Chapter 4

Dual Image Based Steganographic Scheme (DS)

Reversibility is one of the major concern of modern steganographic schemes. Single image based steganographic schemes suffer from irreversibility when the embedding capacity of the scheme is increased. So, designing a reversible steganographic scheme with high embedding capacity along with good visual quality is the need of current research scenario. Here, dual image based reversible steganographic schemes have been developed that exhibits good imperceptibility along with high embedding capacity.

In dual image based steganography, two stego images are created. The main advantages of using dual images in steganography are information sharing, increased payload, enhanced quality and reversibility. The secret information can be shared by embedding the secret message into both of the stego images. The payload of these schemes can be increased as two stego images are used. Some techniques, used in steganography, are not reversible. For those schemes, reversibility can be achieved using dual images. Also, high quality stego images can be generated as the information is shared between two images instead of one image.

In this chapter, two different approaches have been taken to design steganographic schemes using dual image. In the first approach, a dual image based stenographic scheme has been proposed using graph neighbourhood. The graph-based scheme with single stego image has good embedding capacity and generates high-quality stego image. But, it is neither reversible nor secured. Here, the reversibility has been achieved using dual stego images, and the security has been introduced using a shared secret key (κ). The shared secret key is converted into a 512-bit key stream using a secured hash algorithm (SHA-512). The bits of the κ is checked one by one. If the bit is 0, the data is embedded into the pixels of the first stego image (SI1), and if the bit is 1, the data is embedded into the pixels of the second stego image (SI2). So, the secret key κ is used to distribute the secret message between two stego images (SI1 and SI2). The proposed technique exhibits high embedding capacity when compared to some of the standard techniques discussed in the literature. The standard metrics like PSNR, BER, SSIM, and Q-Index are used to evaluate the proposed scheme for different threshold values. The use of the weighted matrix during embedding helps to embed multiple data bits by changing only a single pixel of the pixel block. The concept of graph neighbourhood, used in this scheme, helps to create the pixel block with the pixels in the same range or group. This increases the visual quality and the embedding capacity of the stego image.

In the second approach, a weighted matrix based reversible steganographic scheme has been proposed with the help of dual images. Usually, a weighted matrix based steganographic scheme is not reversible. In this scheme, high payload with good visual quality is achieved, compared to other existing state-of-the-art schemes. The robustness of the proposed scheme is tested through different statistical analysis, and the results obtained during the statistical analysis are promising. The dual stego images are again examined against brute force attacks with an unknown secret key and unknown weighted matrix. It has been observed that the scheme is secure against these attacks.

4.1 DS Using Graph Neighbourhood (DSGN)

In any steganographic scheme, the main challenge is to maintain high imperceptibility of the stego image along with good embedding capacity. To address this issue, a dual image based steganographic scheme based on the concept of graph neighbourhood has been suggested. Two identical images are formed from the cover image. Instead of embedding the secret data in a single image, the secret data is embedded into both the images depending on the value of a shared secret key. Both the images are partitioned into (4×4) non-overlapping pixel blocks. In any pixel block, each and every pixel belongs to a group depending on a range of pixel values. A pixel may have a number of pixels of the same group around it. This number is the degree of neighbourhood of the pixel. The degree of neighbourhood of the pixels are calculated according to the graph neighbourhood degree method [33]. A threshold value is then considered. The pixels whose degree of neighbourhood is equal or more than this threshold value, are selected and grouped. The secret data is embedded in these selected blocks. Instead of embedding data into every pixel, a particular pixel among the group is selected, and secret data is embedded there. In this scheme, a similar group of pixels is selected, and a weighted matrix is employed for data embedding, which not only improves the payload but also enhances the visual quality of the stego image compared to the other existing schemes. Finally, the proposed technique is examined through different types of steganographic attacks and steganalysis to prove its imperceptibility and robustness. The detail embedding and extraction procedure have been described in Sections 4.1.1 and 4.1.2 respectively.

4.1.1 Data Embedding Procedure

The embedding process is depicted using a schematic diagram in Fig. 4.1. At first, a color cover image (C) of size $(m \times n)$ and a secret image (S) is taken. S is converted into a secret bitstream (BS). Dual images (CD1, CD2) are created from C and both CD1 and CD2 are partitioned into (4×4) pixel blocks. The shared secret key is converted to 512-bit stream (κ) using SHA-512. The randomness of the block selection has been done using the value of κ. If the first bit of κ is 1, the pixel block of the CD1 is selected for embedding, otherwise, the pixel block of CD2 is selected. In this way, each and every bit is checked and depending on the value of the bit, the next pixel block is selected either from CD1 or CD2. All the pixels in a pixel block are categorized according to their values mentioned in a range table. For example, a pixel with value 130 is of category 5, and a pixel with value 30 is of category 1. The neighbourhood degree of any pixel is the number of pixels, belonging to the same category, around it. A pixel has a neighbour degree of 2 means that it has two pixels around it, and both belong to the same category. Depending on a threshold value that lies between 0 to 8, the pixels from the pixel block are chosen to form a pixel matrix (PM). If the threshold value is t, all the pixels having neighbourhood degree t or above are chosen. For example, if the threshold value is 2, the pixels having a neighbourhood degree 2 or more are selected to form a matrix. So, the dimension of the matrix actually depends on how the pixels are distributed inside the pixel block. A weighted matrix (WM) is multiplied with the matrix PM using entry-wise multiplication. The sum of the elements (SUM) of the resultant matrix RM is calculated. Then, r bits are collected from secret bit stream BS and converted to its decimal equivalent (DS). The difference D is calculated as: $D = MOD(DS - SUM, 2^r)$. The embedding position POS is calculated as $POS = ABS(2^r - D)$. The pixel value at position POS is increased or decreased by 1 depending on the positive or negative value of POS respectively. No change is made in the pixel block PB if the POS value is zero. In this way, r bit secret data is embedded into the pixel block PB by changing only a single pixel of the pixel matrix. After embedding all the secret bits in CD1 and CD2, dual stego images ($SI1, SI2$) are generated. The embedding procedure of DSGN scheme is presented in Algorithm 4.1.

4.1.2 Data Extraction Procedure

The extraction process of DSGN scheme is depicted in Fig. 4.2 using a schematic diagram. During extraction, both the dual stego images ($SI1, SI2$) are taken and partitioned into (4×4) pixel blocks (PB). The range table, used during data embedding, is used to categorise the pixels according to their degree of neighbourhood. The shared secret key is converted to 512-bit binary key (κ) stream using the SHA-512 algorithm. Depending on the bit value 1 or 0, the pixel blocks are selected either from SI1 or SI2 respectively. After that, depending on the threshold value, the pixels are collected, and a pixel matrix is formed. The weighted matrix (WM) is multiplied with the pixel matrix PM using entry-wise multiplication. The sum of the elements (SUM) of the resultant matrix RM is calculated. The modulo of SUM is then calculated as $D' = MOD(SUM, 2^n)$. This D' is actually the extracted data, and it is converted

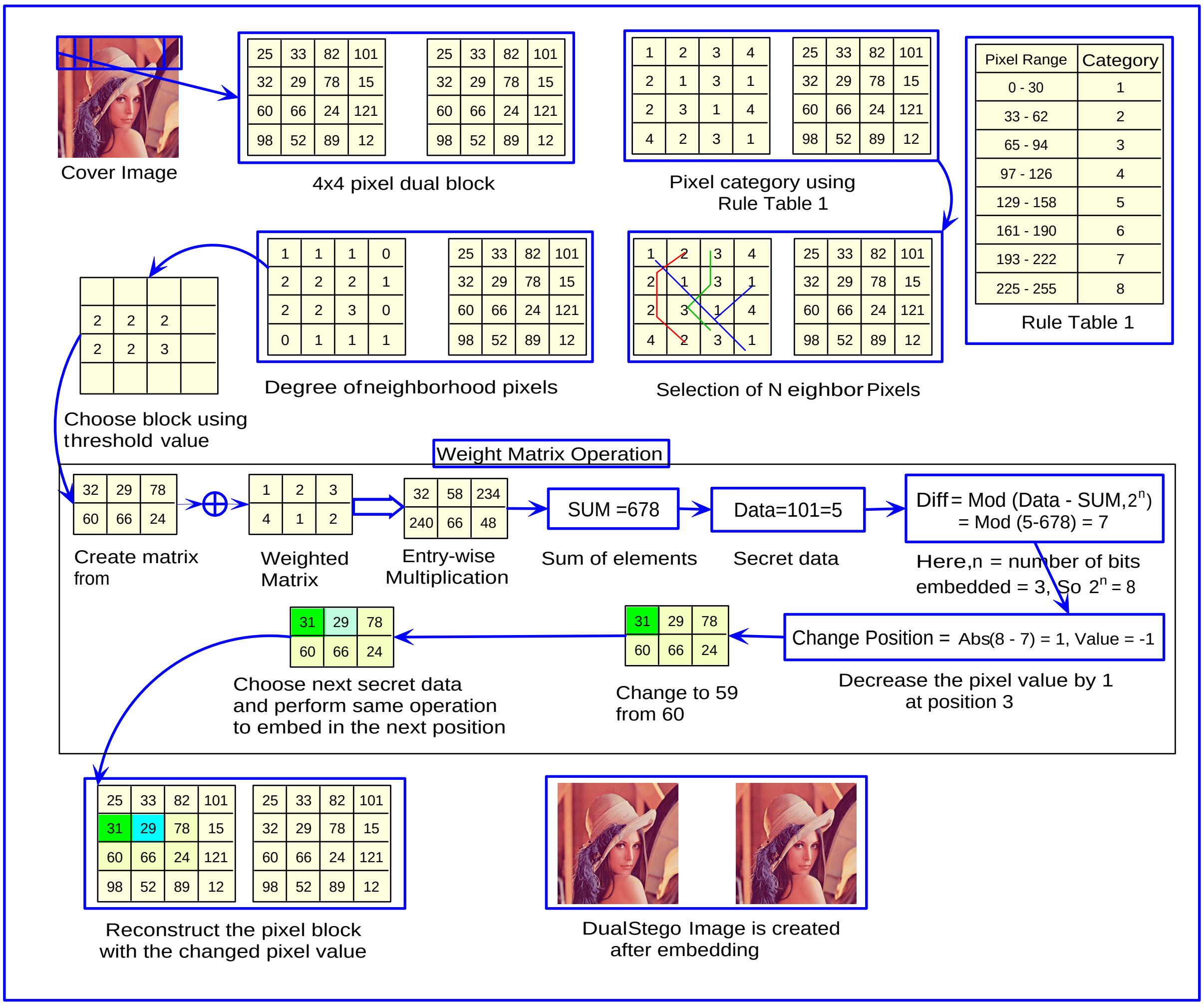


Figure 4.1: *Schematic diagram of data embedding process in DSGN*

to r bit binary string. Secret bits from all other pixel blocks of SI1 and SI2 are extracted and then merged to form the whole secret data and the secret image. Finally, the unchanged pixel blocks from SI1 and SI2 are collected and merged to generate the original cover image. The extraction procedure is presented in Algorithm 4.2.

4.1.3 Experimental Results and Comparisons

The scheme is tested using standard benchmark images shown in Fig. 1.1 and analysed using various evaluation metrics. The experimental results and comparisons are given below:

4.1.3.1 Quality Measurement Analysis

The distortion in the stego images is measured by Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index (SSIM). The PSNR is calculated using the equation 1.2. The higher

Algorithm 4.1: DSGN: Embedding Algorithm

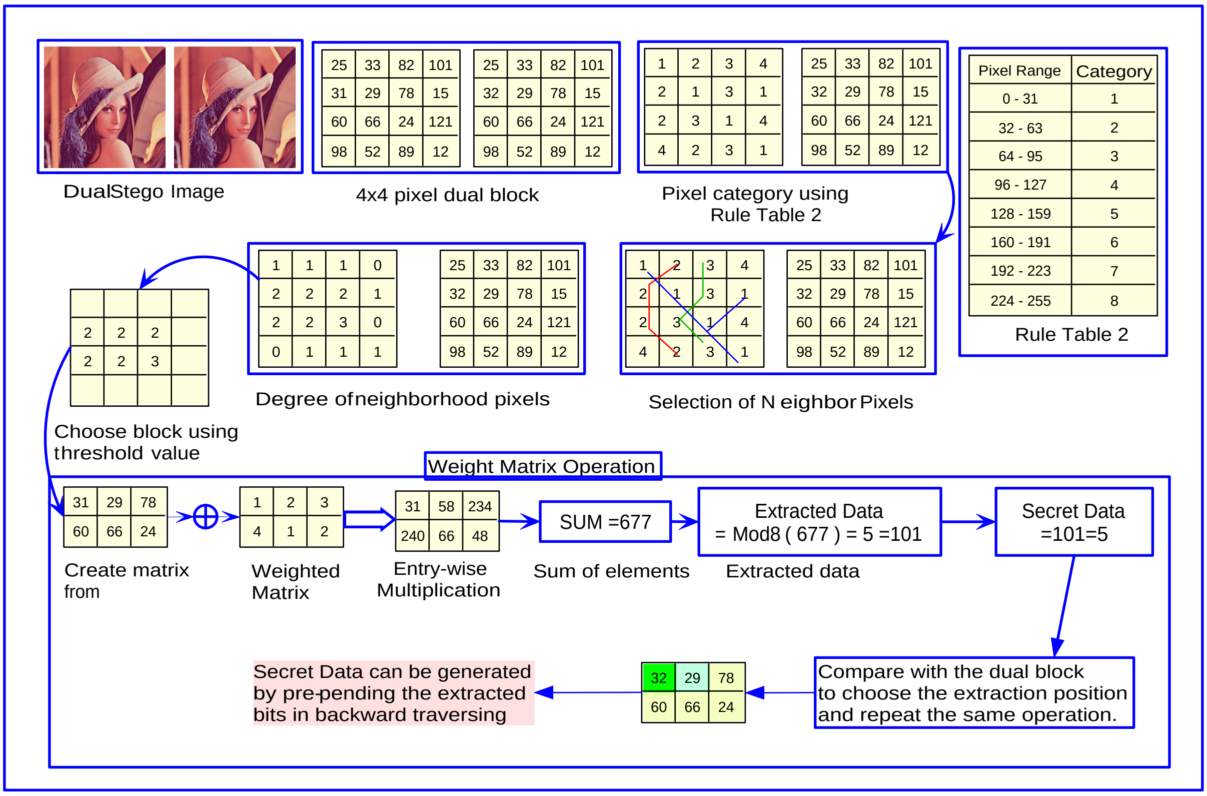
```
input  : A Cover Image of size $m \times$n, Secret Message, Weighted Matrix, Shared Secret Key
output: A Stego Image of size $m \times$n

Algorithm Embedding():
    Step-1: Get 512 bit binary sequence of the shared secret key using SHA-512
    keyBits[] = SHA-512(Shared Secret Key);
    bitIndex = 0;
    for (pixelBlock ← 1 to $(m \times n)/4$ ) do
        Step-2: A matrix $SM_{x \times y}$ is formed using the selected pixels from the $4 \times 4$ pixelBlock of either first dual image or second dual image depending on the
          key bit is 0 or 1
        if ( keyBits[bitIndex] $==$ 0) then
            // Create the pixel matrix from first dual image
            CreatePixelMatrix();
        else
            // Create the pixel matrix from second dual image
            CreatePixelMatrix();
        bitIndex++;
        Step-3: The secret data is embedded in the selected pixels
        EmbedSecretData();
    Step-4: The stego image is generated after embedding the secret bits

Function CreatePixelMatrix():
    Step-1: Get neighbourhood degree of all pixels in $4 \times 4$ block of the cover image
    Step-2: A threshold value is considered here
    Step-3: If the neighbourhood degree of a pixel is more than or equals to the threshold value, add the pixel in the $SM_{x \times y}$ matrix

Function EmbedSecretData():
    Step-1: An weighted matrix ($WM_{x \times y}$) having the same dimension of $SM_{x \times y}$ matrix is formed
    Step-2: An elementary multiplication of $SM_{x \times y}$ with $WM_{x \times y}$ is performed
    Step-3: The sum of elements ($SUM$) of the resultant matrix is calculated.
    for $i$ ← 1 to x do
        for $j$ ← 1 to y do
            SUM $+ = SM[i][j] \times WM[i][j]$
    Step-4: The $r$ number of bits are collected from secret bit stream and converted to its decimal equivalent $DE$
    Step-5: The difference $D$ is calculated by $D = MOD(DE - SUM, 2^r)$
    Step-6: The embedding position ($POS$) is the absolute value of $2^r - D$
    Step-7: The secret data is embedded by changing the value of pixel of the pixel block $PB$ in the position $POS$
    Step-8: The pixel value of $PB$ at position $POS$ is decreased or decreased by 1 depending on the value of $POS$ is negative or positive respectively
    Step-9: If the $POS$ value is zero, no change is made to the pixels of the pixel block $PB$
```

