm. At last, GAC is compared with RAC to check the authenticity of recovered cover image.

3.2.3 Experimental Results and Comparison

A set of benchmark [89], [61], [90], [26] colour images are considered to assess the effectiveness of RWS-CA. Here, three different size of logo images are considered as a watermark (W) as shown in Fig. 3.20 to measure the quality and corresponding capacity. Performances of the related schemes are compared to test its effectiveness. MSE [35], PSNR [35], SSIM [78] and Q-Index are computed to test the perceptible characteristics after embedding. Also NCC [94], BER [65], SD (σ) [35] and CC (ρ) [35] are computed for tamper detection in a watermark image. Performance of RWS-CA has been assessed according to computation time and it has been compared with other existing schemes.

3.2.3.1 Quality Measurement and Payload Analysis

The fundamental necessities of any watermarking scheme are robustness and imperceptibility. The subjective characteristics of the watermarked images is evaluated in RWS-CA and it has been depicted in Fig. 3.19. Evaluation results of RWS-CA in terms PSNR, BPP and Q-Index have been calculated using equation (2.6), (2.7) and (2.12) respectively. After embedding different number of watermark bits, the corresponding results are presented in Table 3.5. It is observed from Fig. 3.19 that no visual distortions are detected after embedding maximum payload of 3, 93, 216 bits watermark. The Q-Index values are close to unity which establishes the














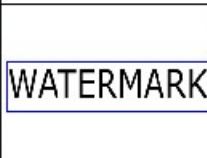



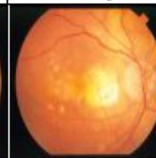
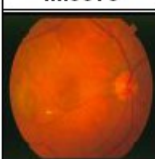
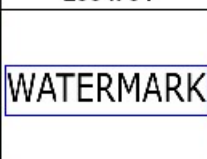
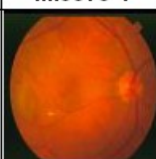
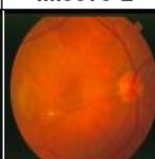
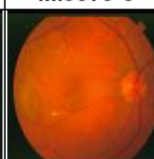
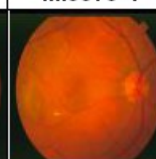

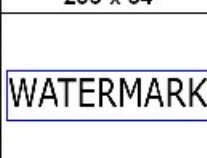

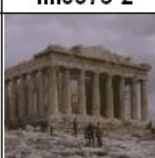


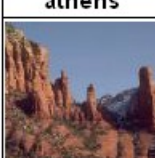
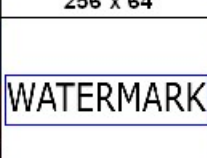
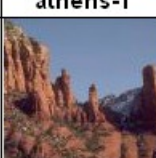
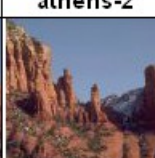
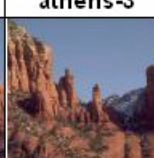
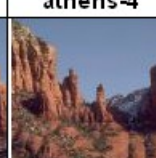
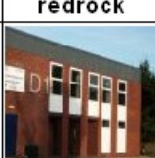
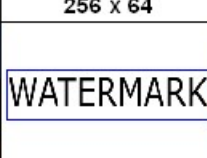
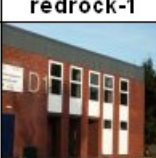
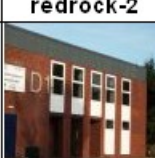
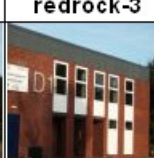

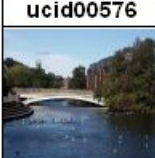
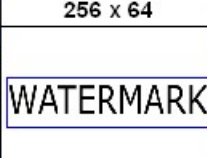

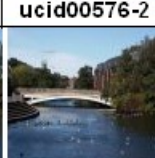


IMAGE DATABASE	COVER IMAGE	WATERMARK	OUTPUT WATERMARKED IMAGES			
USC-SIPI						
	Lena	256 x 64	Lena-1	Lena-2	Lena-3	Lena-4
						
	Tiffany	256 x 64	Tiffany-1	Tiffany-2	Tiffany-3	Tiffany-4
STARE						
	im0370	256 x 64	im0370-1	im0370-2	im0370-3	im0370-4
						
	im0376	256 x 64	im0376-1	im0376-2	im0376-3	im0376-4
HRD						
	athens	256 x 64	athens-1	athens-2	athens-3	athens-4
						
	redrock	256 x 64	redrock-1	redrock-2	redrock-3	redrock-4
UCID						
	ucid00576	256 x 64	ucid00576-1	ucid00576-2	ucid00576-3	ucid00576-4
						
	ucid00617	256 x 64	ucid00348-1	ucid00348-2	ucid00348-3	ucid00348-4

Figure 3.19: Pictorial results of output images in RWS-CA

Algorithm 3.4: RWS-CA: Watermark Extraction Algorithm

```

input : Stego Images  $CI'_i$ , and Secret Keys ( $\mu$ )
output: Cover Image ( $CI_{m \times n}$ ) and Watermark ( $W'$ )

1 Algorithm Extraction():
2   stegoArray[]=readImage(StegoFiles); coverY=0;
3   for ( y=0; y<imageHeight; y=y+3) do
4     coverX=0;
5     for ( x=0; x<imageWidth; x=x+3) do
6       // Color=0 for Red, Color=1 for Green and Color=2 for Blue
7       for (int color=0; color≤ 2; color++) do
8         sh1 = ExtractBitsFromImageBlock(stegoArray[0], color, x, y);
9         sh2 = ExtractBitsFromImageBlock(stegoArray[1], color, x, y);
10        sh3 = ExtractBitsFromImageBlock(stegoArray[2], color, x, y);
11        sh4 = ExtractBitsFromImageBlock(stegoArray[3], color, x, y);
12        avg = getAvgFromCover(coverX, coverY, color);
13        xorValue = sh1 ⊕ sh2 ⊕ sh3 ⊕ sh4 ⊕ coverAgerage;
14      end
15      coverX=coverX+4;
16    end
17    coverY=coverY+4;
18  end

18 Function ExtractBitsFromImageBlock ( stegoArray, color, x, y):
19   String binaryString;
20   // Get last 2 bit from the 1st pixel and add it to the binary string
21   last2Bits = stegoArray[x+1][y+0][color] & 3;
22   binaryString.append(convertTo2BitBinaryString(last2Bits));
23   // Get last 2 bit from the 2nd pixel and add it to the binary string
24   last2Bits = stegoArray[x+0][y+1][color] & 3;
25   binaryString.append(convertTo2BitBinaryString (last2Bits, 2));
26   // Get last 2 bit from the 3rd pixel and add it to the binary string
27   last2Bits = stegoArray[x+1][y+2][color] & 3;
28   binaryString.append(convertTo2BitBinaryString (last2Bits, 2));
29   // Get last 2 bit from the 4th pixel and add it to the binary string
30   last2Bits = stegoArray[x+2][y+1][color] & 3;
31   binaryString.append(convertTo2BitBinaryString (last2Bits, 2));
32   // Convert the binary string to decimal value
33   extractedData = BinaryToDecimal(binaryString);
34   // Return the decimal data
35   return extractedData;

