An Improved Robust and Adaptive Watermarking Algorithm Based on DCT

Q. Zhang*, Y. Li, X. Wei

Key Laboratory of Advanced Design and Intelligent Computing
(Dalian University) of Ministry of Education, Dalian, Liaoning 116622
*zhangq@dlu.edu.cn

ABSTRACT

This paper proposes an improved watermarking algorithm based on DCT (Discrete Cosine Transform). We carried out the algorithm as described as follows. First, we extended both rows and ranks of the watermark by using the proposed method before the embedding stage. After expansion, Sine chaotic system is employed in encrypting the watermark. In the embedding stage, an effective and adaptive embedding method is proposed to embed the watermark into the blocked DCT coefficients. Experimental results demonstrate that the proposed algorithm in this paper works well in resisting both geometry attack and noise attack. It also does well in recovering the watermark after stego image suffered from JPEG compression.

Keywords: Watermark, DCT, chaotic, adaptive, robustness.

1. Introduction

In recent years, with the application and popularization of computer network, there has been a growing concern on the security of digital media information. Extensive research has been done in watermarking since it was first officially proposed by Tirkel in 1993[1]. The main reason is that it improves the security of secret information greatly by hiding its existence on the basis of the conventional encryption technology. In this way, it is hard for hackers to distinguish secret information from that which is not.

Various watermarking methods have been proposed. According to the watermark embedding domain, we can divide them into two main categories: embedding information in spatial domain and embedding information in transformative domain such as DFT (Discrete Fourier Transform), DCT, DWT (Discrete Wavelet Transform), etc. The former one embeds secret information in host image directly while the latter one embeds secret information in transformed domain coefficients of the host image. In spite of the simple operation of the former method, the latter one is more robust, thus, it drives more attention. Among the methods based on transformed domain, methods based on DCT and are DWT widely studied. Based on former research, this article proposes an improved watermarking scheme based on DCT [6~9].

2. Related work

2.1 Sine chaotic system

Its equation is as follows:

$$Xn+1 = \lambda \times \sin(\pi \times Xn) \tag{1}$$

When $\lambda = 0.99$, the system enters the chaotic stage. Experimental data show that its sequence has a good performance in image encryption because of the system's perfect capability (sensitive dependence on initial conditions).

2.2 DCT

DCT is an abbreviation for discrete cosines transform, which is widely applied among the various image processing operations. Its rationale is: with the symmetry of Fourier transform, we adopt image border folding operation to change the image into an even function (cosines). Then apply two-dimensional Fourier transforms to the obtained image. Results from the previous operations will then include cosine items only [2, 5].

One-dimensional discrete cosine transform (DCT) and its inverse transform (IDCT) will be shown as follows:

$$C(u) = a(u)\sum_{x=0}^{N-1} f(x)\cos[\frac{(2x+1)}{2N}]$$

$$u = 0,1,2,\cdots,N-1 \qquad (2)$$

$$f(x) = \sum_{x=0}^{N-1} a(u)C(u)\cos[\frac{(2x+1)u\pi}{2N}]$$

$$x = 0,1,2,\cdots,N-1 \qquad (3)$$

$\mathrm{a(u)}$ is defined as:

$$a(u) = \begin{cases} \sqrt{\frac{1}{N}} & u = 0 \\ \sqrt{\frac{2}{N}} & u = 0,1,2,\cdots,N-1 \end{cases} \qquad (4)$$

Two-dimensional discrete cosine transform (DCT) and its inverse transform (IDCT) are shown as follows:

$$C(u,v) = a(u)a(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos[\frac{(2x+1)u\pi}{2N}]\cos[\frac{(2y+1)v\pi}{2N}]$$

$$u,v = 0,1,2,\cdots,N-1$$

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} a(u)a(v)C(u,v)\cos[\frac{(2x+1)u\pi}{2N}]\cos[\frac{(2y+1)v\pi}{2N}]$$

$$x,y = 0,1,2,\cdots,N-1$$

(5)

$\mathrm{C(u,v)}$ is host image $\mathrm{A's}$ $\mathrm{DCT2}$ coefficient (A is $\mathrm{N \times N}$). In this paper, we adopt $\mathrm{DCT2}$ since it is more suitable for our purpose.