PSNR indicates better visual quality of the stego image. The SSIM is measured by the equation 1.4 and it is calculated on various window size of an image. The experimental results of embedding the secret image within some standard images like Lena, Airplane, Boat, Baboon, Pepper are tabulated in Table 4.1. PSNR, Payload, SSIM, and statistical analysis results of the proposed DSGN scheme have been analysed. It is observed that after hiding $7, 77, 965$ bits as secret data within the original image of size (512×512), the average PSNR is near 55 dB for Lena image. Table 4.2 represents comparison with existing dual image based data hiding schemes in terms of PSNR and Payload after embedding the maximum amount of secret data. The table also represents the PSNR of the color image and grayscale image, which are used in the proposed DSGN scheme. It has been observed that embedding capacity in color image is higher than the grayscale image without compromising the visual quality. The DSGN scheme can hide 7,77,965 bits within color Lena image. Table 4.3 shows PSNR (dB) values of different


DualStego Image
4x4 pixel dual block
Pixel category using Rule Table 2
Rule Table 2
Degree of neighborhood pixels
Selection of N eighbor Pixels
Choose block using threshold value
Weight Matrix Operation
Create matrix from
Weighted Matrix
Entry-wise Multiplication
SUM =677
Sum of elements
Extracted Data = Mod8 (677) = 5 =101
Extracted data
Secret Data =101=5
Compare with the dual block to choose the extraction position and repeat the same operation.
Secret Data can be generated by pre-pending the extracted bits in backward traversing

25	33	82	101
31	29	78	15
60	66	24	121
98	52	89	12

25	33	82	101
32	29	78	15
60	66	24	121
98	52	89	12

1	2	3	4
2	1	3	1
2	3	1	4
4	2	3	1

Pixel Range	Category
0 - 31	1
32 - 63	2
64 - 95	3
96 - 127	4
128 - 159	5
160 - 191	6
192 - 223	7
224 - 255	8

1	1	1	0
2	2	2	1
2	2	3	0
0	1	1	1

2	2	2	
2	2	3	

31	29	78
60	66	24

1	2	3
4	1	2

31	58	234
240	66	48

32	29	78
60	66	24

Figure 4.2: *Illustration of hidden data retrieval and cover image recovery in DSGN*

existing RDH schemes. Though the embedding capacity of the DSGN scheme is lower than some other RDH scheme, the PSNR of the DSGN scheme is very high with respect to other existing schemes. The PSNR is near 54.76 dB for Lena image. It has been observed that in both cases, the proposed DSGN scheme achieves a better result than other existing schemes. In terms of PSNR, the DSGN scheme is almost 18 dB and 8 dB greater than Lu [108] and Yang et al. [107] schemes, respectively. The payload of the DSGN scheme is 2.22 bpp.

4.1.3.2 Robustness Analysis

The dual stego images of DSGN scheme have been evaluated through some standard analysis and attacks like Standard Deviation (SD), Correlation Coefficient(CC) and Brute force attack. The image parameters like SD (σ) and CC, (ρ) are used to check the statistical distortion in the stego image after embedding a large amount of secret information. The σ before and after data embedding and ρ of the original and stego image are summarized in Table 4.1. To avoid statistical attacks, keeping the parameter differences minimum is very important. From Table 4.1, it is noticed that there is no significant difference between the SD of the dual cover and the dual stego image. The σ of the original image is 128.6353 and that of dual stego images are

Algorithm 4.2: DSGN: Extraction Algorithm

```
input  : A Stego Image of size $m \times$n, Weighted Matrix, Shared Secret Key
output: Extracted Secret Message

Algorithm Extraction():
    Step-1: Get 512 bit binary sequence of the shared secret key using SHA-512
    keyBits[] = SHA-512(Shared Secret Key);
    bitIndex = 0;
    for (pixelBlock ← 1 to $(m \times n)/4$ ) do
        Step-2: A matrix $SM_{x \times y}$ is formed using the selected pixels from the $4 \times 4$ pixelBlock of either first dual image or second dual image depending on the
          key bit is 0 or 1
        if ( keyBits[bitIndex] == 0) then
            // Create the pixel matrix from first dual image
            CreatePixelMatrix();
        else
            // Create the pixel matrix from second dual image
            CreatePixelMatrix();
        bitIndex++;
        Step-3: The secret data is extracted from the selected pixels
        ExtractSecretData();
    Step-4: The secret image is generated from the collected secret bits

Function CreatePixelMatrix():
    Step-1: Get neighbourhood degree of all pixels in $4 \times 4$ block of the cover image
    Step-2: A threshold value is considered here
    Step-3: If the neighbourhood degree of a pixel is more than or equals to the threshold value, add the pixel in the $SM_{x \times y}$ matrix

Function ExtractSecretData():
    Step-1: An weighted matrix ($WM_{x \times y}$) having the same dimension of $SM_{x \times y}$ matrix is formed
    Step-2: An elementary multiplication of $SM_{x \times y}$ with $WM_{x \times y}$ is performed
    Step-3: The sum of elements ($SUM$) of the resultant matrix is calculated.
    for $i$ ← 1 to x do
        for $j$ ← 1 to y do
            SUM $+= SM[i][j] \times WM[i][j]$
    Step-4: The modulo of $SUM$ is calculated as $D = MOD(SUM, 2^r)$
    Step-5: This $D$ is actually the extracted secret data. It is converted to $r$ bit binary string
```

128.7123 and 128.7019. Their differences are 0.0770 and 0.0666 respectively for Lena image after embedding $777,965$ secret bits. The ρ between the original image and the stego images is 0.9991 for Lena image. As the data is embedded distributively among the dual images, it is hard to find the position of embedding in the stego image. The image parameters remain almost unchanged in the DSGN scheme. This reduces the probability of detecting secret data and thus make it a secure steganographic scheme. The DSGN scheme has been tested by RS analysis [41] using the equation 1.5 to measure the robustness. From Table 4.4, it is observed that the values of R_M and R_{-M}, S_M and S_{-M} are almost same for stego image. This implies that the average RS value is nearly equal to zero in the DSGN scheme. So, the proposed DSGN scheme is secured against RS attack. The proposed DSGN scheme protects digital multimedia documents by embedding a secret image (logo) using a weighted matrix. Here, the values of the secret bits are stored within dual stego images. Instead of storing the original secret bits, the pixel value of a particular pixel is changed to represent the secret bits. The scheme is secured to prevent possible malicious attacks. The secret image can not be recovered from the stego

Table 4.1: *Experimental results after embedding maximum amount of secret information in DSGN*

Threshold Value	Images	Capacity (in bit)	PSNR (in dB)		SD			CC		SSIM	
			CD1 vs SI1	CD2 vs SI2	CD1	SI1	CD2	CD1 vs SI1	CD2 vs SI2	CD1 vs SI1	CD2 vs SI2
1	Lenna	777965	54.3920	54.7635	128.6353	128.7123	128.7019	0.9991	0.9991	0.9987	0.9968
	Airplane	780457	54.2756	54.3875	142.6353	142.6387	142.6429	0.9992	0.9991	0.9953	0.9955
	Boat	772988	54.2090	54.7739	135.9962	135.5374	135.0092	0.9991	0.9992	0.9928	0.9929
	Baboon	760836	54.2465	54.3872	125.5678	125.3654	126.6354	0.9992	0.9993	0.9960	0.9962
	Peppers	778013	54.2998	54.8892	135.9671	135.9671	135.4463	0.9991	0.9992	0.9961	0.9964
2	Lenna	742598	54.6354	54.3564	128.6353	128.4235	128.4635	0.9992	0.9993	0.9965	0.9967
	Airplane	754238	54.6475	54.6123	142.6353	142.4736	142.4653	0.9993	0.9992	0.9966	0.9967
	Boat	711640	54.8576	54.5867	135.9962	135.7645	135.7462	0.9992	0.9993	0.9965	0.9966
	Baboon	648246	54.6387	54.3786	125.5678	125.3452	125.3876	0.9993	0.9994	0.9968	0.9969
	Peppers	740201	54.4873	54.4482	135.9671	135.7448	135.7439	0.9992	0.9993	0.9968	0.9969
3	Lenna	678760	54.7762	54.8922	128.6353	128.4873	128.4946	0.9993	0.9994	0.9969	0.9971
	Airplane	706011	54.2286	54.4234	142.6353	142.4837	142.4424	0.9994	0.9993	0.9972	0.9971
	Boat	614733	54.2987	54.2789	135.9962	135.7925	135.7736	0.9994	0.9994	0.9972	0.9973
	Baboon	485322	55.8732	55.7238	125.5678	125.3654	126.3276	0.9994	0.9993	0.9974	0.9973
	Peppers	678381	54.9547	54.2398	135.9671	135.7635	135.7291	0.9993	0.9994	0.9973	0.9972
4	Lenna	547387	54.4856	54.7334	128.6353	128.4337	128.4326	0.9994	0.9995	0.9981	0.9980
	Airplane	584402	54.2448	54.2763	142.6353	142.4238	142.4869	0.9995	0.9995	0.9982	0.9981
	Boat	471350	55.7523	55.6589	135.9962	135.7568	135.7652	0.9996	0.9996	0.9982	0.9980
	Baboon	320622	57.2905	57.3549	125.5678	125.3276	126.3827	0.9995	0.9994	0.9984	0.9984
	Peppers	550590	54.6647	54.7019	135.9671	135.7654	135.7345	0.9995	0.9995	0.9982	0.9982
5	Lenna	411864	55.8879	55.7465	128.6353	128.4476	128.4374	0.9995	0.9996	0.9983	0.9984
	Airplane	461269	53.6534	53.4378	142.6353	142.4356	142.4523	0.9996	0.9996	0.9986	0.9986
	Boat	335999	56.8943	56.4398	135.9962	135.4387	135.4442	0.9997	0.9997	0.9984	0.9984
	Baboon	193552	58.4324	58.1244	125.5678	125.3435	126.3362	0.9996	0.9995	0.9985	0.9985
	Peppers	419480	55.2484	55.5438	135.9671	135.7645	135.7226	0.9997	0.9996	0.9986	0.9986
6	Lenna	248639	58.7093	58.0039	128.6353	128.4942	128.4836	0.9996	0.9997	0.9989	0.9989
	Airplane	289907	57.6994	57.5436	142.6353	142.4387	142.4429	0.9997	0.9998	0.9989	0.9989
	Boat	194866	58.3726	58.4123	135.9962	135.3374	135.3192	0.9998	0.9998	0.9989	0.9988
	Baboon	099335	62.7633	62.6999	125.5678	125.3954	126.3254	0.9997	0.9997	0.9988	0.9987
	Peppers	255369	57.2563	57.2543	135.9671	135.7671	135.7763	0.9998	0.9997	0.9988	0.9988
7	Lenna	120474	60.7365	60.5436	128.6353	128.5423	128.5487	0.9997	0.9998	0.9994	0.9995
	Airplane	141291	60.8335	60.3556	142.6353	142.5746	142.5498	0.9998	0.9998	0.9992	0.9993
	Boat	093319	61.3762	61.5436	135.9962	135.8735	135.8809	0.9999	0.9998	0.9998	0.9997
	Baboon	043501	64.4837	64.9361	125.5678	125.4536	126.4435	0.9998	0.9998	0.9996	0.9995
	Peppers	125657	60.3627	60.5468	135.9671	135.8719	135.8653	0.9999	0.9998	0.9998	0.9997
8	Lenna	094017	61.5882	61.7673	128.6353	128.6032	128.6139	0.9999	0.9998	0.9998	0.9999
	Airplane	125437	61.3101	61.4387	142.6353	142.6019	142.6247	0.9999	0.9998	0.9999	0.9999
	Boat	066855	62.0097	62.1203	135.9962	135.9901	135.9843	0.9999	0.9999	0.9997	0.9999
	Baboon	025520	68.6354	68.6211	125.5678	125.5231	126.5129	0.9998	0.9999	0.9999	0.9998
	Peppers	097995	62.8937	62.7342	135.9671	135.9012	135.9182	0.9999	0.9999	0.9998	0.9999

image without knowing the correct weighted matrix and shared secret key. For example, Fig. 4.3 shows the original image, logo image, attack with unknown weighted matrix, and shared secret key. The result indicates that the attacker only acquires noise like images when applying wrong weighted matrix and/or secret key to reveal the secret logo image. In this context, it can be mentioned that the proposed steganographic scheme is highly robust, and only an authorized person can extract the secret image from stego images. A brute force attack may be tried by an attacker to reveal the secret message. In a $(m \times n)$ image there are $(2^{r-1}+1)! \times (m/3 \times n/3)$