```

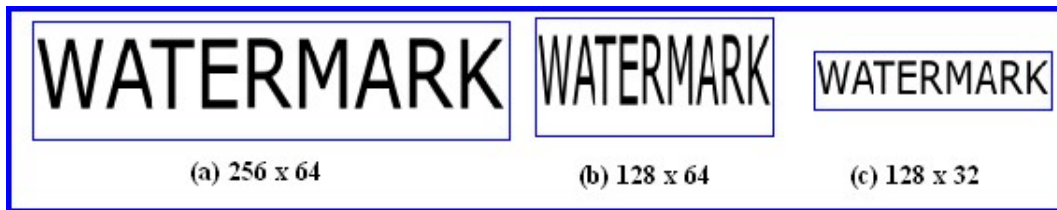


Figure 3.20: Watermark image (logo) with different size used in RWS-CA

acceptability of RWS-CA. From the Table 3.5, it is also seen that the PSNR quality will be increase with decreasing the embedding capacity.

The RWS-CA has been tested taking more than 100 sample images from four different standard benchmark image databases and experimental outcomes are exhibited in Table 3.6. It has been noticed that, approximately 50 dB average PSNR can be achieved after embedding a high-

Table 3.5: Capacity, PSNR, Q-Index and Payload values for standard benchmark images in RWS-CA

Dataset	Image	Capacity (bits)	PSNR	Q-Index	Payload (bpp)
USC-SIPI [90]	Lena	98,304	53.03	0.99999	0.375
		1,96,608	51.35	0.99998	0.75
		3,93,216	50.21	0.99997	1.5
UCID [61]	Jerusalem	98,304	53.17	0.99999	0.375
		1,96,608	52.67	0.99997	0.75
		3,93,216	50.91	0.99996	1.5
STARE [89]	Im0001	98,304	53.15	0.99998	0.375
		1,96,608	50.93	0.99996	0.75
		3,93,216	49.62	0.99995	1.5
HDR [26]	Medical1	98,304	53.59	0.99998	0.375
		1,96,608	51.35	0.99997	0.75
		3,93,216	49.27	0.99995	1.5

est amount of 3, 93, 216 bits watermark.

The Fig. 3.21 depicts the graphical representation of the experimental results on PSNR con-

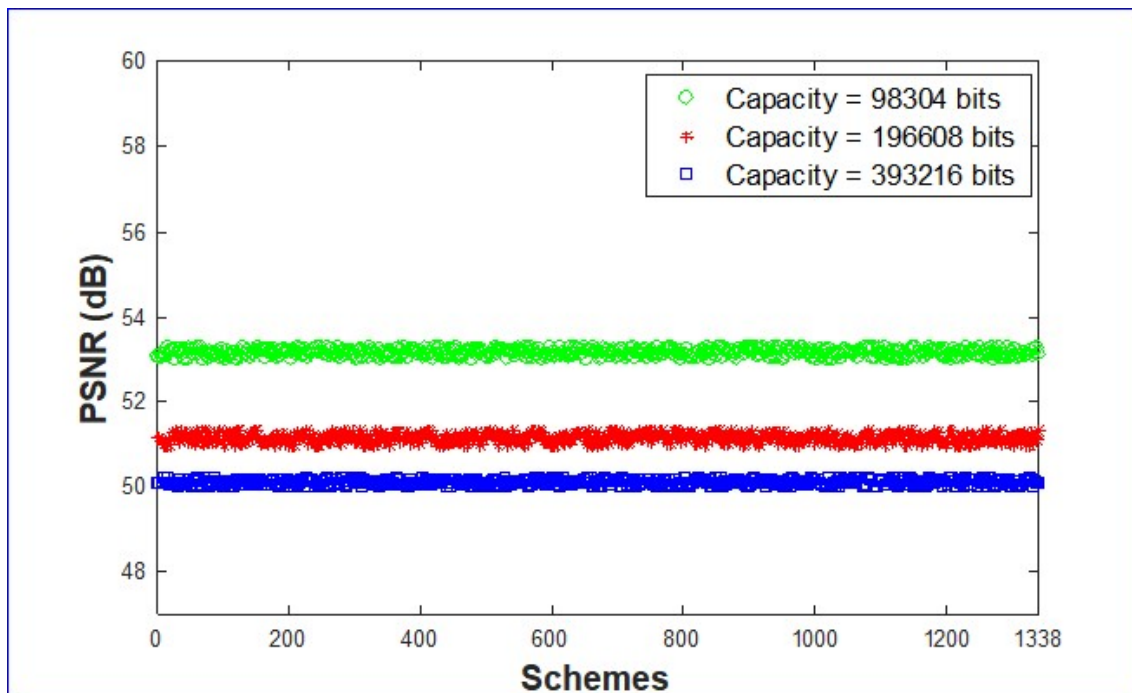


Figure 3.21: Graphical representation of PSNR (dB) on UCID image database [61] in RWS-CA considering 1338 images from UCID image database [61] at different level of embedding capacity.