2.3 Algorithm proposed in this paper

Flowcharts in Figure 1 represent the main process of the proposed scheme; host image I is the image we use to carry the secret watermark. Stego image II is the image we get after the embedding process. Stego image II can be transmitted through Internet because it is hard for hackers to recognize whether the image carries extra information or not. It is especially useful in confidential conditions. 1-2 show its extracting process (i-preprocess refers to the inverse process of the host image I).

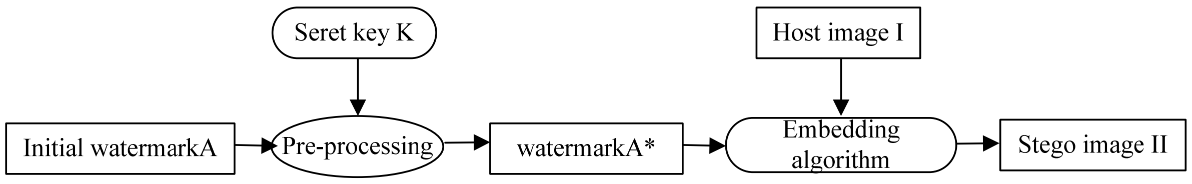
2.4 Embedding process

The embedding process is divided in two stages: the preprocessing stage and the watermark embedding stage. The preprocessing stage includes two steps: the expansion step and the scrambling step. The first step consists of extending the watermark to a size four times larger than the initial mark in order to improve the algorithm's robustness. The chaotic sequence generated by the Sine equation is applied in the scrambling step. The watermark we choose here is a $32 * 32$ binary image while the host image is a $256 * 256 * 8$ gray scale one. Details are listed next.

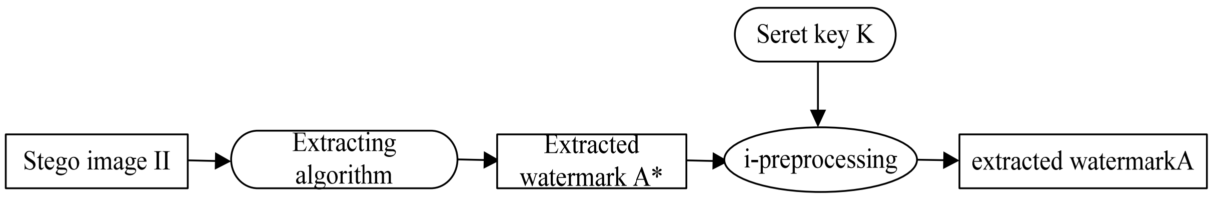
(1) **Expansion of the initial watermark**. Use the following formula to extend the watermark:

$$\begin{cases} A'(i,j) = A(i,j) \\ A'(i+m,j) = A(i,j) \\ A'(i,j+n) = A(i,j) \\ A'(i+m,j+n) = A(i,j) \end{cases} \tag{6}$$

A is the initial watermark, m and n are the dimensions of the rows and ranks of A, respectively. $\mathrm{A(i,j)}$ is the value of pixel in location $(\mathrm{i,j})$ of A . $\mathrm{A'}$ is the watermark after expansion. A is a 32*32 binary image while $\mathrm{A'}$ after the above operations is a 64*64 binary image.



flowchart of the embedding process



1-2 flowchart of the extracting process

Figure 1. Flowcharts of the watermarking process.

(2) **Scrambling stage**. With two different initial values we can obtain two chaotic sequences using Sine equation, which are set to $X1$ and $X2$ respectively. Choose $X1$'s 1001th~1064th and $X2$'s 801th~864th numbers to create two new sequences, which we set to $x1$ and $x2$. With the sort function in MATLAB we can get two sorted index number sequences $y1$ and $y2$.Then exchange the pixel's position in A' according to the following formula:

$$\begin{cases} t = A'(i,j) \\ A^*(i,j) = A'(Y1(i), Y2(j)) \\ A^*(Y1(i), Y2(j)) = t \end{cases} \tag{7}$$

t is an intermediate variable, A^* is watermark after the scrambling stage; we can perform scrambling operations several times in order to enhance its robustness. The initial value chosen for the Sine equation and times of the scrambling operations done can be kept as secret key for the extracting process.

The above steps finish the watermark's pre-processing. Here we choose $(1.9, 2.6)$ as secret key. The first and second parameters are the two initial values for the Sine equation, the last one is the iterated times of the scrambling operations.