Table 4.2: *Comparison of proposed scheme with existing dual image based schemes in terms of PSNR(dB) and payload(bpp) in DSGN*

Image (512 x 512)	**Comparison Metric**		**Chang et al. [100]**	**Chang et al. [101]**	**Qin et al. [102]**	**DSGN (Color Image)**	**DSGN (Grayscale Image)**
Lena	PSNR	Dual 1	45.19	45.21	45.21	54.3920	46.5391
		Dual 2	45.2	45.21	45.21	54.7635	54.4859
	Payload		524288	802895	557052	777,965	259,321
Baboon	PSNR	Dual 1	45.18	39.91	52.04	54.2465	53.9931
		Dual 2	45.19	39.91	41.56	54.3872	53.7892
	Payload		524288	802524	557096	760,836	253,612
Pepper	PSNR	Dual 1	45.21	39.94	51.25	54.2998	53.9283
		Dual 2	45.21	39.94	41.52	54.8892	54.3957
	Payload		523356	799684	557245	778,013	259,337
Barbara	PSNR	Dual 1	45.20	39.89	52.12	54.3928	53.9992
		Dual 2	45.21	39.89	41.58	54.2287	54.1091
	Payload		524288	802888	557339	789,348	263,116
Zelda	PSNR	Dual 1	45.66	39.88	51.72	54.8954	53.0818
		Dual 2	45.2	39.88	41.75	54.7681	53.6848
	Payload		524288	802789	557264	778,982	259,660

Table 4.3: *Comparison of proposed scheme with existing reversible data hiding schemes in terms of average PSNR (dB) and payload (bpp) in DSGN*

Scheme	Lena (512 × 512)		Baboon (512 × 512)		Pepper (512 × 512)		Barbara (512 × 512)	
	Secret Image(bits)	PSNR(dB)	Secret Image(bits)	PSNR(dB)	Secret Image(bits)	PSNR(dB)	Secret Image(bits)	PSNR(dB)
Hwang et al. [103]	5336	48.22	5208	48.20	15300	48.40	7501	48.25
Hu et al. [104]	60241	48.69	21411	48.34	77254	48.86	28259	48.40
Luo et al. [105]	71674	48.82	22696	48.36	84050	48.94	38734	48.50
Abadi et al. [106]	73207	48.78	45043	48.52	86964	48.92	43420	48.51
Jung et al. [23]	155598	41.87	324795	38.10	152448	41.49	209222	38.98
Yang et al. [107]	155598	45.77	324795	42.03	152448	44.41	209222	43.30
Jana B. [99]	776224	35.80	776224	36.12	776224	35.78	776224	35.42
Lu [108]	786732	36.03	786732	35.99	786732	36.3	786732	36.03
Kuo et al. [109]	851958	37.24	851958	37.25	851958	37.25	851958	37.25
DSGN (Grayscale Image)	259321	54.4859	253612	53.7892	259337	54.3957	263116	54.1091
DSGN (Color Image)	777965	54.7635	760836	54.3872	778013	54.8892	789348	54.2287

Table 4.4: *Results of RS analysis in DSGN*

Cover Image (C)	**Secret Data (Bits) (S)**	**Stego Images**										
		R_M		R_{-M}		S_M		S_{-M}		**RS Value**		
		SI1	**SI2**	**SI1**	**SI2**	**SI1**	**SI2**	**SI1**	**SI2**	**SI1**	**SI2**	**Average**
Lena	777,965	24827	24939	24492	24872	14295	14872	12898	12441	0.0461	0.0597	0.0529
Baboon	760,836	3869	3969	4096	4101	4128	4097	4078	4132	0.0606	0.0678	0.0642
Barbara	789,348	21457	21979	24863	24536	19104	19234	16276	16241	0.1537	0.1069	0.1303
Zelda	778,982	19450	19664	24129	24243	15737	15678	11990	11921	0.0935	0.1087	0.1011
Pepper	778,013	21215	21289	24013	24122	19907	19899	17305	17379	0.1313	0.0948	0.1130

possible ways to reveal the hidden message, where r denotes the number of secret bits embedded in each pixel block. This method enhances the security of the scheme. Again, the distribution of

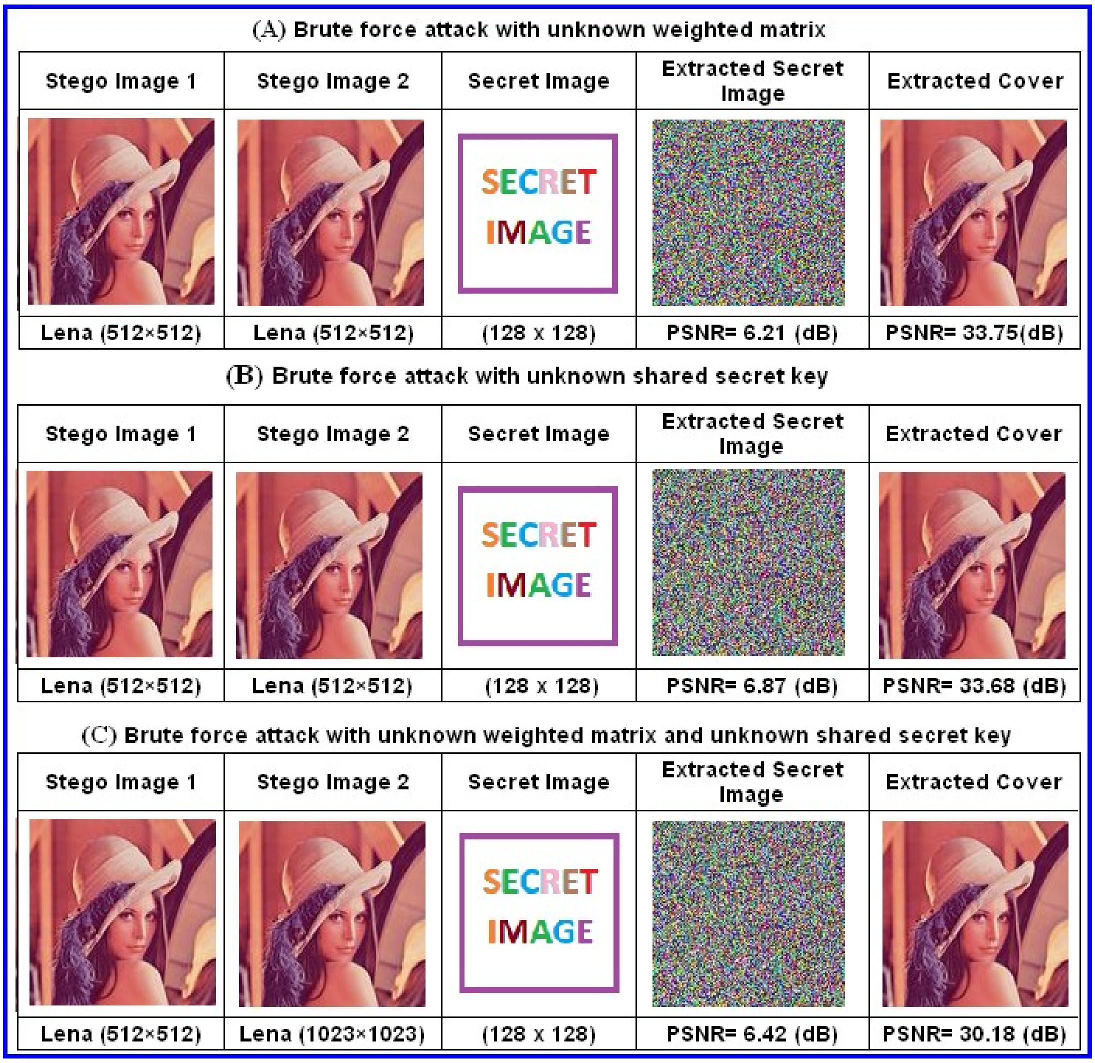


Figure 4.3: *Experimental results of brute force attack with unknown weighted matrix and shared secret key in DSGN*

secret bits between the two stego images has been performed depending on 0 or 1 value of the shared secret key bit. The number of possible weighted matrix will be $(2^{r-1}+1)! \times (m/3 \times n/3)$ and only one of them is required to recover secret message. So, the robustness of the scheme against brute force attack is very high due to this huge number of attempts. The proposed DSGN scheme shows higher robustness against various other attacks also. Moreover, the logo image and the cover image can be fully retrieved from the stego images using actual weighted matrix and shared secret key.

4.2 DS Using Weighted Matrix (DSWM)

Dual image based steganographic scheme is a new aspect in the field of multimedia security to enhance embedding capacity, improve visual quality and security. Few steganographic schemes have been developed employing interpolated gray-scale image but they suffer from limited hiding capacity with low visual quality. Here, a steganographic scheme has been developed by taking the help of weighted matrix, where more data bits can be embedded through only one bit modification in the original pixel. Here, the secret message and embedding location are stored alternately on the dual images. The dual images are partitioned into (3×3) pixel blocks and the first pixel block of the first image contains the secret message whereas the first pixel block of the second image carries the modification location of the corresponding pixel block of the first image. Similarly, in the next iteration, the second pixel block of the second image contains the secret message and the second pixel block of the first image carries the location of alteration of the corresponding pixel block of the second image. In this way, the secret message and the data embedding locations are stored within two images alternately. Distributing the valuable secret information in two images and updating the weighted matrix with the shared key before each iteration enhances the security of the scheme. The experimental results show the effectiveness of the proposed DSWM scheme, which provides visual quality more than 47 dB PSNR after embedding $1,754,460$ bits secret information.

4.2.1 Data Embedding Procedure

In this approach, a dual image based reversible steganographic scheme has been proposed using a weighted matrix for a color image. At first, a color image $(C_{m\times n})$ is considered as cover image. Two interpolated color images are created from the original cover image $(C_{m\times n})$ by equation 4.1.

$$\begin{cases} CI_{min} = \min C(i,j), C(i+2,j), C(i,j+2), C(i+2,j+2) \\ CI_{max} = \max C(i,j), C(i+2,j), C(i,j+2), C(i+2,j+2) \\ AD = \frac{3\times CI_{min}+2\times CI_{max}}{5} \\ C(i,j) = CI(i,j) \\ C(i,j+1) = \frac{AD+(C(i,j)+C(i,j+2))/2}{2} \\ C(i+1,j) = \frac{AD+(C(i,j)+C(i+2,j))/2}{2} \\ C(i+1,j+1) = \frac{(C(i,j)+C(i+1,j)+C(i,j+1)}{3} \end{cases} , \tag{4.1}$$

where $i = 2m, j = 2n$, and $m, n = 0, 1, 2, ..., k$. Here, m and n are considered as the row and column of the cover image. The interpolated image (CI) is generated from original cover image (C).

The dual images $CD1_{2m-1\times 2n-1}$ and $CD2_{2m-1\times 2n-1}$ are copied from $CI_{2m-1\times 2n-1}$ and decomposed into (3×3) pixel blocks. Thereafter, Red, Green, and Blue color components are separated and considered as three separate matrices $P_{-RED_{ij}}$, $P_{-GREEN_{ij}}$ and $P_{-BLUE_{ij}}$, where $i, j = 0, 1, 2$. Now, 4 bits secret data are taken from the secret message M. The data bits are embedded within $P_{-RED_{ij}}$ pixel block of SI1 image using the Algorithm 4.3 and the embedding location stored the $P_{-RED_{ij}}$ pixel block of SI2 image. This step is repeated 5 times to store data bits within the same pixel block. Likewise, the secret information is also embedded in the $P_{-GREEN_{ij}}$ and $P_{-BLUE_{ij}}$ pixel blocks. The odd pixel blocks of SI1 image contain secret data bits, and odd pixel blocks of SI2 image keeps the embedding location of SI1 whereas the even pixel blocks of SI1 contain embedding location of SI2, and even pixel blocks of SI2 contain secret data bits. Thus, the secret data, as well as the embedding locations are distributed among dual images, CD1 and CD2. After performing Algorithm 4.3, dual stego images are generated. The detail schematic diagram of the data embedding process within R, G, B, color components through the modified weighted matrix is shown in Fig. 4.4.

A weighted matrix WM of size (3×3) is shared by sender and receiver before data communication. Each element in the matrix is actually the value from the set $(1, 2, \ldots, 2^{r-1})$ and each element from this set appears at least once in the weight matrix WM. Then, r data bits, say $b_1b_2 \ldots b_r$, are embedded into image block $P_{-RED_{ij}}$ by $d = (b_1b_2 \ldots b_r)_2 - SUM(P_{-RED_{ij}} \otimes W)(\bmod 2^r)$, where i,j= 0,1,2 and $\otimes$ is the entry-wise multiplication operator. If the value of d is zero, $P_{-RED_{ij}}$ remains unchanged; otherwise, modify $P_{-RED_{ij}}$ to $P^{'}_{-RED_{ij}}$ to satisfy: $SUM(P^{'}_{-RED_{ij}} \otimes WM) = b_1b_2...b_r(\bmod 2^r)$ where the function $SUM()$ represents the modular sum of all the entries of the matrix $(P_{-RED_{ij}} \otimes W)$. The modification from $P_{-RED_{ij}}$ to $P^{'}_{-RED_{ij}}$ is done by adding or subtracting the value at d^{th} position of the weighted matrix. Addition or subtraction depends on the value of the $sign$ is positive or negative, respectively. Here, the $sign$ is considered as a positive or negative unary operator. For example, if 5th position of the $P_{-RED_{ij}}$ is 20 and $d = +5$, then $P^{'}_{-RED_{ij}}$ will be $20 + 1 = 21$. Otherwise, if $d = -5$, $P^{'}_{-RED_{ij}}$ will be $20 - 1 = 19$.

The receiver can deduce $b_1b_2 \ldots b_r$ by computing $SUM(P^{'}_{-RED_{ij}} \otimes WM)\ (mod\, 2^r)$. In this

DSWM scheme, the above operation is performed separately for R, G, B color blocks. That means $P_{-GREEN_{ij}}$ and $P_{-BLUE_{ij}}$ are used in the same manner. Finally, three modified blocks $P^{'}_{-RED_{ij}}$, $P^{'}_{-GREEN_{ij}}$ and $P^{'}_{-BLUE_{ij}}$ are combined to get modified original image block $C^{'}_{i}$, where $i = 0, 1, 2, \ldots$ number of blocks. Before each entry-wise multiplication operation, WM is updated using $WM_{i+1} = (WM_i \times \delta)\ mod\ 9$ where $i = 0, 1, 2, \ldots, 2^r$ and $gcd\ (\delta, 9) = 1$. The original weighted matrix WM and δ are shared between the sender and the receiver during data communication. The value of the matrix at d^{th} position of the weight matrix is either increased or decreased during the embedding phase. During extraction, only the value of $SUM(P^{'}_{-RED_{ij}} \otimes W)(\text{mod } 2^r)$ is required.