Table 3.6: Average PSNR of various yardstick image datasets considering 25 to 100 images in RWS-CA

Dataset	Image Size	Total Image	PSNR (Average)
UCID [61]	512 × 512	25	49.93
		50	50.34
		100	49.91
USC-SIPI [90]	512 × 512	25	50.09
		50	49.65
		100	50.21
STARE [89]	512 × 512	25	50.73
		50	49.83
		100	49.62
HDR [26]	512 × 512	25	50.79
		50	49.94
		100	49.27

Again MSE, NCC, SSIM and BER value has been calculated using the equation (2.5), (2.11), (2.8), and (2.13) respectively. Table 3.7 exhibits the test results concerning MSE, PSNR, NCC, SSIM, Q-Index and BER with color cover images of four different yardstick datasets. From Table 3.7, it is found that the average PSNR for the aforesaid image databases are greater than 50 dB. Also the NCC, SSIM and Q-Index values of the proposed scheme are close to one, which establish the efficiency of RWS-CA.

The comparison with respect to PSNR (dB) and payload (bpp) for Lena, Airplane, Pepper, Baboon, Tiffany and Boat images are presented in Table 3.8 and graphical result is depicted in Fig. 3.22. From Fig. 3.22 it is seen that RWS-CA furnishes better results in terms of capacity, compared with other existing schemes.

3.2.3.2 Robustness Analysis

The quality metrics such as NCC [94], BER [65], SD and CC [35] are evaluated to presents the robustness of RWS-CA. Also, RWS-CA has been assessed against salt and pepper noise, cropping and copy-move forgery attacks.

The statistical distortion of RWS-CA is evaluated using statistical parameters like SD (σ)

Table 3.7: MSE, PSNR, NCC, SSIM, Q-Index and BER results for different images in RWS-CA

Image Dataset	Images	MSE	PSNR (dB)	NCC	SSIM	Q-Index	BER
HDR [26]	Bird	1.067	47.88	0.99996	99.37%	0.9999	0.019
	Jerusalem	1.108	49.70	0.99994	99.32%	0.9998	0.019
	Redrock2	1.011	50.11	0.99996	99.43%	0.9999	0.019
	Safari04	1.026	50.03	0.99997	99.26%	0.9999	0.019
	Average	1.036	50.01	0.99995	99.30%	0.99985	0.0187
SIPI [90]	Lenna	0.985	50.22	0.99998	99.46%	0.9999	0.019
	Baboon	1.043	49.97	0.99998	99.57%	0.9999	0.019
	Tiffany	1.139	49.57	0.99999	99.13%	0.9996	0.019
	Barbara	1.056	49.91	0.99997	99.39%	0.9999	0.019
	Average	1.0449	49.96	0.99998	99.34%	0.99986	0.0185
UCID [61]	ucid00085	1.056	47.91	0.99997	99.78%	0.9999	0.019
	ucid00091	1.048	49.95	0.99997	99.56%	0.9999	0.019
	ucid00104	1.042	49.97	0.99998	99.42%	0.9999	0.019
	ucid00341	1.059	49.90	0.99996	99.78%	0.9999	0.019
	Average	1.1063	49.88	0.99997	99.45%	0.9999	0.019
STARE [89]	im0001	1.127	47.62	0.99995	98.17%	0.9998	0.018
	im0370	1.057	49.91	0.99997	98.68%	0.9999	0.018
	im0371	1.064	49.89	0.99996	98.61%	0.9999	0.019
	im0373	1.057	49.91	0.99997	98.60%	0.9999	0.018
	Average	1.076	49.83	0.99996	98.46 %	0.99987	0.0184

Table 3.8: Comparison of different RWT in sub-sample image with respect to PSNR and embedding capacity in RWS-CA

Image	Lin & Tsai [52]		Chang et al. [11]		Parah et al. [65]		Shin & Jung [75]		Lin & Chang [53]		RWS-CA	
	PSNR	Bpp	PSNR	Bpp	PSNR	Bpp	PSNR	Bpp	PSNR	Bpp	PSNR	Bpp
Lena	39.16	1/4	40.92	(t-1)/3	40.58	3/64	45.13	(t-2)/4	46.95	1/2	50.22	3/2
Baboon	39.15	1/4	40.92	(t-1)/3	39.60	3/64	43.9	(t-2)/4	46.92	1/2	49.97	3/2
Airplane	39.21	1/4	40.87	(t-1)/3	41.18	3/64	45.48	(t-2)/4	46.90	1/2	49.57	3/2
Peppers	39.20	1/4	40.96	(t-1)/3	40.43	3/64	44.41	(t-2)/4	46.85	1/2	50.18	3/2
Boat	39.18	1/4	40.93	(t-1)/3	41.32	3/64	44.11	(t-2)/4	46.96	1/2	49.96	3/2
Tiffany	39.13	1/4	40.89	(t-1)/3	41.35	3/64	45.1	(t-2)/4	46.84	1/2	50.13	3/2
Average	39.18	1/4	40.92	(t-1)/3	40.74	3/64	44.68	(t-2)/4	46.91	1/2	50.01	3/2