(3) Perform the DCT2 operation in the host image. When the size of each block is designated to 8*8, then the host image can be divided into blocks. After the operation, we can get small coefficient matrixes [10].

(4) **Selection of coefficients to embed the watermark**. In order to reach the aim of robustness, we select coefficients that are relatively smaller after attacks of the image to embed secret information. Based on the analysis of a large amount of experimental data, we finally choose coefficient matrixes of the host image after 20% compression as the reference matrixes to achieve better performance. We choose the four least changed coefficients in each block to embed the watermark. Take the first block as an example. The host image we use here is a gray scale image . Set its coefficient matrix to B, and set the matrix after compression (20%) to . We can get the deference matrix as the following Table 1 shows:

-3.3516	0.0054	-0.0042	0.0133	-0.0054	0.0056	0.0082	0.0152
0.0202	**0.0003**	-0.0024	-0.0049	-0.0025	0.0023	-0.0183	0.0021
0.0028	-0.0104	-0.0026	0.0067	-0.0156	-0.0144	0.0031	-0.0154
0.0111	0.0056	0.0153	**-0.0001**	0.0021	0.0054	-0.0092	0.0090
-0.0009	**0.0004**	-0.0092	0.0021	-0.0097	0.0116	-0.0110	0.0029
-0.0147	-0.0068	-0.0084	0.0198	0.0110	0.0113	0.0263	-0.0086
-0.0030	**-0.0001**	-0.0067	0.0018	0.0114	0.0206	-0.0041	0.0092
0.0071	-0.0178	-0.0011	-0.0015	0.0119	-0.0218	-0.0036	0.0109

Table 1. deference matrix.

As Table 1 shows, coordinates of the four least values are $(4,4)$, $(7,2)$, $(2,2)$ and $(5,2)$, respectively. The same operations are performed in each block. Later, four bits of the watermark will be embedded into the same pixel of the selected location in every small block.

(5) **Watermark embedding step**. Coordinates of the four selected coefficients are set in each deference matrix to $(a1,b1)$, $(a2,b2)$, $(a3,b3)$ and $(a4,b4)$, respectively. $|Q(a1,b1)|<=|Q(a2,b2)|<=|Q(a3,b3)|<=|Q(a4,b4)|$. Here we use S to denote embedding intensity. With the increase of the value of the four deference coefficients, their intensity is set to S, $\frac{2}{3}S$, $\frac{1}{2}S$ and $\frac{1}{2}S$, separately. Specific operations are described as the following formula:

$$\begin{array}{lll} if & A*(i,j)=1 & B(a1,b1)=B(a1,b1)+S \\ if & A*(i,j)=0 & B(a1,b1)=B(a1,b1)-S \\ \cdots\cdots\cdots & & \\ if & A*(i+1,j+1)=1 & B(a4,b4)=B(a4,b4)+\frac{1}{2}S \\ if & A*(i+1,j+1)=0 & B(a4,b4)=B(a4,b4)-\frac{1}{2}S \end{array}$$

(8)

$A*$ is watermark after pretreatment, B is a divided coefficient matrix.

2.5 Extracting process

With the inverse steps in the embedding process described above, we can get the extracted watermark with the correct key .It is just the same as the initial mark. Exact experimental data will be listed next. There is an important point we should pay more attention to; when it comes to the inverse-expansion stage, use the following formula to finally extract the watermark:

$$s = A(i,j) + A(i,j+n) + A(i+m,j) + A(i+m,j+n)$$

(9)

A is watermark after i-scrambling (inverse operation of the scrambling step), m and n are numbers of the initial watermark's row and rank. When $s>=3$, we set the extracted bit to 1, otherwise set it to 0. With four extracted pixels, we determine one pixel's value. In this way it is more effective to resist attacks such as geometry attack. The extracted watermark also has a larger precision.

3. Analysis of the proposed scheme

A very important parameter used to measure the performance of the watermarking algorithm is the NC-normalization coefficient. It is quite effective for measuring the similarity of the initial watermark and the extracted one.

$$NC = \frac{\sum_i \sum_j W'(i,j) \times W(i,j)}{\sum_i \sum_j W(i,j)^2}$$

(10)

W denotes the initial watermark and W' represents the extracted watermark. The visual effect of the obtained watermark increases as $NC's$ increase.