The embedding procedure of DSWM scheme is presented in Algorithm 4.3 and explained step by step using two procedures: $EmbedDataBitsInPixelBlock()$ and $Embedding - Main()$.

EmbedDataBitsInPixelBlock():

Step 1: The summation of entry-wise multiplication of pixel block P with weighted matrix W is stored in $sumOfMatrixElements$.

Step 2: The decimal equivalent of 4 bits secret message is stored in $intNBits$.

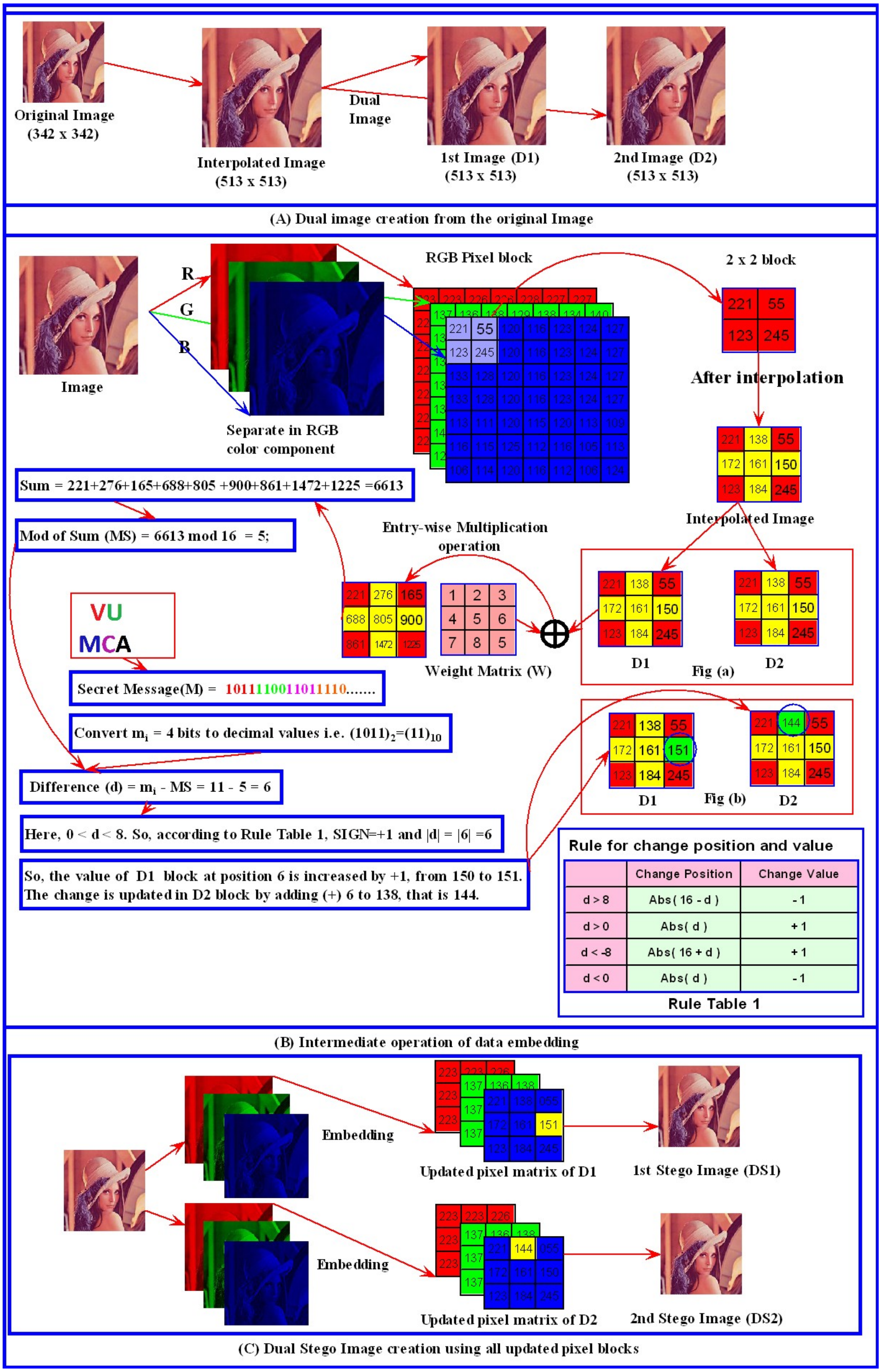
Step 3: The difference d is calculated by deducting SUM from $intNBits$. The value of d and $sign$ is assigned accordingly. $absD$ stores the absolute value of d.

Step 4: $posX$ and $posY$ store the row and column index of the item $absD$, present in the weight matrix WM.

Step 5: The $integerXY$ stores the integer position of $absD$ in the Weight Matrix starting from 1 to 9.

Step 6: The value of the pixel in the position $posX$, $posY$ of the pixel block of the first image SI1 is incremented or decremented by 1 in accordance with the value of $sign$ that is, $+ve$ or $-ve$ respectively.

Step 7: At the same time, to store the position of the change in the first image, the value of the interpolated pixel of the second image SI2 has to be updated by adding the value of $integerXY$ multiplied by $sign$.



221	55
123	245

221	138	55
172	161	150
123	184	245

221	276	165
688	805	900
861	1472	1225

1	2	3
4	5	6
7	8	5

Sum = 221+276+165+688+805 +900+861+1472+1225 =6613

Mod of Sum (MS) = 6613 mod 16 = 5;

Secret Message(M) = 1011110011011110.......

Convert m_i = 4 bits to decimal values i.e. $(1011)_2=(11)_{10}$

Difference (d) = m_i - MS = 11 - 5 = 6

Here, 0 < d < 8. So, according to Rule Table 1, SIGN=+1 and |d| = |6| =6

So, the value of D1 block at position 6 is increased by +1, from 150 to 151. The change is updated in D2 block by adding (+) 6 to 138, that is 144.

Fig (b) D1:

221	138	55
172	161	151
123	184	245

Fig (b) D2:

221	144	55
172	161	150
123	184	245

Rule for change position and value

	Change Position	Change Value
d > 8	Abs(16 - d)	- 1
d > 0	Abs(d)	+ 1
d < -8	Abs(16 + d)	+ 1
d < 0	Abs(d)	- 1

Rule Table 1

Figure 4.4: *Schematic diagram of data embedding process in DSWM*

Algorithm 4.3: DSWM: Embedding Algorithm

```
input  : Cover Image $C_{(m \times n)}$, Secret Image $M = m_1 m_2 m_3 m_4 \ldots Length(M/4)$, Where $m_i = 4$ bits secret message
output: Stego Images SI1 and SI2

Algorithm Embedding-Main():
    Create interpolated dual images CD1 & CD2 from cover image C;
    // Loop to embed data into all pixel blocks
    for $(y = 0; y < n/3; y++)$ do
        for $(int\ x = 0; x < m/3; x++)$ do
            // If the block number is odd, embed data in CD1 and store changed location information in CD2
            if $(blockNumber\ is\ Odd)$ then
                for $(r = 0; r < 5; r++)$ do
                    // Embed data in RED component
                    EmbedDataBitsInPixelBlock($P_{-RED}, r, x, y, CD1, CD2$);
                for $(r = 0; r < 5; r++)$ do
                    // Embed data in GREEN component
                    EmbedDataBitsInPixelBlock($P_{-GREEN}, r, x, y, CD1, CD2$);
                for $(r = 0; r < 5; r++)$ do
                    // Embed data in BLUE component
                    EmbedDataBitsInPixelBlock($P_{-BLUE}, r, x, y, CD1, CD2$);
            // If the block number is even, embed data in CD2 and store changed location information in CD1
            else if $(blockNumber\ is\ Even)$ then
                for $(r = 0; r < 5; r++)$ do
                    // Embed data in RED component
                    EmbedDataBitsInPixelBlock($P_{-RED}, r, x, y, CD2, CD1$);
                for $(r = 0; r < 5; r++)$ do
                    // Embed data in GREEN component
                    EmbedDataBitsInPixelBlock($P_{-GREEN}, r, x, y, CD2, CD1$);
                for $(r = 0; r < 5; r++)$ do
                    // Embed data in BLUE component
                    EmbedDataBitsInPixelBlock($P_{-BLUE}, r, x, y, CD2, CD1$);
    Create first stego image SI1 from CD1;
    Create second stego image SI2 from CD2;

Function EmbedDataBitsInPixelBlock($P_{Color}$, rotation, x, y, dataImage, infoImage):
    Secret4BitMessage ← Get 4 bits from secret image;
    MultipliedMatrix ← Multiply weight matrix $dataImage$ and pixel matrix $P_{Color}$ ;
    sumOfMatrixElements ← Get sum of $MultipliedMatrix$ matrix elements ;
    SUM ← Mod(sumOfMatrixElements, 16);
    int4BitMessage ← Convert binary $Secret4BitMessage$ to integer ;
    d ← (intNBits - SUM); int modD=0; int sign=0;
    if $(d > 8)$ then d ← 16-d;  sign ← -1 ;
    else if $(d > 0)$ then sign ← +1 ;
    else if $(d < -8)$ then d ← 16+d;  sign ← +1 ;
    else if $(d < 0)$ then sign ← -1 ;
    absD ← Absolute value of $d$ ;
    posX ← X position of the absD value in the Weight Matrix;
    posY ← Y position of the absD value in the Weight Matrix;
    integerXY ← integer position of the value in the Weight Matrix starting from 1 to 9;
    Update weighted matrix by $WM_{i+1} \leftarrow (WM_i \times \delta)\ mod\ 9,$ where $i = 0, 1, 2, \ldots, 2^r$ and $gcd\ (\delta, 9) = 1$;
    // Change the data pixel block
    dataImage$[3 * (y/3) + posY][3 * (x/3) + posX][P_{Color}]$ ← dataImage$[3 * (y/3) + posY][3 * (x/3) + posX][P_{Color}]$ + sign;
    // Change the info pixel block
    if (rotation==0) then infoImage$[3 * (y/3) + 0][3 * (x/3) + 1][P_{Color}] \leftarrow$
      infoImage$[3 * (y/3) + 0][3 * (x/3) + 1][P_{Color}] + (sign * integerXY)$ ;
    else if (rotation ==1) then infoImage$[3 * (y/3) + 1][3 * (x/3) + 0][P_{Color}] \leftarrow$
      infoImage$[3 * (y/3) + 1][3 * (x/3) + 0][P_{Color}] + (sign * integerXY)$ ;
    else if (rotation ==2) then infoImage $[3 * (y/3) + 1][3 * (x/3) + 1][P_{Color}]$ ← infoImage
      $[3 * (y/3) + 1][3 * (x/3) + 1][P_{Color}] + (sign * integerXY)$ ;
    else if (rotation ==3) then infoImage $[3 * (y/3) + 1][3 * (x/3) + 2][P_{Color}]$ ← infoImage
      $[3 * (y/3) + 1][3 * (x/3) + 2][P_{Color}] + (sign * integerXY)$ ;
    else if (rotation ==4) then infoImage $[3 * (y/3) + 2][3 * (x/3) + 1][P_{Color}]$ ← infoImage
      $[3 * (y/3) + 2][3 * (x/3) + 1][P_{Color}] + (sign * integerXY)$ ;
```

Embedding-Main():

Step 1: The function $EmbedDataBitsInPixelBlock()$ is called 5 times for each pixel block. Total 20 bits per iteration are embedded in a single color component. So, for three different color pixel blocks, a total of 60 bits of secret data are embedded in a (3×3) color pixel block.

Step 2: For the first iteration, the first pixel block of SI1 is used for secret data embedding whereas the first pixel block of SI2 is used to store changed location information. In the next iteration, the second pixel block of SI2 is used for secret data embedding, whereas the second pixel block of SI1 is used to store changed location information and so on. In this way, the pixel blocks of alternative stego images are used for data embedding.

Numerical Example of Embedding Procedure:

The embedding process is presented step by step with an example here:

Step 1: A 3×3 pixel block (CD1) and a weighted matrix (WM) is taken as below:

221	138	055
172	161	150
123	184	245

CD1

1	2	3
4	5	6
7	8	5

WM

Step 2: Another pixel block (CD2) is created from the pixel block (CD1).

Step 3: An elementary multiplication is done between the pixel block (CD1) and the weight matrix (WM) to get the matrix below:

221	276	165
688	805	900
861	1472	1225

Step 4: Sum of elements of this matrix is calculated as $221 + 276 + 165 + 688 + 805 + 900 + 861 + 1472 + 1225 = 6613$

Step 5: Modulo 16 of the sum is then calculated as $MS = 6613\ Mod\ 16 = 5$.

Step 6: 4-bit binary secret data is taken as $(1011)_2$

Step 7: The decimal equivalent of this data is then calculated as $DE = (11)_{10}$

Step 8: The difference (d) is calculated by subtracting modulo sum from this decimal data. $d = DE - MS = 11 - 5 = 6$.