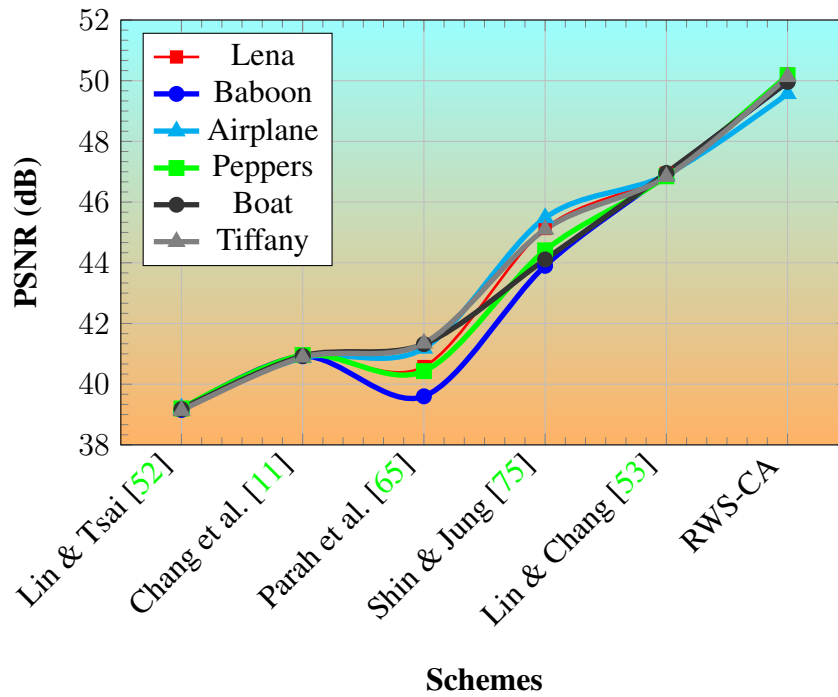


Figure 3.22: Comparison graph in terms PSNR (dB) with existing schemes in RWS-CA and CC (ρ) which has been calculated by using the equation (2.9) and (2.10) respectively. The experimental outcomes are illustrated in Table 3.9. From Table 3.9, it is observed that there is no substantial difference of σ between the CI and WI. Average σ value of CI and WI is 127.8765 and 127.7015 respectively, and their difference is 0.175 for Lena image after embedding 3,93,216 bits watermark. The average ρ value within the CI and WI is 0.9999 for Lena image. So finding the watermark from the WI become quite difficult. This results represent that RWS-CA provides a good camouflage of watermark, decrease the probability of watermark detection and ensures the robustness of the scheme.

3.2.3.3 Tamper Detection and Recovery

Robustness of RWS-CA is analyzed by evaluating the quality metrics such as PSNR, SSIM, Q-Index, NCC and BER in presence of salt and pepper noise, cropping and copy-move forgery attacks. The attacking results on RWS-CA after applying salt and pepper noise, cropping and copy-move forgery attack with different noise density level are depicted in Fig. 3.23, Fig. 3.24 and Fig. 3.25 respectively. It is clear that after extraction, the objective quality of the extracted watermark is slightly changed but recovered cover image has been identified successfully. Also the BER and NCC results of cover and extracted cover image and as well as BER and NCC

Table 3.9: SD and CC results for different image datasets RWS-CA

Database	Image	σ of CI (avg.)	σ of WI (avg.)	ρ (avg.)
SIPI [90]	Lenna	127.8765	127.7015	0.9999
	Baboon	123.1512	123.1908	0.9999
	Tiffany	77.2014	76.9679	0.9997
	Barbara	141.2811	141.3621	0.9999
	Zelda	137.5968	137.6723	0.9999
	Pepper	135.3631	135.3238	0.9999
	BoatsColor	161.0483	161.0385	0.9999
UCID [61]	Ucid00006	218.9172	218.8121	0.9999
	Ucid00085	173.9327	173.9272	0.9999
	Ucid00091	176.5012	176.6390	0.9999
	Ucid00104	171.1419	171.0576	0.9999
	Ucid00341	138.9078	138.9214	0.9999
	Ucid00786	143.0516	143.0617	0.9999
	Ucid00797	245.1584	244.6556	0.9999
HDR [26]	bird	140.2114	139.6596	0.9999
	jerusalem	118.9881	119.0321	0.9998
	redrock2	199.4528	199.6689	0.9999
	sedona01	150.8240	150.9086	0.9999
	stonehse	185.1848	184.7796	0.9999
STARE [89]	im0001	104.4130	104.6566	0.9998
	im0370	188.0491	188.0544	0.9999
	im0371	146.7582	146.7793	0.9999
	im0373	172.8893	172.8928	0.9999
	im0374	108.2031	108.2079	0.9998

results of watermark and extracted watermark exhibits the robustness of proposed scheme. The different objective metrics are presented in Table 3.10 when extraction is performed from tamper image. From Table 3.10, it is noted that the less BER values and near to unity Q-Index and NCC values indicate the robustness of proposed method during some standard attacks. Moreover, Table 3.10 represents that robustness of RWS-CA varies inversely with the noise density.

The algorithmic complexity and the computation time of any watermarking scheme is an important parameters to present the effectiveness of the scheme. The execution time of RWS-CA has been compared with some recent schemes [65, 78, 91] and comparison results are presented in Table 3.11. It is found that RWS-CA requires 0.563 seconds for total execution which is

Table 3.10: PSNR, SSIM, Q-Index, NCC and BER results of distorted watermark images due to salt pepper noise, cropping and copy-move forgery attacks in RWS-CA