3.1 Experimental results

We can get the stego image II after the above embedding steps. Results are shown in Figure 2.

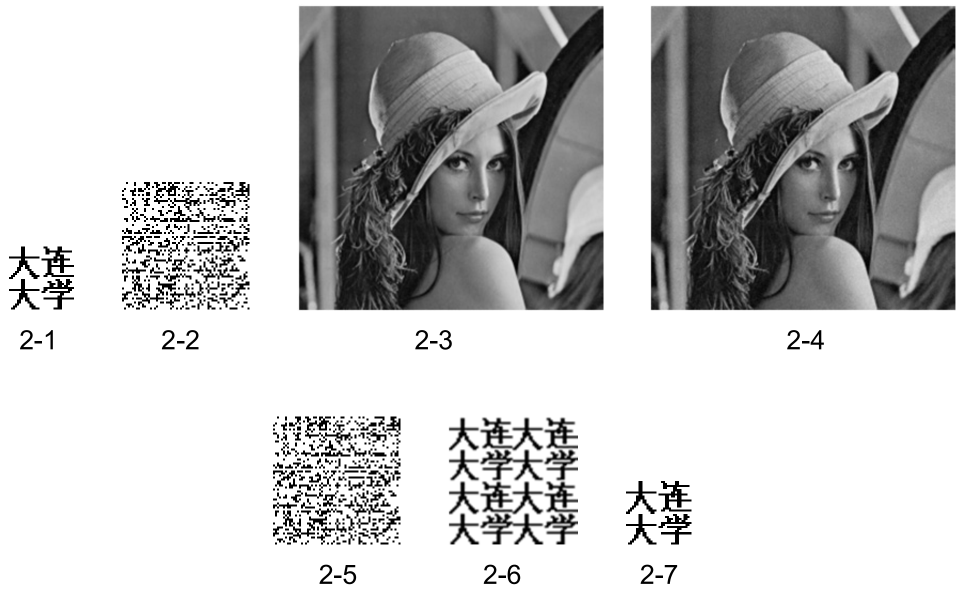


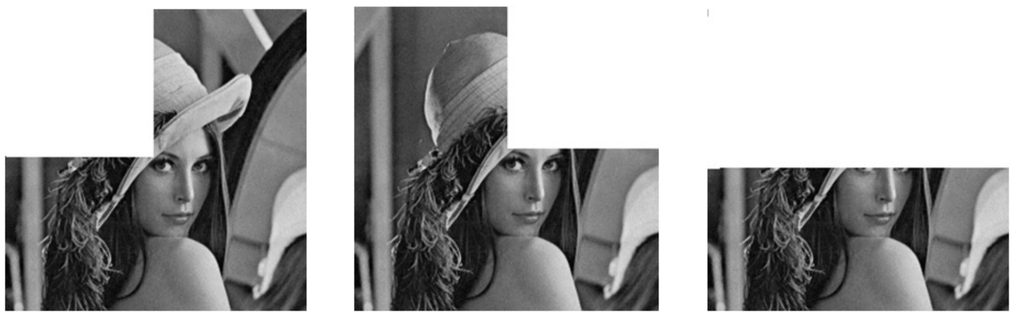
Figure 2. Experimental results: 2-1 is the initial watermark; 2-2 watermark after steps 2.1(1) and (2); 2-3 host image lena I; 2-4 stego image II; 2-5 extracted watermark; 2-6 watermark after i-pre-processing; 2-7 final extracted watermark.

2-1 shows the initial watermark we intend to embed. 2-2 is the image after 2.1 (1) and (2), it is the one we actually embedded. From comparison between 2-3 and 2-4, we can see that the stego image is so similar to the host image that they can hardly be differentiated from each other, which means that our scheme has a good performance in invisibility. As we calculated, the value of NC between the initial watermark and the extracted one is 1. This experiment sufficiently shows the efficiency of our method.