Step 9: Here $d > 0$. According to Rule Table 1 (Fig. 4.4), the position of change is the absolute value of d i.e., 6 and change value is $+1$. So, in the CD1 pixel matrix, at position 6, the pixel value will be increased by 1 (i.e. $150\ to\ 151$). This position (i.e., 6) is stored in the CD2 pixel matrix by adding 6 to the pixel value in the first interpolated pixel position (i.e., $138\ to\ 144$). The updated stego pixel blocks SI1 and SI2 from CD1 and CD2 are shown below:

221	138	055
172	161	151
123	184	245

SI1

221	144	055
172	161	150
123	184	245

SI2

4.2.2 Data Extraction Procedure

During extraction and cover image reconstruction, dual stego images $SI1$ and $SI2$ are considered. The $SI1$ and $SI2$ are decomposed into (3×3) pixel blocks and separated into RED, GREEN and BLUE color component blocks $P_{-RED_{ij}}$, $P_{-GREEN_{ij}}$, and $P_{-BLUE_{ij}}$ respectively. The pixel blocks of the dual stego images where the data is embedded can be considered as a data pixel block, and the pixel blocks where the embedding position is stored can be considered as info pixel block. In the proposed DSWM scheme, the odd pixel blocks of SI1 and the even pixel blocks of SI2 are data pixel blocks whereas the odd pixel blocks of SI2 and the even pixel blocks of SI1 are info pixel blocks. Now data extraction is started by entry-wise multiplication with weighted matrix WM and a data pixel block. After extracting data from the data pixel block, it is compared with its corresponding info pixel block. After comparison, the pixel value of the data pixel block at a particular position is increased or decreased by one to make it suitable for the next iteration. In this way, all data are extracted from a data pixel block after five iterations. The extraction process continues for each block, taking into consideration the $P_{-BLUE_{ij}}$, $P_{-GREEN_{ij}}$, and $P_{-RED_{ij}}$ color components. The data bits are extracted from the RED data pixel block of SI1 using the information stored in the info pixel block of SI2. This step is repeated for five times to extract data bits from the same data pixel block. The data

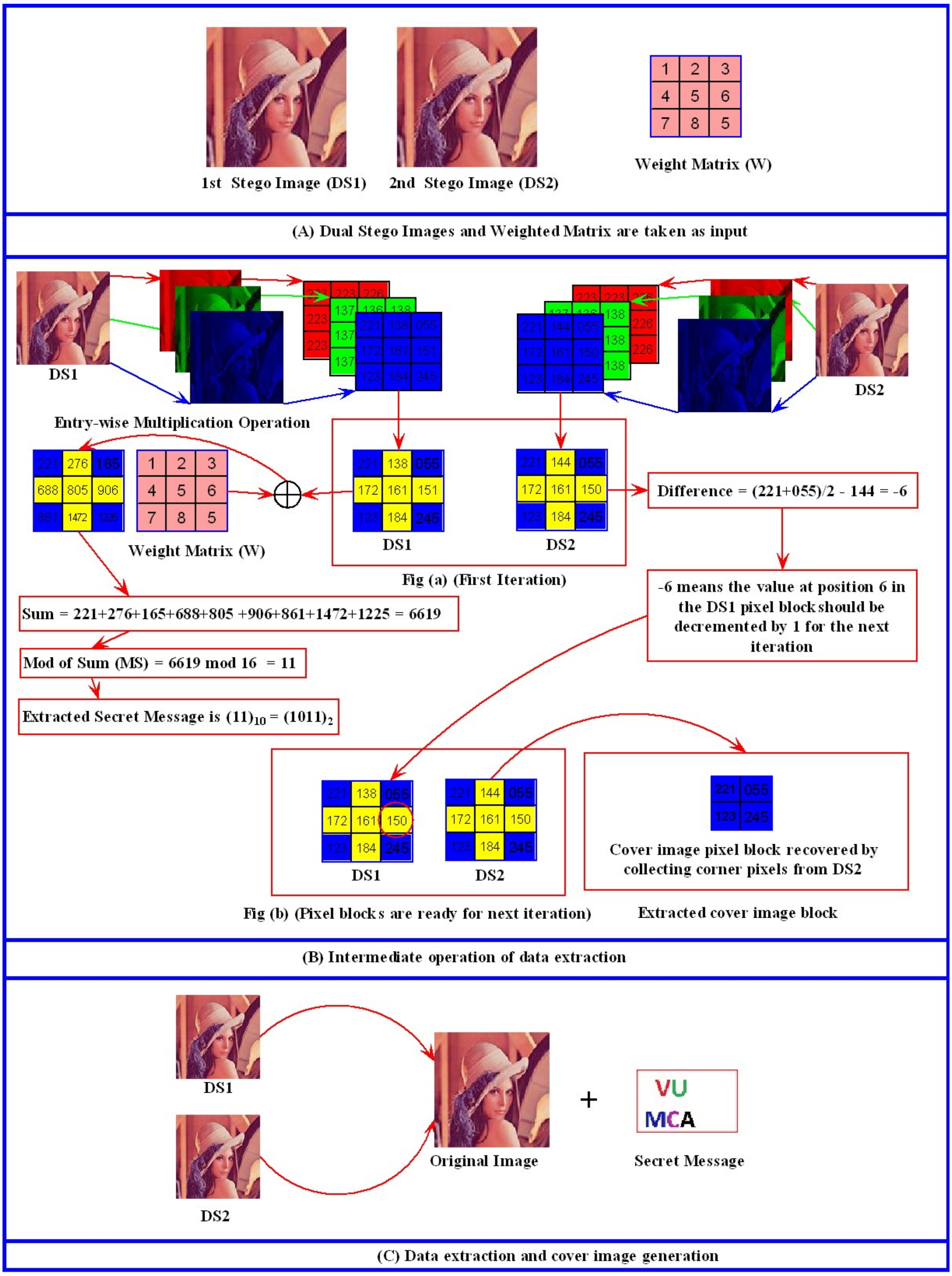

1st Stego Image (DS1)
2nd Stego Image (DS2)
Weight Matrix (W)
(A) Dual Stego Images and Weighted Matrix are taken as input
DS1
DS2
Entry-wise Multiplication Operation
221 276 165
688 805 906
861 1472 1225
1 2 3
4 5 6
7 8 5
Weight Matrix (W)
221 138 055
172 161 151
123 184 245
DS1
221 144 055
172 161 150
123 184 245
DS2
Fig (a) (First Iteration)
Difference = (221+055)/2 - 144 = -6
-6 means the value at position 6 in the DS1 pixel block should be decremented by 1 for the next iteration
Sum = 221+276+165+688+805 +906+861+1472+1225 = 6619
Mod of Sum (MS) = 6619 mod 16 = 11
Extracted Secret Message is $(11)_{10} = (1011)_2$
221 138 055
172 161 150
123 184 245
DS1
221 144 055
172 161 150
123 184 245
DS2
Fig (b) (Pixel blocks are ready for next iteration)
221 055
123 245
Cover image pixel block recovered by collecting corner pixels from DS2
Extracted cover image block
(B) Intermediate operation of data extraction
DS1
DS2
Original Image
+
VU
MCA
Secret Message
(C) Data extraction and cover image generation

Figure 4.5: *Schematic diagram of data extraction process in DSWM*

are extracted from the GREEN and BLUE data pixel blocks also in the same way, and thus the secret bits from an RGB data pixel block are retrieved. After extracting secret bits from all the data pixel blocks of SI1 and SI2, the secret information is formed. During embedding, the cor-

Algorithm 4.4: DSWM: Extraction Algorithm

```
input  : Dual Stego Image $SI1_{(m \times n)}$ and $SI2_{(m \times n)}$, and $\delta$
output: Secret Image $S$; Original Cover Image $C_{(m \times n)}$

Algorithm Extraction-Main():
    for $(int x = 1; x <= m/3; x + +)$ do
        if $(blockNumber\ is\ Odd)$ then
            for $(r = 5; r >= 1; r + +)$ do
                // Extract data from RED component
                ExtractDataBitsFromPixelBlock($P_{-RED}$, $r$, $x$, $y$, $interpolatedImage1$, $interpolatedImage2$);
            for $(r = 5; r >= 1; r + +)$ do
                // Extract data from GREEN component
                ExtractDataBitsFromPixelBlock ($P_{-GREEN}$, $r$, $x$, $y$, $interpolatedImage1$, $interpolatedImage2$);
            for $(r = 5; r >= 1; r + +)$ do
                // Extract data from BLUE component
                ExtractDataBitsFromPixelBlock ($P_{-BLUE}$, $r$, $x$, $y$, $interpolatedImage1$, $interpolatedImage2$);
        else if $(blockNumber\ is\ Even)$ then
            for $(r = 5; r >= 1; r + +)$ do
                // Extract data from RED component
                ExtractDataBitsFromPixelBlock ($P_{-RED}$, $r$, $x$, $y$, $interpolatedImage2$, $interpolatedImage1$);
            for $(r = 5; r >= 1; r + +)$ do
                // Extract data from GREEN component
                ExtractDataBitsFromPixelBlock ($P_{-GREEN}$, $r$, $x$, $y$, $interpolatedImage2$, $interpolatedImage1$);
            for $(r = 5; r >= 1; r + +)$ do
                // Extract data from BLUE component
                ExtractDataBitsFromPixelBlock ($P_{-BLUE}$, $r$, $x$, $y$, $interpolatedImage2$, $interpolatedImage1$);
    Create secret image from SI1 and SI2;

Function ExtractDataBitsFromPixelBlock ($P_{Color}$, r, x, y, dataImage, infoImage):
    MultipliedMatrix ← Multiply weight matrix $dataImage$ and pixel matrix $P_{Color}$ ;
    sumOfMatrixElements ← Sum of $MultipliedMatrix$ matrix elements ;
    SUM ← Mod(sumOfMatrixElements, 16);
    extracted4Bits ← Convert $SUM$ to binary ;
    Append $extracted4Bits$ to $TotalExtractedBits$ ;
    if $(count == 1)$ then
        changedValue ← (infoImage[3*(y/3)+0][3*(x/3)+0][$P_Color$] + infoImage [3*(y/3)+0][3*(x/3)+2][$P_Color$])/2 - infoImage
          [3*(y/3)+0][3*(x/3)+1][$P_Color$];
    else if $(count == 2)$ then
        changedValue ← (infoImage[3*(y/3)+0][3*(x/3)+0][$P_Color$] + infoImage [3*(y/3)+2][3*(x/3)+0][$P_Color$])/2 -
          infoImage[3*(y/3)+1][3*(x/3)+0][$P_{Color}$];
    else if $(count == 3)$ then
        changedValue ← ((infoImage [3*(y/3)+0][3*(x/3)+0][$P_Color$] + infoImage [3*(y/3)+0][3*(x/3)+2][$P_Color$] + infoImage
          [3*(y/3)+2][3*(x/3)+0][$P_Color$] + infoImage [3*(y/3)+2][3*(x/3)+2][$P_Color$])/4) - infoImage[3*(y/3)+1][3*(x/3)+1][$P_Color$];
    else if $(count == 4)$ then
        changedValue ← ((infoImage [3*(y/3)+0][3*(x/3)+2][$P_Color$] + infoImage [3*(y/3)+2][3*(x/3)+2][$P_Color$])/2) - infoImage
          [3*(y/3)+1][3*(x/3)+2][$P_Color$];
    else if $(count == 5)$ then
        changedValue ← ((infoImage [3*(y/3)+2][3*(x/3)+0][$P_Color$] + infoImage [3*(y/3)+2][3*(x/3)+2][$P_Color$])/2) - infoImage
          [3*(y/3)+2][3*(x/3)+1][$P_Color$];
    changeValueMod ← Abs(changedValue);
    posX ← X position of the $changeValueMod$ number in the Weight Matrix;
    posY ← Y position of the $changeValueMod$ number in the Weight Matrix;
    if (changedValue<0) then
        dataImage[3*(y/3)+posY][3*(x/3)+posX][$P_{Color}$] ← dataImage[3*(y/3)+posY][3*(x/3)+posX][$P_{Color}$] - 1;
    else if (changedValue > 0) then
        dataImage[3*(y/3)+posY][3*(x/3)+posX][$P_{Color}$] ← dataImage[3*(y/3)+posY][3*(x/3)+posX][$P_{Color}$] + 1;
    Update weighted matrix by $W_{i+1} \leftarrow (W_i \times \delta)\ mod\ 9,$ where $i = 0, 1, 2, \ldots, 2^r$ and $gcd\ (\delta, 9) = 1$ ;
    // Original cover image recovery
    Extract corner pixels from $3 \times 3$ $dataImage$ matrix to form original $2 \times 2$ cover image pixel block ;
```

ner pixels of all the (3×3) info pixel blocks of SI1 and SI2 remain unchanged. So, extracting the four corner pixels from all the 3×3 info pixel blocks of SI1 and SI2 form the original cover image. The extraction procedure is presented in Algorithm 4.4. The block diagram is shown in Fig. 4.5.

The proposed extraction procedure, presented in Algorithm 4.4, is explained step by step using two procedures: $ExtractDataBitsFromPixelBlock()$ and $Extraction - Main()$.

ExtractDataBitsFromPixelBlock():

Step 1: The sum of entry-wise multiplication with weighted matrix WM and the data pixel block is stored in $sumOfMatrixElements$

Step 2: The modulo 16 of $sumOfMatrixElements$ is stored in SUM

Step 3: Then, the SUM is converted to its equivalent 4-bit binary number string. These 4-bits are actually the extracted data bits.

Step 4: These 4-bits data are then appended to $TotalExtractedBits$ bit string.

Step 5: The absolute value of $changedValue$ is collected from info matrix and stored into $absChangedValue$.

Step 6: The $posX$ and $posY$ are the row and column index of $absChangedValue$ present in the weight matrix WM.

Step 7: Pixel value in the data pixel matrix in the position $posX$ and $posY$ is incremented or decremented by 1 if the value of $changedValue$ is $+ve$ or $-ve$, respectively. The pixel value remains unchanged if $changedValue$ is zero.

Step 8: Corner pixels are extracted from (3×3) data pixel matrix to form original (2×2) cover image pixel block.

Extraction-Main():

Step 1: The function $ExtractDataBitsInPixelBlock()$ is called 5 times for a pixel block for a particular color component. In each iteration, 20 bits of data are extracted. So, for three different color components (R, G, B), a total of 60 bits of secret data are extracted from a particular color pixel block.

Step 2: Then, the secret data or image is formed from the collected extracted data from the pixel blocks.

Step 3: The original cover image is formed by eliminating interpolated pixels from the dual stego images and then merging the remaining pixel blocks from both the images.

Numerical Example for Extraction

Step 1: Two 3×3 pixel blocks SI1 and SI2 are taken from the first and the second stego images. The weighted matrix used during embedding is also taken.

221	138	055
172	161	151
123	184	245

SI1

221	144	055
172	161	150
123	184	245

SI2

1	2	3
4	5	6
7	8	5

WM

Step 2: An elementary multiplication is done between the pixel block SI1 and the weight matrix WM to get the matrix below:

221	276	165
688	805	906
861	1472	1225

Step 3: Sum of elements of this matrix is calculated as $221 + 276 + 165 + 688 + 805 + 906 + 861 + 1472 + 1225 = 6619$

Step 4: Modulo 16 of the sum is then calculated as $6619 \ mod\ 16 = 11$.

Step 5: This decimal number 11 is converted to 4-bit binary number as $(1011)_2$. This is the extracted 4-bit secret message.

Step 6: The difference (d') is calculated from SI2 as $(221 + 055)/2 - 144 = -6$ where 221 and 055 are the corner pixels of the SI2 block.