Noise	Sample	Perturbation	PSNR		SSIM		Q-Index		NCC		BER		
			CI	WI	CI	WI	CI	WI	CI	WI	CI	WI	
Salt and Pepper	C1	0.01	25.26	20.30	77.64	67.92	0.9541	0.9739	0.9949	0.9940	0.0016	0.0065	
		0.1	21.36	16.43	57.69	53.63	0.8926	0.9372	0.9876	0.9855	0.0040	0.0157	
		0.5	18.62	13.55	39.02	44.93	0.8126	0.8753	0.9769	0.9714	0.0075	0.0308	
	C2	0.01	25.36	20.40	77.57	68.32	0.9542	0.9737	0.9950	0.9942	0.0015	0.0064	
		0.1	21.34	16.45	57.67	53.66	0.8929	0.9371	0.9877	0.9859	0.0042	0.0159	
		0.5	18.62	13.58	39.06	44.96	0.8125	0.8754	0.9762	0.9716	0.0076	0.0307	
	C3	0.01	25.63	20.25	77.89	67.87	0.9543	0.9736	0.9947	0.9941	0.0014	0.0059	
		0.1	21.37	16.48	57.64	53.61	0.8927	0.9377	0.9871	0.9851	0.0044	0.0154	
		0.5	18.62	13.56	39.07	44.97	0.8127	0.8755	0.9767	0.9717	0.0077	0.0304	
	C4	0.01	25.21	20.34	77.89	67.98	0.9541	0.9735	0.9948	0.9945	0.0012	0.0062	
		0.1	21.38	16.44	57.66	53.67	0.8921	0.9379	0.9873	0.9857	0.0048	0.0156	
		0.5	18.65	13.54	39.08	44.95	0.8124	0.8757	0.9764	0.9715	0.0074	0.0296	
	C1 & C2	0.01	22.23	17.55	61.37	57.53	0.9110	0.9479	0.9898	0.9888	0.0030	0.0135	
		0.1	19.32	15.53	42.14	46.92	0.8926	0.8726	0.9876	0.9855	0.0057	0.0155	
		0.5	15.60	12.68	27.54	34.56	0.6961	0.7123	0.9546	0.9483	0.0151	0.0543	
	C1 & C2 & C3	0.01	20.46	15.79	49.58	51.83	0.8707	0.9202	0.9848	0.9832	0.0049	0.0158	
		0.1	17.54	13.59	36.64	45.26	0.8926	0.8294	0.9526	0.9545	0.0089	0.0249	
		0.5	14.16	10.15	19.34	29.34	0.5934	0.6084	0.9364	0.9341	0.0234	0.0697	
	C1 & C2 & C3 & C4	0.01	19.23	14.63	41.15	48.87	0.8341	0.8955	0.9799	0.9781	0.0060	0.0245	
		0.1	15.46	11.71	22.35	34.12	0.6423	0.6821	0.9456	0.9451	0.0103	0.0642	
		0.5	12.57	8.26	12.57	23.80	0.4798	0.5359	0.9188	0.9178	0.0312	0.0912	
	Cropping	C1	10 %	21.01	16.20	91.73	90.12	0.8921	0.9240	0.9871	0.9846	0.0043	0.0144
			25 %	17.31	10.27	79.88	73.63	0.7809	0.7184	0.9728	0.9381	0.0109	0.0448
			50 %	13.46	5.82	59.64	47.89	0.5441	0.4087	0.9425	0.8189	0.0231	0.1021
C2		10 %	21.03	16.21	91.77	90.15	0.8915	0.9241	0.9873	0.9844	0.0045	0.0145	
		25 %	17.32	10.26	79.87	73.67	0.7806	0.7182	0.9724	0.9386	0.0107	0.0442	
		50 %	13.43	5.81	59.61	47.87	0.5442	0.4089	0.9423	0.8183	0.0232	0.1028	
C3		10 %	21.05	16.22	91.71	90.13	0.8917	0.9239	0.9872	0.9845	0.0041	0.0147	
		25 %	17.33	10.24	79.84	73.66	0.7803	0.7186	0.9729	0.9383	0.0105	0.0444	
		50 %	13.49	5.86	59.66	47.84	0.5446	0.4085	0.9429	0.8185	0.0239	0.1024	
C4		10 %	21.09	16.23	91.75	90.14	0.8919	0.9243	0.9876	0.9847	0.0047	0.0149	
		25 %	17.35	10.21	79.83	73.61	0.7801	0.7189	0.9722	0.9387	0.0102	0.0446	
		50 %	13.47	5.87	59.67	47.91	0.5444	0.4086	0.9424	0.8186	0.0236	0.1026	
C1 & C2		10 %	17.80	14.31	90.35	86.44	0.7992	0.8816	0.9751	0.9760	0.0095	0.0183	
		25 %	14.56	11.76	73.45	63.49	0.4538	0.7456	0.9456	0.9358	0.0153	0.0453	
		50 %	10.46	8.56	56.23	47.84	0.3296	0.5556	0.9109	0.9082	0.0461	0.0761	
C1 & C2 & C3		10 %	16.08	15.05	90.41	88.08	0.7303	0.8991	0.9652	0.9799	0.0141	0.0183	
		25 %	12.25	10.23	69.23	61.91	0.4315	0.6724	0.9236	0.8564	0.0456	0.0546	
		50 %	8.16	6.23	54.36	46.52	0.2463	0.4561	0.8986	0.7643	0.07324	0.0953	
C1 & C2 & C3 & C4		10 %	14.85	11.17	95.64	88.17	0.6736	0.7784	0.9563	0.9499	0.0188	0.0228	
		25 %	11.23	9.26	68.49	59.57	0.4126	0.5482	0.9146	0.8357	0.0413	0.0654	
		50 %	7.45	4.35	52.65	44.53	0.0831	0.3188	0.8829	0.7284	0.0922	0.1091	
Copy Move Forgery		C1	5 %	27.29	22.06	98.67	96.51	0.9934	0.9804	0.9999	0.9962	0.0003	0.0037
			10 %	26.35	20.94	98.47	94.56	0.9885	0.9748	0.9999	0.9938	0.0009	0.0048
			20 %	25.38	18.12	98.32	91.56	0.9814	0.9556	0.9999	0.9901	0.0014	0.0081
	C2	5 %	27.25	22.05	98.65	96.54	0.9950	0.9804	0.9999	0.9965	0.0002	0.0038	
		10 %	26.33	20.91	98.44	94.54	0.9883	0.9744	0.9999	0.9936	0.0010	0.0046	
		20 %	25.36	18.13	98.33	91.54	0.9869	0.9558	0.9999	0.9903	0.0015	0.0082	
	C3	5 %	27.55	22.04	98.66	96.52	0.9899	0.9804	0.9999	0.9964	0.0001	0.0036	
		10 %	26.36	20.92	98.42	94.52	0.9898	0.9746	0.9999	0.9934	0.0006	0.0047	
		20 %	25.35	18.16	98.34	91.51	0.9897	0.9554	0.9999	0.9904	0.0012	0.0085	
	C4	5 %	27.29	22.06	98.63	96.53	0.9837	0.9804	0.9981	0.9962	0.0002	0.0035	
		10 %	26.57	20.95	98.48	94.51	0.9874	0.9742	0.9963	0.9935	0.0008	0.0044	
		20 %	25.73	18.11	98.31	91.53	0.9814	0.9553	0.99945	0.9900	0.0013	0.0084	
	C1 & C2	5 %	27.25	21.77	98.52	96.20	0.9898	0.9805	0.9927	0.9957	0.0011	0.0044	
		10 %	25.60	18.94	97.12	93.48	0.9897	0.9604	0.9965	0.9902	0.0018	0.0076	
		20 %	24.79	16.90	97.89	89.84	0.9895	0.9403	0.9933	0.9868	0.0025	0.0107	
	C1 & C2 & C3	5 %	26.55	20.36	97.45	95.89	0.9898	0.9745	0.9902	0.9938	0.0016	0.0051	
		10 %	24.95	18.12	97.96	91.23	0.9896	0.9536	0.9869	0.9899	0.0024	0.0392	
		20 %	23.05	16.13	97.46	88.54	0.9893	0.9356	0.9935	0.9870	0.0038	0.0703	
	C1 & C2 & C3 & C4	5 %	27.25	19.52	96.30	95.26	0.9897	0.9659	0.9973	0.9928	0.0021	0.0059	
		10 %	24.66	17.39	96.94	91.57	0.9894	0.9456	0.9810	0.9903	0.0036	0.0729	
		20 %	22.72	15.72	96.21	87.64	0.9891	0.9224	0.9750	0.9872	0.0052	0.1401	