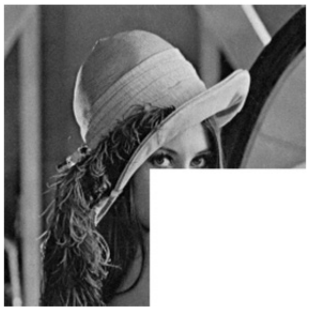
3.2 Robustness analysis

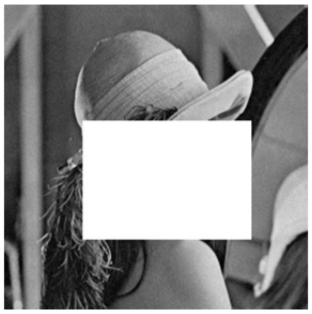
Various attacks are inevitable when the original secret information transfers from the transmitter to the receiver. Therefore, the performance of the method is dependant on the ability to resist attacks to a great degree, which refers to the robustness of the method. In the following analysis, the main attacks include cropping attack, graffiti attack, JPEG compressor attack and noise attacks.

3.2.1 Cropping attack and graffiti attack



3-1 Top left corner 3-2 Top right corner 3-3 Top part


3-4 Bottom right


3-5 Middle key part

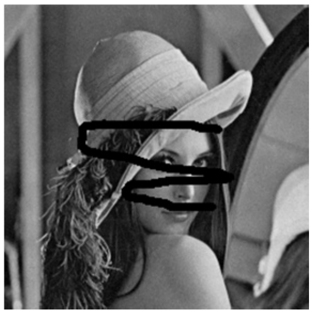

3-6 Graffiti

Figure 3. Stego image after attacks

Extracted results are shown in Table 2:

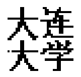
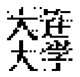
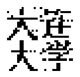
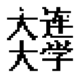
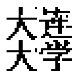
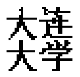
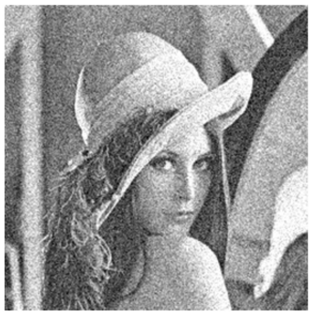
Attack	Fig. 3-1	Fig.3-2	Fig.3-3	Fig.3-4	Fig.3-5	Fig.3-6
NC	1	0.9845	0.9845	0.9986	0.9944	1
Extracted watermark	 大连大学	 大连大学	 大连大学	 大连大学	 大连大学	 大连大学

Table 2. deference matrix.

From the results shown in Table 2 we can see that our scheme does very well at resisting cropping and graffiti attacks with both host image Lena and host image girl.Though the stego image is cropped by a rather large part, NC of the extracted watermark with our scheme is 1or value close to 1.Thus this is an evident strong point of this algorithm.

3.2.2 Noise attack and JPEG *compression*

4-1 Gaussian noise (1%)


4-2 Gaussian noise (20%)

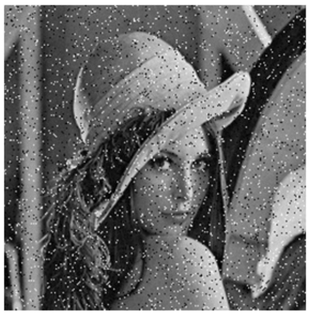

4-3 Salt and pepper (2%)

Figure 4. Stego image after attacks.

Gaussian noise	1%	5%	8%	20%	50%	80%
NC	0.9211	0.9282	0.9211	0.9056	0.8986	0.8127
Salt and pepper noise	0.4%	0.8%	2%	4%	6%	10%
NC	1	0.9859	0.9479	0.9099	0.8986	0.8352
JPEG attack	90%	80%	70%	50%	20%	15%
NC	1	0.9887	0.9634	0.9268	0.8887	0.8099

Table 3. Analysis of results.

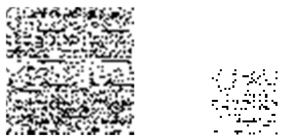
Attack	cropping	Gaussian (5%)	JEPG (15%)
Ref.[4]	0.91(small cut)	0.61	0.91
This paper	1 (cropped 1/4)	0.9282	0.8099

Table 4. Table 4. Comparisons.

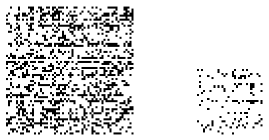
From Table 3 we can see it also does well at resisting Gaussian attack, salt and pepper attack and JPEG compression attack. From the comparison between the method proposed in [4] and our scheme as shown in Table 4, it is clear our scheme is more effective; though it does do better in all respects, it is superior overall.