Step 7: As $d' = -6$, the pixel value at position 6 (i.e., 151) of SI1 pixel block becomes 150 by decrementing 1. No change is required in the SI2 pixel block. So, for the next iteration, the pixel block SI1 and SI2 becomes:

221	138	055
172	161	150
123	184	245

SI1

221	144	055
172	161	150
123	184	245

SI2

Overflow and Underflow Control

An overflow or underflow situation occurs when the value of the updated pixel becomes more than 255 or less than 0 respectively. During embedding, the value of d is added or subtracted from the original pixel (OP) to generate the new interpolated pixel (IP). For example, it is considered that OP = 248, d = 8 and $sign$ = 1, then IP = $OP + d$ = (248 + 8) = 256. Here, an overflow situation arises as the value of IP exceeds the maximum gray scale value 255. Similarly, consider OP = 6, d = 7 and $sign$ = -1, then $IP = OP - d = (6 - 7) = -1$. Here, the value of IP is less than zero. This indicates that an underflow occurs. So, the pixel value needs to be adjusted before embedding in order to avoid these overflow and underflow situations. The value of IP is updated via the equation 4.2.

$$IP = \begin{cases} 247 + (sign)d, & OP > 247 \\ 8 + (sign)d, & OP < 8 \\ OP + (sign)d, & \text{otherwise} \end{cases} \tag{4.2}$$

$$d' = \begin{cases} IP - 247, & IP > 247 \\ IP - 8, & IP < 8 \\ IP - OP, & \text{otherwise} \end{cases} \tag{4.3}$$

During extraction of d, equation 4.3 is followed in case of overflow and underflow situation. For an example, during embedding, if $OP = 248, d = 8\ and\ sign\ = 1$, which satisfy the first rule of equation 4.2, IP becomes $247 + (1)8 = 255$. During extraction, as $IP = 255$, which satisfies the first rule of equation 4.3, d' becomes $IP - 247 = 255 - 247 = 8$ (i.e $d = 8$ and $sign = 1$). Again, during embedding, if $OP = 7, d = 8$ and $sign = -1$, which satisfy the second rule of equation 4.2, IP becomes $8 + (-1)8 = 0$. During extraction, as $IP = 0$, which satisfies the second rule of equation 4.3, d' becomes $IP - 8 = 0 - 8 = -8$ (i.e. $d = 8$ and $sign = -1$).

4.2.3 Experimental Results and Comparisons

The DSWM scheme is tested using standard benchmark images shown in Fig. 1.1 and evaluated using various evaluation metrics. The experimental results and comparisons are given below:

4.2.3.1 Quality Measurement Analysis

The distortion in the stego images is measured by Peak Signal to Noise Ratio $(PSNR)$ and Structural Similarity Index $(SSIM)$. The $PSNR$ is calculated using equation 1.2. The difference between the original and stego images was assessed by PSNR (dB). The higher the PSNR, the better the visual quality of the stego image. The SSIM is the measurement method for predicting the perceived quality of a digital image. It is measured by the equation 1.4, and it is calculated on various window sizes of an image. Table 4.5 displays the experimental results after embedding logo image within the standard images Lena, Baboon, Barbara, Zelda, Pepper. It shows PSNR, Payload, SSIM, and statistical analysis results of the DSWM scheme. It is observed that after hiding $1,754,460$ bits secret data within the original image of size (342×342) pixels the PSNR is near 47.5 dB for Lena image. So, the maximum embedding capacity of the proposed DSWM scheme is

Table 4.5: *Experimental results after embedding maximum amount of secret information in DSWM*

Cover	Secret	PSNR				SD			CC	SSIM	
(513x513)	(Bits)	CD1 vs SI1	Payload (CD1)	CD2 vs SI2	Payload (CD2)	CD1	SI1	SI2		CD1 vs SI1	CD2 vs SI2
Lena	1,754,460	47.1681	2.22	47.1418	2.22	128.6353	128.6813	128.6744	0.9994	94.76	94.73
Baboon	1,754,460	47.1399	2.22	47.1551	2.22	125.5678	125.6188	125.6328	0.9994	97.88	97.88
Pepper	1,754,460	44.6909	2.22	44.6580	2.22	135.9671	134.4708	134.4844	0.9994	94.13	94.11
Barbara	1,754,460	47.1540	2.22	47.1712	2.22	142.6456	142.7165	142.7226	0.9995	95.83	95.85
Zelda	1,754,460	45.9836	2.22	45.9811	2.22	137.0877	137.8890	137.8840	0.9995	94.06	94.05

$$= \frac{60\times \text{ no of blocks}}{3 \times (513 \times 513) \text{ pixel}} = \frac{60 \times 513/3 \times 513/3}{3 \times (513 \times 513)}\, bpp = \frac{60}{27}\, bpp = 2.22\, bpp.$$

Table 4.6 shows the comparison of the proposed scheme with existing dual image based steganographic schemes in terms of PSNR and payload after embedding maximum amount of data. The table also displays the PSNR of color image and grayscale image which are used in the proposed DSWM scheme. It has been observed that embedding capacity in color image is higher than the grayscale image without compromising the visual quality. This approach can hide $1,754,460$ bits within color image which are $1,230,172$ bits, $951,565$ bits, and $1,197,408$ bits greater than Chang et al. [100] , Chang et al. [101] and Qin et al. [102] schemes, respectively. Table 4.7 shows PSNR (dB) values of different existing RDH schemes. The embedding

Table 4.6: *Comparison of proposed scheme with existing dual image based scheme in terms of PSNR(dB) and payload(bpp) in DSWM*

Image (513 x 513)	**Comparison Metric**		**Chang et al. [100]**	**Chang et al. [101]**	**Qin et al. [102]**	**DSWM (Color Image)**	**DSWM (Grayscale Image)**
Lena	PSNR	Dual 1	45.19	45.21	45.21	47.1681	46.5391
		Dual 2	45.2	45.21	45.21	47.1418	46.4859
	Payload		524288	802895	557052	1,754,460	584,820
Baboon	PSNR	Dual 1	45.18	39.91	52.04	47.1399	46.2188
		Dual 2	45.19	39.91	41.56	47.1551	46.5991
	Payload		524288	802524	557096	1,754,460	584,820
Pepper	PSNR	Dual 1	45.21	39.94	51.25	44.6909	44.0908
		Dual 2	45.21	39.94	41.52	44.6580	44.0031
	Payload		523356	799684	557245	1,754,460	584,820
Barbara	PSNR	Dual 1	45.20	39.89	52.12	47.1540	46.5879
		Dual 2	45.21	39.89	41.58	47.1712	46.2987
	Payload		524288	802888	557339	1,754,460	584,820
Zelda	PSNR	Dual 1	45.66	39.88	51.72	45.9836	45.1746
		Dual 2	45.2	39.88	41.75	45.9811	45.0049
	Payload		524288	802789	557264	1,754,460	584,820

Table 4.7: *Comparison of proposed scheme with existing reversible data hiding schemes in terms of average PSNR (dB) and payload (bpp) in DSWM*

Scheme	Lena (513 × 513)		Baboon (513 × 513)		Pepper (513 × 513)		Barbara (513 × 513)	
	Secret Image(bits)	PSNR(dB)	Secret Image(bits)	PSNR(dB)	Secret Image(bits)	PSNR(dB)	Secret Image(bits)	PSNR(dB)
Hwang et al. [103]	5336	48.22	5208	48.20	15300	48.40	7501	48.25
Hu et al. [104]	60241	48.69	21411	48.34	77254	48.86	28259	48.40
Luo et al. [105]	71674	48.82	22696	48.36	84050	48.94	38734	48.50
Abadi et al. [106]	73207	48.78	45043	48.52	86964	48.92	43420	48.51
Jung et al. [23]	155598	41.87	324795	38.10	152448	41.49	209222	38.98
Yang et al. [107]	155598	45.77	324795	42.03	152448	44.41	209222	43.30
Jana B. [99]	776224	35.80	776224	36.12	776224	35.78	776224	35.42
Lu [108]	786732	36.03	786732	35.99	786732	36.3	786732	36.03
Kuo et al. [109]	851958	37.24	851958	37.25	851958	37.25	851958	37.25
DSWM(Grayscale Image)	584820	47.19234	584820	47.2735	584820	47.1456	584820	44.2371
DSWM(Color Image)	1754460	47.15495	1754460	47.1475	1754460	47.1624	1754460	44.7388

capacity of the DSWM scheme is $1,754,460$ bits and the visual quality of the proposed scheme is greater than 47 dB. The PSNR of the DSWM scheme is much higher than other existing RDH schemes after embedding maximum $584,820$ bits in a grayscale cover image and $1,754,460$ bits in a color cover image.

Table 4.8 shows experimental results of PSNR (dB) and capacity (bits) for different image sizes. It shows that the average PSNR is greater than 47 dB for all image sizes. The maximum secret bits embedded for each image size is also tabulated. Fig. 4.6, and Fig. 4.7 shows the comparison graph with respect to visual quality measured by PSNR (dB) and secret information embedding capacity measured by payload (bits) of DSWM scheme with other existing schemes,

Table 4.8: *Comparison of PSNR (dB) and capacity (bits) for different image sizes in DSWM*

Original Image Size	Interpolated Image Size	Average PSNR (dB)	Maximum Secret Bits (bits)
50x50	75x75	47.0036	37500
100x100	150x150	47.0189	150000
256x256	384x384	47.0987	983040
342x342	513x513	47.11423	1754460
512x512	768x768	47.15495	3932160
682x682	1023x1023	47.18765	6976860
1024x1024	1536x1536	47.18879	15728640