Cover Image (510x510) (C)	Watermark (170 x170)	watermarked image(CW1)	watermarked image(CW2)	watermarked image(CW3)	watermarked image(CW4)	Recovered watermark	Recovered Cover Image	Experimental Results
	WATERMARK					WATERMARK		HCC(C & C) = 0.9949 HCC(W & W) = 0.9948 BER (C & C) = 0.0016 BER (W & W) = 0.0065
Lena	Logo Image	SaltPepper 0.01	No Attack	No Attack	No Attack	PSNR=20.30(dB)	PSNR=25.26(dB)	
	WATERMARK					WATERMARK		HCC (C & C) = 0.9898 HCC (W & W) = 0.9888 BER (C & C) = 0.003 BER (W & W) = 0.0135
Lena	Logo Image	SaltPepper 0.01	SaltPepper 0.01	No Attack	No Attack	PSNR=17.55(dB)	PSNR=22.23(dB)	
	WATERMARK					WATERMARK		HCC (C & C) = 0.9848 HCC (W & W) = 0.9832 BER (C & C) = 0.0049 BER (W & W) = 0.0158
Lena	Logo Image	SaltPepper 0.01	SaltPepper 0.01	SaltPepper 0.01	No Attack	PSNR=15.79(dB)	PSNR=20.16(dB)	
	WATERMARK					WATERMARK		HCC (C & C) = 0.9799 HCC (W & W) = 0.9781 BER (C & C) = 0.0060 BER (W & W) = 0.0245
Lena	Logo Image	SaltPepper 0.01	SaltPepper 0.01	SaltPepper 0.01	SaltPepper 0.01	PSNR=19.23(dB)	PSNR=14.63(dB)	
	WATERMARK					WATERMARK		HCC (C & C) = 0.9876 HCC (W & W) = 0.9855 BER (C & C) = 0.0040 BER (W & W) = 0.0157
Lena	Logo Image	SaltPepper 0.1	No Attack	No Attack	No Attack	PSNR=16.43(dB)	PSNR=21.36(dB)	
	WATERMARK					WATERMARK		HCC (C & C) = 0.9876 HCC (W & W) = 0.9855 BER (C & C) = 0.0057 BER (W & W) = 0.0155
Lena	Logo Image	SaltPepper 0.1	SaltPepper 0.1	No Attack	No Attack	PSNR=15.53(dB)	PSNR=19.32(dB)	
	WATERMARK					WATERMARK		HCC (C & C) = 0.9526 HCC (W & W) = 0.9545 BER (C & C) = 0.0089 BER (W & W) = 0.0249
Lena	Logo Image	SaltPepper 0.1	SaltPepper 0.1	SaltPepper 0.1	No Attack	PSNR=13.59(dB)	PSNR=17.54(dB)	
	WATERMARK					WATERMARK		HCC (C & C) = 0.9156 HCC (W & W) = 0.9451 BER (C & C) = 0.0103 BER (W & W) = 0.0642
Lena	Logo Image	SaltPepper 0.1	SaltPepper 0.1	SaltPepper 0.1	SaltPepper 0.5	PSNR=11.71(dB)	PSNR=15.16(dB)	
	WATERMARK					WATERMARK		HCC (C & C) = 0.9769 HCC (W & W) = 0.9711 BER (C & C) = 0.0075 BER (W & W) = 0.0308
Lena	Logo Image	SaltPepper 0.5	No Attack	No Attack	No Attack	PSNR=13.55(dB)	PSNR=18.62(dB)	
	WATERMARK					WATERMARK		HCC (C & C) = 0.9546 HCC (W & W) = 0.9483 BER (C & C) = 0.0151 BER (W & W) = 0.0543
Lena	Logo Image	SaltPepper 0.5	SaltPepper 0.5	No Attack	No Attack	PSNR=12.68(dB)	PSNR=15.60(dB)	
	WATERMARK					WATERMARK		HCC (C & C) = 0.9361 HCC (W & W) = 0.9341 BER (C & C) = 0.0231 BER (W & W) = 0.0697
Lena	Logo Image	SaltPepper 0.5	SaltPepper 0.5	SaltPepper 0.5	No Attack	PSNR=10.15(dB)	PSNR=14.16(dB)	
	WATERMARK					WATERMARK		HCC (C & C) = 0.9188 HCC (W & W) = 0.9178 BER (C & C) = 0.0312 BER (W & W) = 0.0912
Lena	Logo Image	SaltPepper 0.5	SaltPepper 0.5	SaltPepper 0.5	SaltPepper 0.5	PSNR=8.26(dB)	PSNR=12.57(dB)	

Figure 3.23: Effect of salt pepper noise on Lena image in RWS-CA

0.3315 seconds, 0.5532 seconds and 0.0894 seconds faster than Su et al. [78], Verma et al. [91] and Parah et al. [65] schemes respectively. During embedding only 0.504 seconds time is ac-