3.2.3 Key space analysis

With $(1.9,2.5,6)$ used in the embedding process, we can extract the exact watermark correctly. If we use $K1$ $(1.90000000000001,2.5,6)$ and $K2$ $(1.9,2.5,20)$ to decrypt, the wrong results are obtained as shown in Figure 5; it is evident that its key space is lager enough to resist exhaustive attack. By a simple calculation, we can obtain the key space $s >> 2^{200}$.



5-1 Extracted results with K1

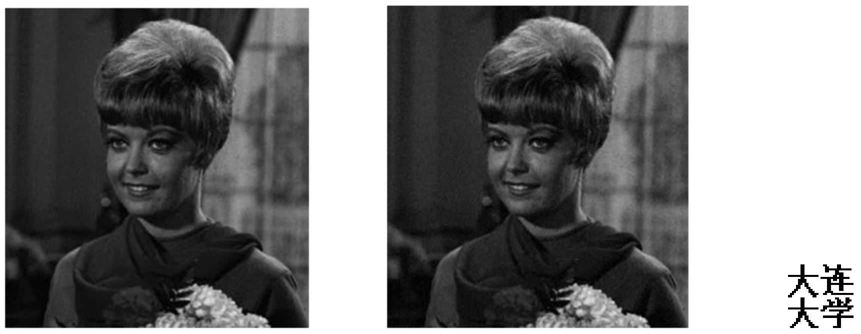


5-2 Extracted results with K2

Figure 5. Extracted results with the wrong keys.

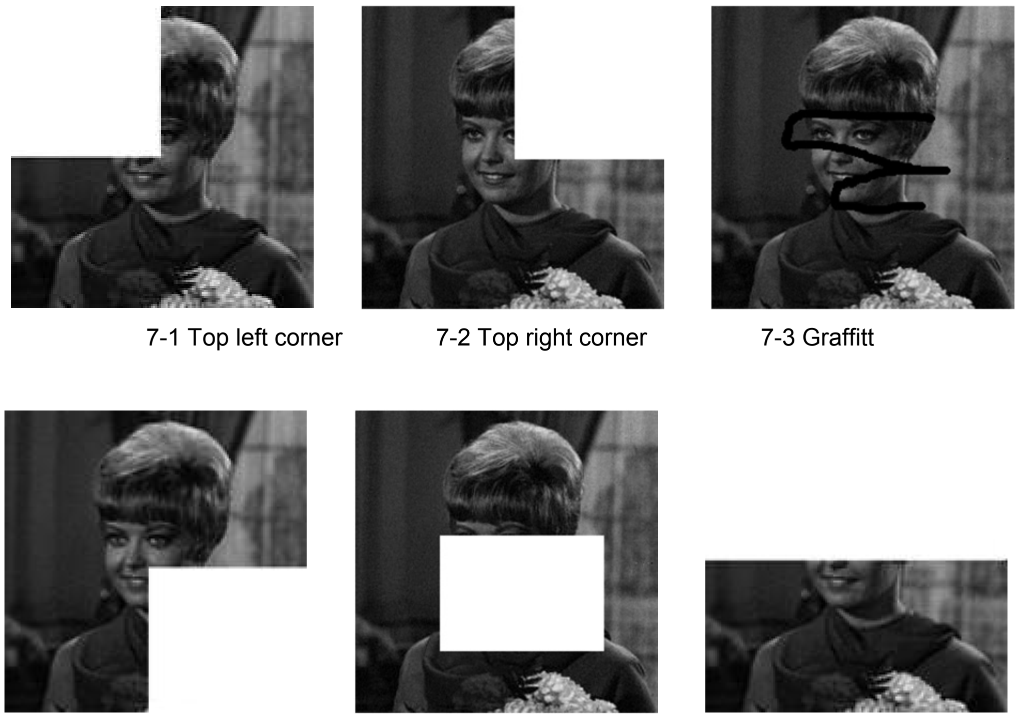
3.2.4 Results for another host image

In this section, in order to further show the results, we will change the host image from Lena to Girl with the same size. The experimental results are shown in Figure 6. The results for the same abovementioned attacks are illustrated in Figure 6 and Tables 5-9.