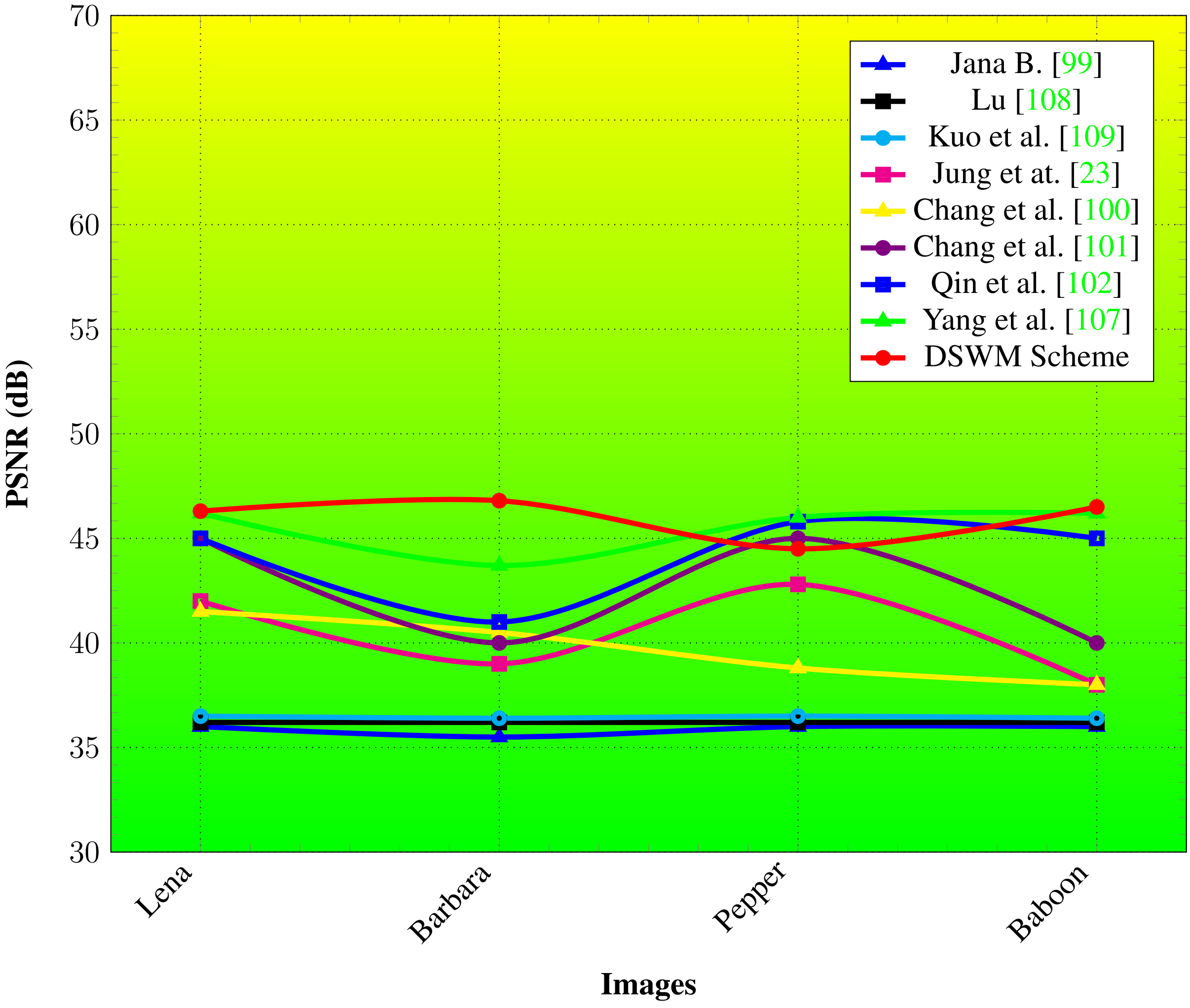


Figure 4.6: *Comparison graph with existing schemes in terms of PSNR (dB)*

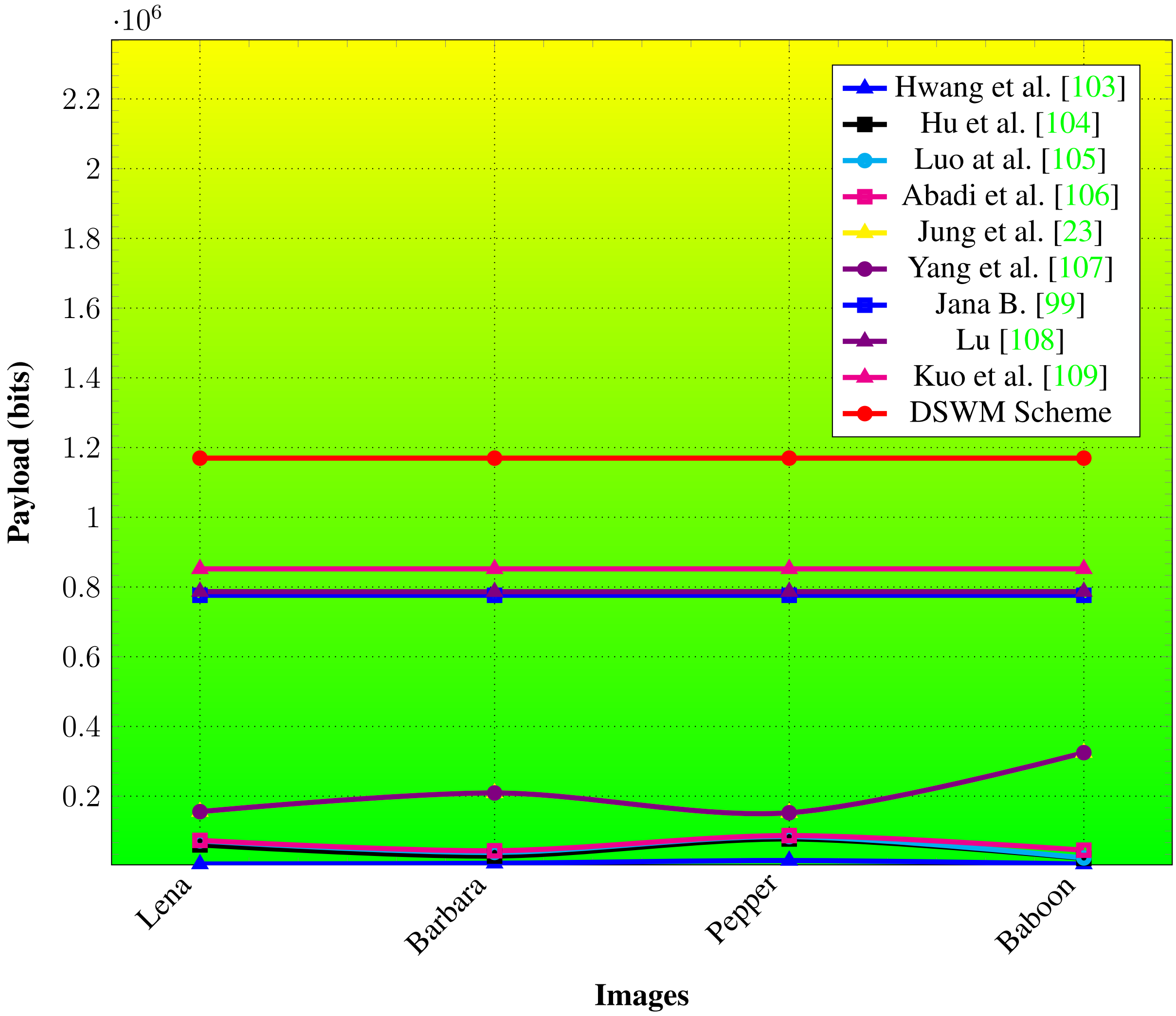
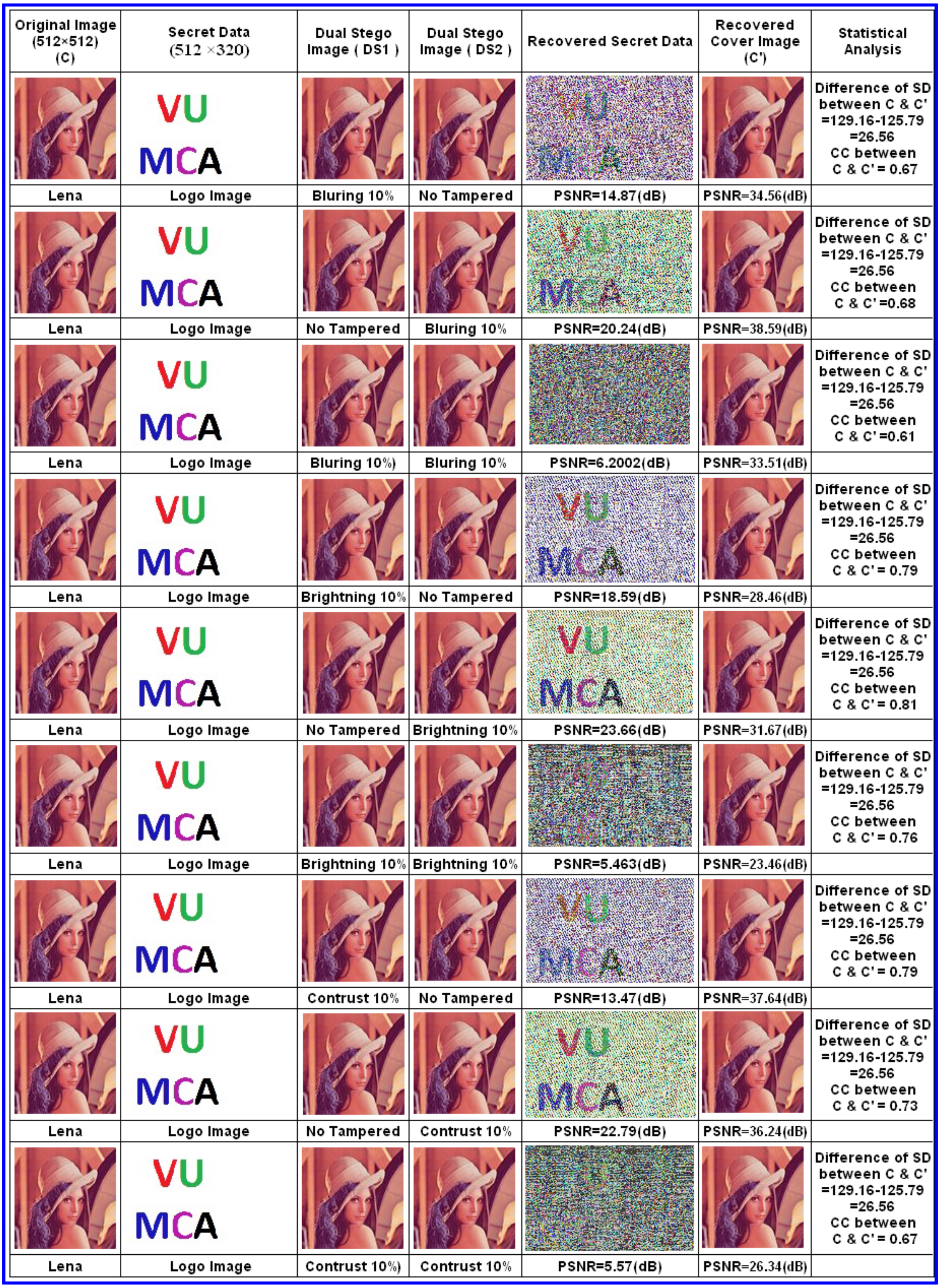


Figure 4.7: *Comparison graph with existing schemes in terms of payload (bits)*

respectively. It has been observed that in both cases, the proposed DSWM scheme achieves a better result than other existing schemes. In terms of PSNR, the DSWM scheme is 1.38 dB and 11.12 dB greater than Lu [108] and Yang et al. [107] schemes, respectively. Also, the DSWM scheme can hide 5 times and 25 times more secret data than Lu [108] and Yang et al. [107] scheme, respectively. The payload of the DSWM scheme is 2.22 bpp.

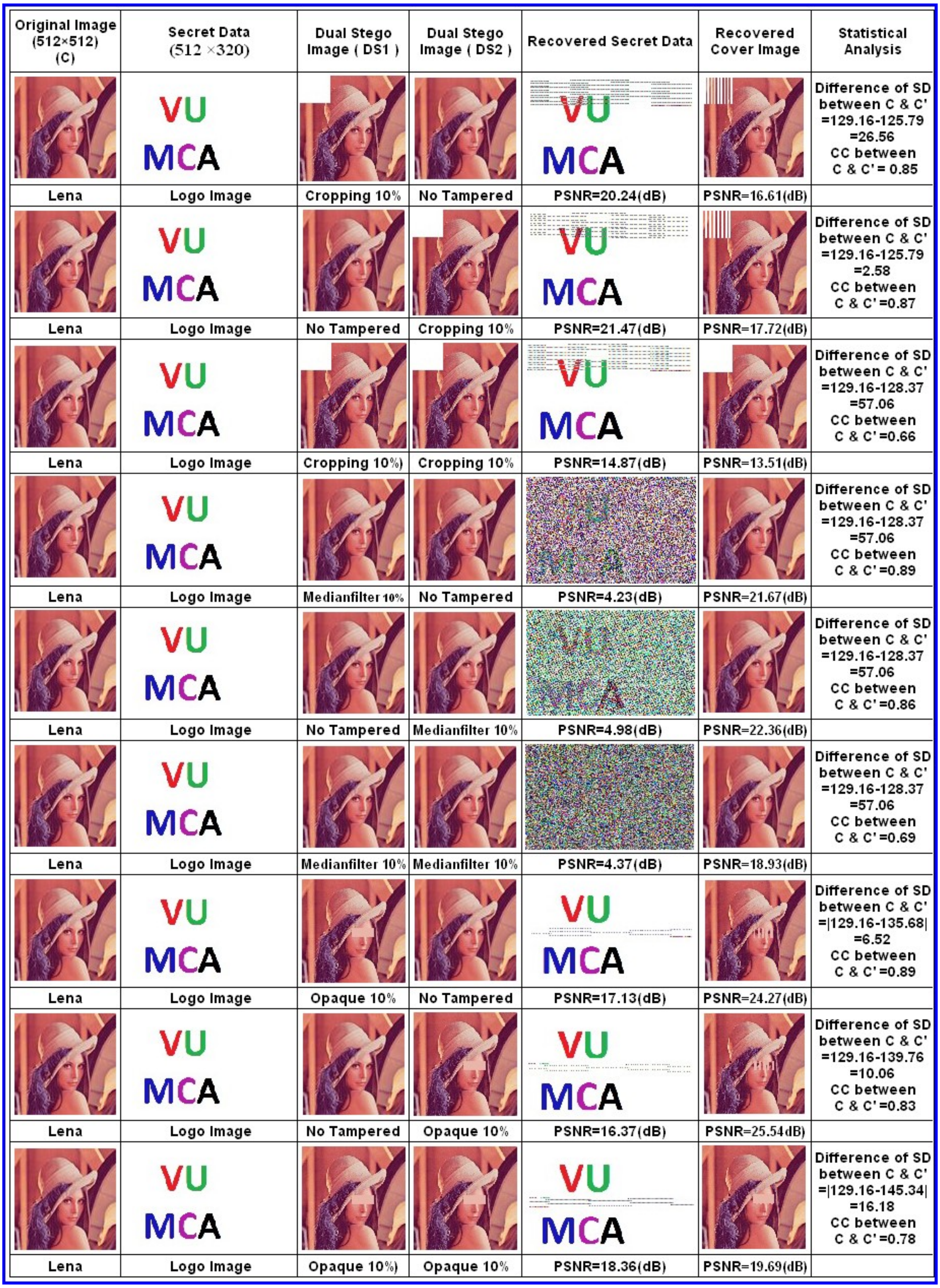
4.2.3.2 Robustness Analysis

The DSWM scheme has been evaluated through some standard analysis and attacks like Standard Deviation (SD), Correlation Coefficient (CC) and Brute force attack. The DSWM scheme is assessed by some image parameters like SD (σ) and CC (ρ) to check statistical distortion in the stego image after embedding a large amount of secret information. The SD and CC of the original and stego image are summarized in Table 4.5. Minimizing the parameter differences



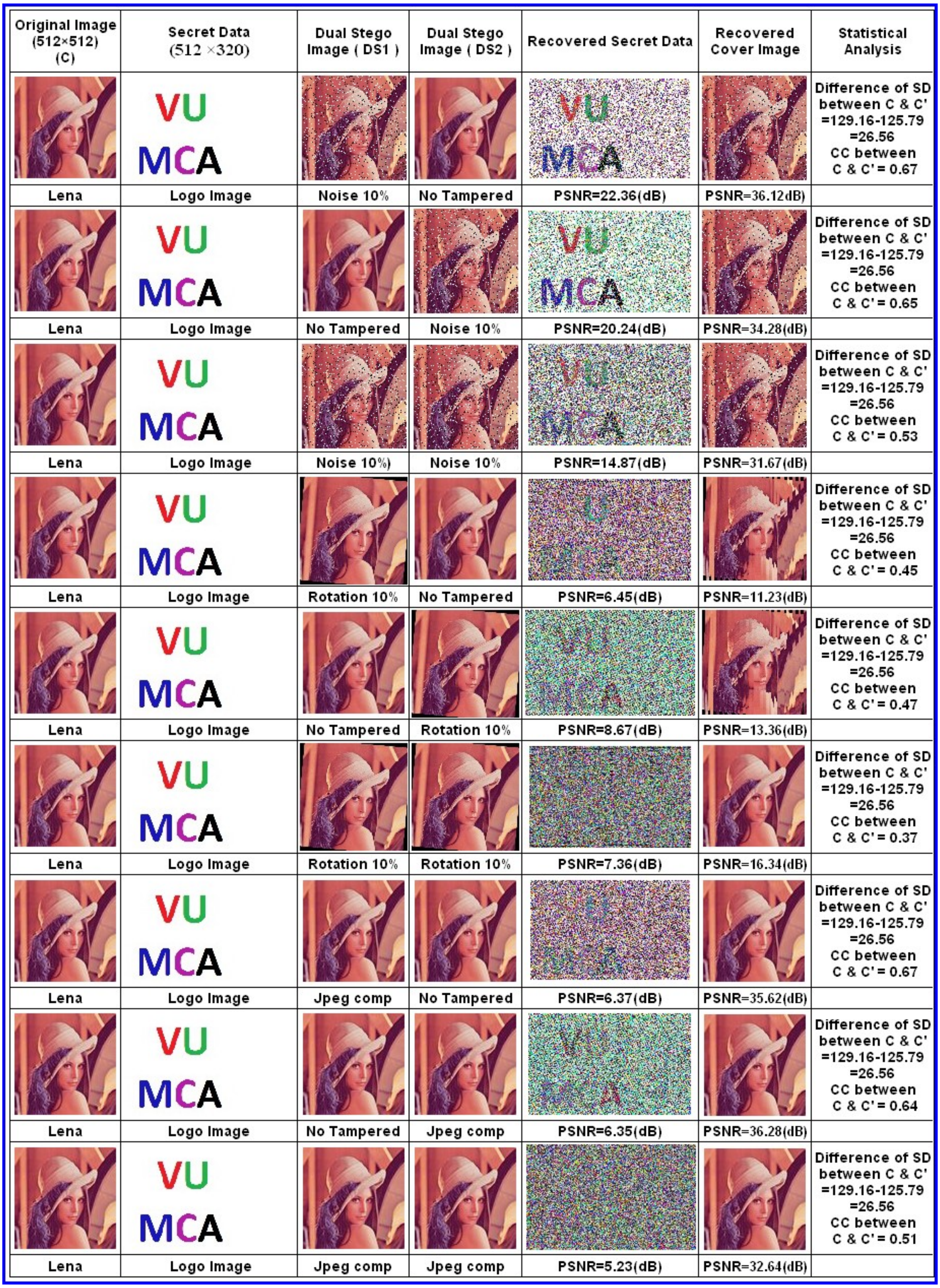
Original Image (512×512) (C)	Secret Data (512 ×320)	Dual Stego Image (DS1)	Dual Stego Image (DS2)	Recovered Secret Data	Recovered Cover Image (C')	Statistical Analysis
Lena	Logo Image	Bluring 10%	No Tampered	PSNR=14.87(dB)	PSNR=34.56(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.67
Lena	Logo Image	No Tampered	Bluring 10%	PSNR=20.24(dB)	PSNR=38.59(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' =0.68
Lena	Logo Image	Bluring 10%)	Bluring 10%	PSNR=6.2002(dB)	PSNR=33.51(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' =0.61
Lena	Logo Image	Brightning 10%	No Tampered	PSNR=18.59(dB)	PSNR=28.46(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.79
Lena	Logo Image	No Tampered	Brightning 10%	PSNR=23.66(dB)	PSNR=31.67(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.81
Lena	Logo Image	Brightning 10%	Brightning 10%	PSNR=5.463(dB)	PSNR=23.46(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.76
Lena	Logo Image	Contrust 10%	No Tampered	PSNR=13.47(dB)	PSNR=37.64(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.79
Lena	Logo Image	No Tampered	Contrust 10%	PSNR=22.79(dB)	PSNR=36.24(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.73
Lena	Logo Image	Contrust 10%)	Contrust 10%	PSNR=5.57(dB)	PSNR=26.34(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.67