Cover Image (510×510) (C)	Watermark (170 ×170)	watermarked image(CW1)	watermarked image(CW2)	watermarked image(CW3)	watermarked image(CW4)	Recovered watermark	Recovered Cover Image	Experimental Results
								HCC(C & C') = 0.9871 HCC(W & W')= 0.9946 BER (C & C') = 0.0043 BER (W & W')=0.0141
Lena	Logo Image	Cropping 10%	No Attack	No Attack	No Attack	PSNR=20.30(dB)	PSNR=25.26(dB)	
								HCC(C & C') = 0.9751 HCC(W & W')= 0.9769 BER (C & C') = 0.0095 BER (W & W')=0.0183
Lena	Logo Image	Cropping 10%	Cropping 10%	No Attack	No Attack	PSNR=20.40(dB)	PSNR=25.36(dB)	
								HCC(C & C') = 0.9652 HCC(W & W')= 0.9799 BER (C & C') = 0.0141 BER (W & W')=0.0213
Lena	Logo Image	Cropping 10%	Cropping 10%	Cropping 10%	No Attack	PSNR=14.87(dB)	PSNR=31.67(dB)	
								HCC(C & C') = 0.9563 HCC(W & W')= 0.9499 BER (C & C') = 0.0188 BER (W & W')=0.0228
Lena	Logo Image	Cropping 10%	Cropping 10%	Cropping 10%	Cropping 10%	PSNR=6.45(dB)	PSNR=11.23(dB)	
								HCC(C & C') = 0.9728 HCC(W & W')= 0.9381 BER (C & C') = 0.0109 BER (W & W')=0.0448
Lena	Logo Image	Cropping 25%	No Attack	No Attack	No Attack	PSNR=8.67(dB)	PSNR=13.36(dB)	
								HCC(C & C') = 0.9156 HCC(W & W')= 0.9358 BER (C & C') = 0.0153 BER (W & W')=0.0463
Lena	Logo Image	Cropping 25%	Cropping 25%	No Attack	No Attack	PSNR=7.36(dB)	PSNR=16.34(dB)	
								HCC(C & C') = 0.9236 HCC(W & W')= 0.8561 BER (C & C') = 0.0456 BER (W & W')=0.0546
Lena	Logo Image	Cropping 25%	Cropping 25%	Cropping 25%	No Attack	PSNR=6.37(dB)	PSNR=35.62(dB)	
								HCC(C & C') = 0.9446 HCC(W & W')= 0.8356 BER (C & C') = 0.0413 BER (W & W')=0.0564
Lena	Logo Image	Cropping 25%	Cropping 25%	Cropping 25%	Cropping 25%	PSNR=6.35(dB)	PSNR=36.28(dB)	
								HCC(C & C') = 0.9125 HCC(W & W')= 0.8189 BER (C & C') = 0.0231 BER (W & W')=0.1021
Lena	Logo Image	Cropping 50%	No Attack	No Attack	No Attack	PSNR=5.23(dB)	PSNR=32.64(dB)	
								HCC(C & C') = 0.9189 HCC(W & W')= 0.9082 BER (C & C') = 0.0161 BER (W & W')=0.0761
Lena	Logo Image	Cropping 50%	Cropping 50%	No Attack	No Attack	PSNR=5.23(dB)	PSNR=32.64(dB)	
								HCC(C & C') = 0.8986 HCC(W & W')= 0.7643 BER (C & C') = 0.0732 BER (W & W')=0.0953
Lena	Logo Image	Cropping 50%	Cropping 50%	Cropping 50%	No Attack	PSNR=5.23(dB)	PSNR=32.64(dB)	
								HCC(C & C') = 0.8829 HCC(W & W')= 0.7281 BER (C & C') = 0.0922 BER (W & W')=0.1091
Lena	Logo Image	Cropping 50%	Cropping 50%	Cropping 50%	Cropping 50%	PSNR=5.23(dB)	PSNR=32.64(dB)	

Figure 3.24: Effect of cropping attacks on Lena image in RWS-CA

quired to insert (256 × 64) watermark image (i.e., 3,93,216 bits) into (512 × 512) that is 6,291,456 bits cover image and 0.059 seconds time acquired at the time of extraction. The

Cover Image (510×510) (C)	Watermark (170 ×170)	watermarked image(CW1)	watermarked image(CW2)	watermarked image(CW3)	watermarked image(CW4)	Recovered watermark	Recovered Cover Image	Experimental Results
								HCC (C & C') = 0.9981 HCC (W & W') = 0.9962 BER (C & C') = 0.0003 BER (W & W') = 0.0037
Lena	Logo Image	CopyMove 5%	No Attack	No Attack	No Attack	PSNR=27.06(dB)	PSNR=34.29(dB)	
								HCC (C & C') = 0.9963 HCC (W & W') = 0.9957 BER (C & C') = 0.0011 BER (W & W') = 0.0041
Lena	Logo Image	CopyMove 5%	CopyMove 5%	No Attack	No Attack	PSNR=21.77(dB)	PSNR=31.25(dB)	
								HCC (C & C') = 0.9945 HCC (W & W') = 0.9938 BER (C & C') = 0.0016 BER (W & W') = 0.0051
Lena	Logo Image	CopyMove 5%	CopyMove 5%	CopyMove 5%	No Attack	PSNR=20.36(dB)	PSNR=29.55(dB)	
								HCC (C & C') = 0.9927 HCC (W & W') = 0.9928 BER (C & C') = 0.0021 BER (W & W') = 0.0059
Lena	Logo Image	CopyMove 5%	CopyMove 5%	CopyMove 5%	CopyMove 5%	PSNR=19.52(dB)	PSNR=28.25(dB)	
								HCC (C & C') = 0.9965 HCC (W & W') = 0.9938 BER (C & C') = 0.0009 BER (W & W') = 0.0048
Lena	Logo Image	CopyMove 10%	No Attack	No Attack	No Attack	PSNR=26.91(dB)	PSNR=31.57(dB)	
								HCC (C & C') = 0.9933 HCC (W & W') = 0.7643 BER (C & C') = 0.0732 BER (W & W') = 0.0953
Lena	Logo Image	CopyMove 10%	CopyMove 10%	No Attack	No Attack	PSNR=18.91(dB)	PSNR=28.60(dB)	
								HCC (C & C') = 0.9902 HCC (W & W') = 0.9899 BER (C & C') = 0.0021 BER (W & W') = 0.0392
Lena	Logo Image	CopyMove 10%	CopyMove 10%	CopyMove 10%	No Attack	PSNR=18.12(dB)	PSNR=26.95(dB)	
								HCC (C & C') = 0.9869 HCC (W & W') = 0.9903 BER (C & C') = 0.0036 BER (W & W') = 0.0729
Lena	Logo Image	CopyMove 10%	CopyMove 10%	CopyMove 10%	CopyMove 10%	PSNR=17.39(dB)	PSNR=25.66(dB)	
								HCC (C & C') = 0.9854 HCC (W & W') = 0.9965 BER (C & C') = 0.0011 BER (W & W') = 0.0081
Lena	Logo Image	CopyMove 20%	No Attack	No Attack	No Attack	PSNR=25.12(dB)	PSNR=28.73(dB)	
								HCC (C & C') = 0.9873 HCC (W & W') = 0.9868 BER (C & C') = 0.0025 BER (W & W') = 0.0107
Lena	Logo Image	CopyMove 20%	CopyMove 20%	No Attack	No Attack	PSNR=16.90(dB)	PSNR=25.79(dB)	
								HCC (C & C') = 0.9810 HCC (W & W') = 0.9870 BER (C & C') = 0.0038 BER (W & W') = 0.0703
Lena	Logo Image	CopyMove 20%	CopyMove 20%	CopyMove 50%	No Attack	PSNR=16.13(dB)	PSNR=24.05(dB)	
								HCC (C & C') = 0.9750 HCC (W & W') = 0.9872 BER (C & C') = 0.0052 BER (W & W') = 0.1401
Lena	Logo Image	CopyMove 20%	CopyMove 20%	CopyMove 20%	CopyMove 20%	PSNR=15.72(dB)	PSNR=22.72(dB)	