6-1 Host image *girl* 6-2 *girl* After watermark embedded 6-3 Extracted watermark

Figure 6. Experimental results for the image *girl*.



7-1 Top left corner 7-2 Top right corner 7-3 Graffitt

7-4 Bottom right corner 7-5 Middle key part 7-6 Top part

Figure 7. Stego image after attacks.

Attack	Fig.6-1	Fig.6-2	Fig.6-6	Fig.6-4	Fig.6-5	Fig.6-3
NC	1	1	0.9958	1	0.9986	1

Table 5. Results after cropping and graffiti.

Gaussian noise	gs（1%）	gs（5%）	gs（8%）	gs (20%)	gs (50%)	gs (80%)
NC	0.9254	0.9324	0.9085	0.9141	0.9155	0.8817

Table 6. Results after Gaussian noise attacks.

Salt & pepper noise	sp（0.4%）	sp（0.8%）	sp（2%）	sp (4%)	sp (6%)	sp (10%)
NC	0.9958	0.9859	0.9310	0.8845	0.8746	0.8352

Table 7. Results after salt and pepper attacks.

JPEG attack	90%	80%	70%	50%	20%	15%
NC	1	0.9901	0.9789	0.9634	0.8549	0.7592

Table 8. Results after $JPEG$ attacks.

	Stego $Lena$	Stego $girl$
$PSNR$	49.76	52.83

Table 9. Experimental $PSNR$ of the stego image.

4. Conclusions

The algorithm we proposed in this paper has a good performance in anti-attacks. In the pre-processing stage we do expansion of the watermark before its scrambling operations. After the scrambling step with the Sine system, a watermark similar to noise is generated. In the embedding procedure, an adaptive method which has a good performance is adopted to embed the watermark in the coefficients effectively. In the inverse pre-processing stage of the extracting procedure, we make good use of the expansion features. This results in better performance. As the experimental results show, our scheme is perfectly effective for resisting geometry attack. It also does well at resisting noise attack and JPEG compression attack.

Acknowledgment

The work was supported by the National High Technology Research and Development Program ("863" Program) of China (No. 2009AA01Z416), the National Natural Science Foundation of China (31170797, 30870573), Program for Changjiang Scholars and Innovative Research Team in University (No.IRT1109), the Program for Liaoning Innovative Research Team in University (No. LT2011018), the Program for Liaoning Excellent Talents in University (No. LR201003), the Program for Liaoning Innovative Research Team in University (No. LT2011018) and by the Program for Liaoning Science and Technology Research in University (No. LS2010179).

References

[1] A. Z. Tirkel, et al. Electronic watermark. Digital image computing Technology and Applications, Macquarie University, 1:666-673, 1993.

[2] Fang Liu, Feng Yang. An improved blind watermarking algorithm based on DCT. Computer Engineering and Applications. 45(13):124-126, 2009.

[3] Bing-Xi Wang. Watermarking Technology. Xi'an publisher of Xi Dian University Xi'an China, 2003.

[4] Guo-Ming Wang, Zheng-Feng Hou. Watermarking scheme base on DCT. Computer Engineering and Design. 29(21):5635-5637, 2008.

[5] Chin-Chen Chang ,Chia-Chen Lin, Chun-Sen Tseng, Wei-Liang Tai. Reversible hiding in DCT-based compressed images information Sciences .177: 2768-2786, 2007.

[6] C.C. Chang, T.S. Chen, L.Z. Chung. A steganographic method based upon JPEG and quantization table modification, Information Sciences 141:123-138, 2002

[7] M. Iwata, K. Miyake, A. Shiozaki, Digital steganography utilizing features of JPEG images, I- EICE Transactions on Fundamentals E87-A: 929-936, 2004.

[8] Er. Ashish Bansal, Dr. Sarita Singh Bhadauria. Watermarking Using Neural Network and Hiding the trained network within the cover image. Journal of Theoretical and Applied Information Technology, 2008.

[9] Mohammad Ali Bani Younes, Aman Jantan. Image Encryption Using Block-Based Transformation Algorithm. International Journal of Computer Science, 2008.

[10] Sen Yang, Guo Ying lv. Research on robustness of watermark based on DCT. Tech Vocabulary. 174:28-29,2010.