Figure 4.8: *Secret information extraction from stego image when tampered by blurring, brightening and contrast stretching in DSWM*



Original Image (512×512) (C)	Secret Data (512 ×320)	Dual Stego Image (DS1)	Dual Stego Image (DS2)	Recovered Secret Data	Recovered Cover Image	Statistical Analysis
Lena	Logo Image	Cropping 10%	No Tampered	PSNR=20.24(dB)	PSNR=16.61(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.85
Lena	Logo Image	No Tampered	Cropping 10%	PSNR=21.47(dB)	PSNR=17.72(dB)	Difference of SD between C & C' =129.16-125.79 =2.58 CC between C & C'=0.87
Lena	Logo Image	Cropping 10%)	Cropping 10%	PSNR=14.87(dB)	PSNR=13.51(dB)	Difference of SD between C & C' =129.16-128.37 =57.06 CC between C & C'=0.66
Lena	Logo Image	Medianfilter 10%	No Tampered	PSNR=4.23(dB)	PSNR=21.67(dB)	Difference of SD between C & C' =129.16-128.37 =57.06 CC between C & C'=0.89
Lena	Logo Image	No Tampered	Medianfilter 10%	PSNR=4.98(dB)	PSNR=22.36(dB)	Difference of SD between C & C' =129.16-128.37 =57.06 CC between C & C'=0.86
Lena	Logo Image	Medianfilter 10%	Medianfilter 10%	PSNR=4.37(dB)	PSNR=18.93(dB)	Difference of SD between C & C' =129.16-128.37 =57.06 CC between C & C'=0.69
Lena	Logo Image	Opaque 10%	No Tampered	PSNR=17.13(dB)	PSNR=24.27(dB)	Difference of SD between C & C' =|129.16-135.68| =6.52 CC between C & C'=0.89
Lena	Logo Image	No Tampered	Opaque 10%	PSNR=16.37(dB)	PSNR=25.54dB)	Difference of SD between C & C' =129.16-139.76 =10.06 CC between C & C'=0.83
Lena	Logo Image	Opaque 10%)	Opaque 10%	PSNR=18.36(dB)	PSNR=19.69(dB)	Difference of SD between C & C' =|129.16-145.34| =16.18 CC between C & C'=0.78

Figure 4.9: *Secret information extraction from stego image when tampered by cropping, median filtering and opaque in DSWM*



Original Image (512×512) (C)	Secret Data (512 ×320)	Dual Stego Image (DS1)	Dual Stego Image (DS2)	Recovered Secret Data	Recovered Cover Image	Statistical Analysis
Lena	Logo Image	Noise 10%	No Tampered	PSNR=22.36(dB)	PSNR=36.12(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.67
Lena	Logo Image	No Tampered	Noise 10%	PSNR=20.24(dB)	PSNR=34.28(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.65
Lena	Logo Image	Noise 10%)	Noise 10%	PSNR=14.87(dB)	PSNR=31.67(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.53
Lena	Logo Image	Rotation 10%	No Tampered	PSNR=6.45(dB)	PSNR=11.23(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.45
Lena	Logo Image	No Tampered	Rotation 10%	PSNR=8.67(dB)	PSNR=13.36(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.47
Lena	Logo Image	Rotation 10%	Rotation 10%	PSNR=7.36(dB)	PSNR=16.34(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.37
Lena	Logo Image	Jpeg comp	No Tampered	PSNR=6.37(dB)	PSNR=35.62(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.67
Lena	Logo Image	No Tampered	Jpeg comp	PSNR=6.35(dB)	PSNR=36.28(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.64
Lena	Logo Image	Jpeg comp	Jpeg comp	PSNR=5.23(dB)	PSNR=32.64(dB)	Difference of SD between C & C' =129.16-125.79 =26.56 CC between C & C' = 0.51

Figure 4.10: *Secret information extraction from stego image when tampered by noise, rotation and jpeg compression in DSWM*

are required to avoid the chance of being traced through the statistical attacks. From Table 4.5, it is found that the difference of the standard deviation between the cover image and the stego image is almost negligible.

The σ of the original image and stego images are 128.6813 and 128.6744 respectively, whereas their difference is 0.0069 for Lena image after embedding $1,754,460$ bits. The ρ between the image C and SI is 0.9994 for Lena image which indicates that any change in the cover image will produce a change in the stego image in the same direction. As the data is embedded distributively among the dual images, it is hard to find the position of embedding in the stego image. Again, the degree of modification in the stego image is almost negligible as compared to the original image. This minimizes the chance of noticing any secret data in the stego image. Thus, it is proved to a very secure steganographic scheme.

Figure 4.8, 4.9 and 4.10 depict some interesting experimental results during secret information extraction from tampered dual stego images through the DSWM algorithm. Nine different kinds of tampered dual stego images are used for the experiment to show the presence of secret information within tampered dual stego images. The tampering through blurring, brightening, and contrast stretching are shown in Fig. 4.8. After performing secret information extraction, it has been observed that DSWM scheme successfully detects the tamper in stego images and particularly the logo image that is present within the tampered dual stego images. The tampering through opaque, cropping, and median filtering are shown in Fig. 4.9. It is shown that the hidden secret logo has been recovered using extraction algorithm and tampered cover image is also extracted after tampering through cropping and opaque but can not recover from median filtering. The difference of σ between the cover image and stego images are 2.58, which is negligible and ρ is 0.97, which points to negligible changes. However, in all the cases, the presence of secret information from tampered stego image has been found. The Fig. 4.10 shows the data extraction from tampered dual stego image using noise, rotation, and JPEG compression. It is observed that the secret information can not be fully retrieved but tampering in stego images can be detected.

The dual stego images of DSWM scheme have been tested by RS analysis [41] to measure the perceptibility. From Table 4.9, it is observed that the values of R_M and R_{-M}, S_M and S_{-M} are almost same for stego image. It implies that their difference is nearly zero. So, the DSWM scheme is secure against RS attack. The DSWM scheme protects multimedia documents by

Table 4.9: *Results of RS analysis in DSWM*

Image	Data (Bits)	Stego Image										
		R_M		R_{-M}		S_M		S_{-M}		RS value		
		SI1	SI2	SI1	SI2	SI1	SI2	SI1	SI2	SI1	SI2	Average
Lena	1,754,460	58422	56973	57430	55439	30990	30974	31255	31897	0.0141	0.0279	0.0210
Baboon	1,754,460	50534	50228	50059	50554	39252	39544	39620	39463	0.0094	0.0045	0.0070
Barbara	1,754,460	52046	51649	52383	52328	37082	37156	36681	36603	0.0083	0.0139	0.0111
Zelda	1,754,460	51837	51923	51647	51927	31239	31483	31534	31531	0.0058	0.0006	0.0032
Pepper	1,754,460	56802	56766	56205	56518	40447	40918	41103	40968	0.0129	0.0031	0.0080

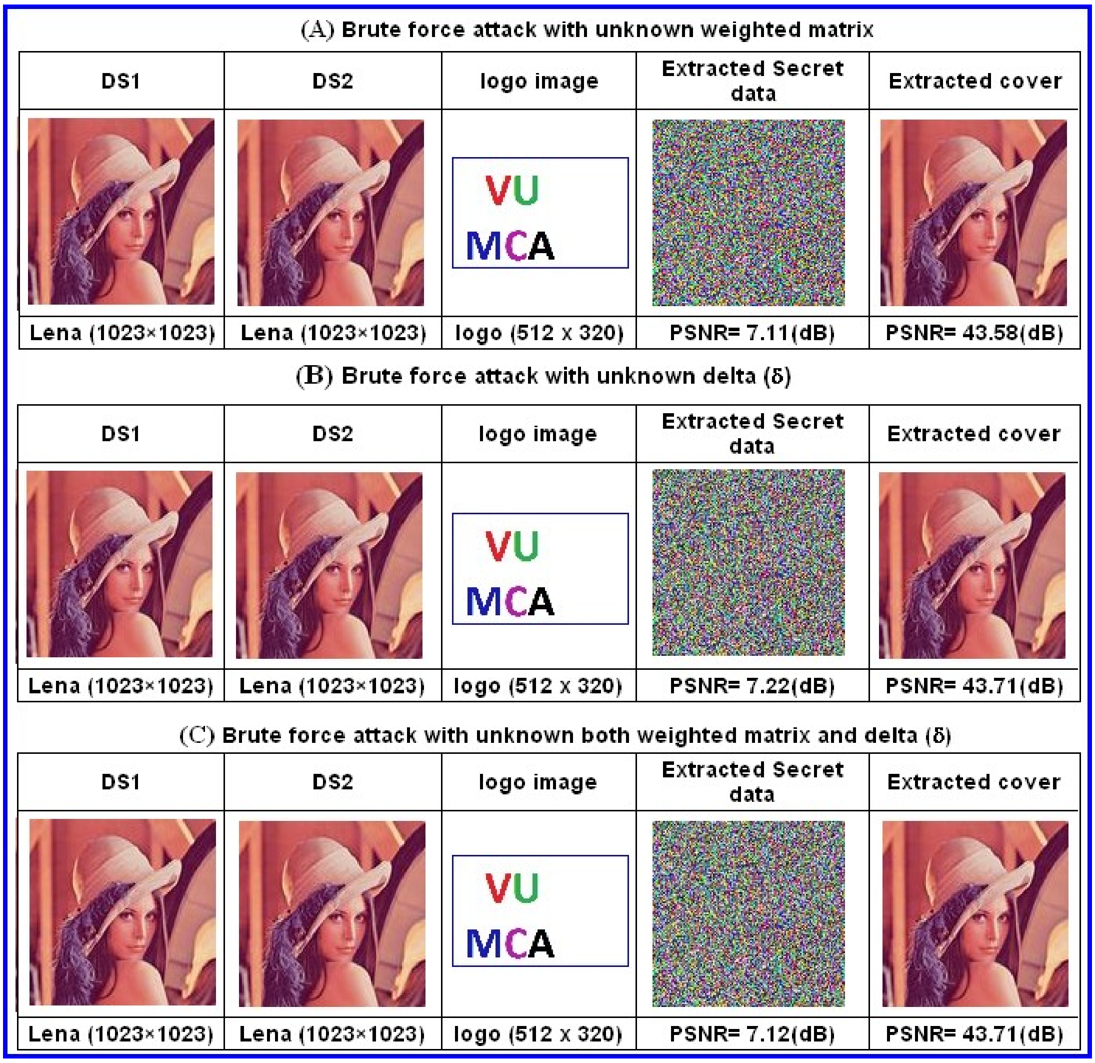


Figure 4.11: *Experimental results of brute force attack with wrong weighted matrix and secret key in DSWM*

embedding a secret image (logo) using the weighted matrix. Here, instead of the original secret information, the position values of the secret bits are stored within dual images. The δ has been used to update the weighted matrix during each operation on every R, G, B block. The DSWM scheme is secured to prevent possible malicious attacks. The Fig. 4.11 shows the results when wrong weighted matrix and wrong secret key δ are used to reveal the secret image. From the figure, it is clear that the secret logo cannot be revealed without a proper secret key (δ) and a proper weighted matrix (WM). For example, Fig. 4.11 shows the original image, logo image, attack with unknown weighted matrix and δ. The result shows that only noise like images will be generated when wrong weighted matrix and/or wrong secret key (δ) are used to reveal the logo image. But, in all cases, the extracted cover image is similar to the original image with good visual quality. In this context, it is mentioned that the proposed DSWM scheme is highly robust, and only the authorized person can extract the logo image from the stego image. Furthermore, a brute force attack may be tried by an attacker to reveal the secret message. In a $(m \times n)$ image there are $(2^{r-1} + 1)! \times (m/3 \times n/3)$ possible ways to reveal the hidden message, where r denotes the number of secret bits to be embedded which enhances the security of the scheme. Again, the distribution of original and stego pixel among the two stego images have been performed depending on the odd and even pixels. The maximum possible number of the weighted matrix that can be formed is $(2^{r-1}+1)! \times (m/3 \times n/3)$ to retrieve secret message, which is hard to compute. It is robust against brute force attack because such a huge number of attempts are computationally impractical for present generation computers. The DSWM scheme exhibits high robustness against several attacks. Again, the secret logo image along with the original image can be fully retrieved from the stego image when a valid weighted matrix and a secret key are used.

4.3 Analysis and Discussion

The comparison of the two proposed dual image based steganographic schemes (DSGN and DSWM) is summarized in Table 4.10. The table indicates that the payload is better in DSWM scheme, whereas the visual quality of the image in terms of PSNR is better in the DSGN scheme. The visual quality is better in DSGN scheme because the data is embedded into a similar type of pixels in a pixel block. A better payload in the DSWM scheme is attained because of the use of interpolated pixels and repeated application of weighted matrix to embed data into the interpolated pixels. Both the DSGN and DSWM schemes are reversible. The reversibility has been achieved using the dual stego images. The weighted matrix and the shared secret key increase the security of both the schemes. The RS analysis is used to analyse the stego images. The relative entropy and the correlation coefficient (ρ) have been calculated. These are shown in Table 4.11. The steganographic schemes discussed in this section is not robust against jpeg compression, rotation, scaling, filtering. To improve the security against attacks, two new dual image based steganographic schemes in transform domain have been proposed in the next chapter.

Table 4.10: *Comparison between DSGN and DSWM in terms of PSNR (dB) and payload (bpp)*

Schemes	Reversible/Irreversible	Capacity (bits)	PSNR (dB)	Payload (bpp)
DSGN	Reversible	777965	53.27	1.33
DSWM	Reversible	3932160	47.15	2.22

Table 4.11: *Comparison of steganalysis values of DSGN and DSWM*

Schemes	RS Value	Relative Entropy	CC
DSGN	0.0398	0.0097	0.9852
DSWM	0.0574	0.0083	0.9869

Key features of the schemes discussed in this chapter:

i) Reversibility along with high imperceptibility can be achieved in the steganographic scheme by using dual image.

ii) Shared secret key is used to distribute secret image between dual stego images.

iii) The security is enhanced through dual image based steganographic scheme using graph neighbourhood and weighted matrix.

iv) A weight matrix is used as a shared secret key where a list of random numbers is used to form the matrix.