Figure 3.25: Effect of copy move forgery attacks on Lena image in RWS-CA

lesser execution time in RWS-CA is achieved due to simple algebraic manipulations and the threading concept of Java. A cover image size of $(M \times N)$ to determine the algorithmic com-

Table 3.11: Comparison table in terms of computation time in RWS-CA

Schemes	Size of Image	Embedding time (sec)	Extraction time (sec)	Total time (sec)
Su et al. [78]	512×512	0.5244	0.3701	0.8945
Verma et al. [91]	512×512	0.5173	0.5989	1.1162
Parah et al. [65]	512×512	0.59	0.0624	0.6524
RWS-CA	512×512	0.504	0.059	0.563

plexity. Time complexity for doing the operations described in Algorithm 3.3 is $\mathcal{O}(MN)$. On the other hand, at the time of extraction, the complexity is $\mathcal{O}(MN)$, considering Algorithm 3.4.

3.3 Discussion

In this chapter we are trying to overcome the first and foremost challenges in image watermarking scheme that is, to achieve authentication and tamper detection in addition to maintain a good trade off among capacity, imperceptibility and robustness. In RWS-WM, weighted matrix has



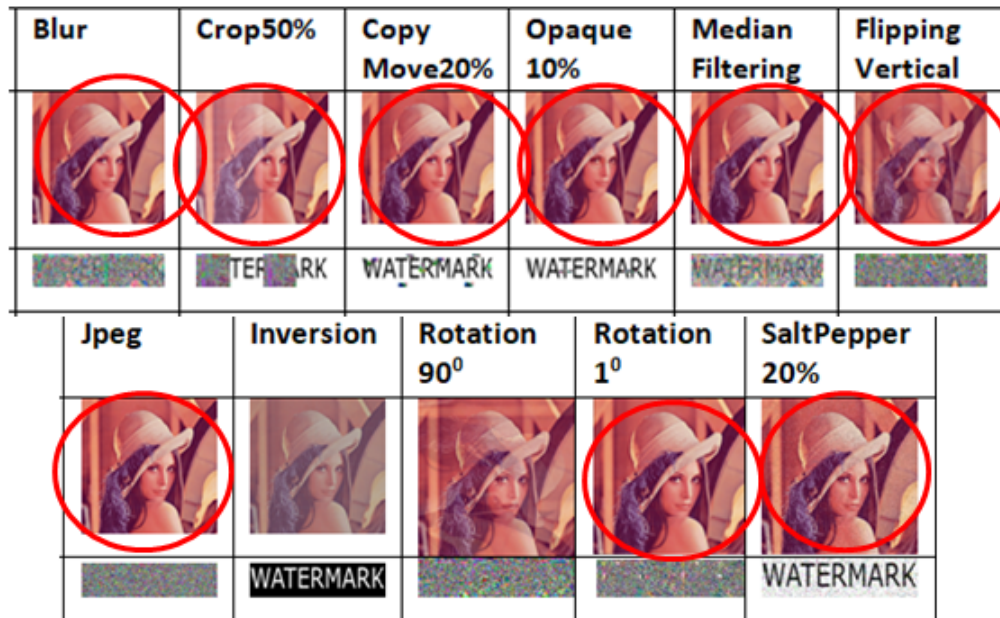
Cover image recovered: 8

Watermark recovered: 4

Figure 3.26: Effects of different types of attacks on Lena image for RWS-WM

been employed to increase embedding capacity while maintaining good visual quality and the experimental results shows that we are successful to do so. But when we analyse our scheme in terms of robustness, it has been seen that RWS-WM can only resist four types of attack and

the cover image can be successfully recovered from eight types of attacks after performing ten special types of attacks shown in Fig. 3.26.



Cover image recovered: 9
 Watermark recovered: 5

Figure 3.27: Effects of different types of attacks on Lena image for RWS-CA

Table 3.12: Effects of 10 different types of attacks

Schemes	Image Recovered	Salt Pepper	Cropping	Copy move	Opaque	Median Filtering	Flipping (Vertical)	JPEG Compression	Blurring	Rotation	Inverction
RWS-WM	CI	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗
	W	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
RWS-CA	CI	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
	W	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓

So, to increase robustness a new watermarking scheme has been developed in sub-sample image with the help of special type one dimensional periodic cellular automata named CAA shown in RWS-CA. Though CA is used to increase security, especially for secret sharing but capacity of this watermarking scheme is reasonably low. Moreover in the sense of robustness it has been seen that the proposed scheme can resist only one more attack i.e. five types of attack and cover image can recover successfully from nine types of attacks after performing ten special types of attacks depicted in Fig. 3.27. The overall outcomes are tabulated in Table. 3.12.

3.3.1 Salient Feature of this Chapter

- In this chapter, authentication has been achieved successfully in both the schemes. An InF is used to attain authentication in RWS-WM and SHA-512 algorithm is used to generate authentication code in RWS-CA.
- Here, a shared secret key has been considered to enhance the security of both the schemes. Shared secret key has been XORed with watermark bits and generates an encrypted message which is used in CA for distribution in the sub-sampled image to increase the security of RWS-CA. It is tough to extract watermark information without knowing shared secret key and proper CA rule. Moreover, modified WM and CAA have been applied to strengthen the security of RWS-WM and RWS-CA respectively.
- Sharing the watermark into four different sub-sample images gives the flavour of secret sharing in RWS-CA. So, it is hard to extract watermark information without all the sub-sampled images. Moreover, interpolation in sub-sample images is used to increase embedding capacity, security and achieve reversibility.
- Payload is increased using repeated embedding within same block in RWS-WM.

So to increase the robustness, we are developed our schemes and employ local binary pattern (LBP) operator in watermarking schemes. We have presented two more scheme DRWS-LBP and RWS-LBP-HC in chapter 4 to perform authentication along with tamper detection.