Advances in Computer Science and Information Technology (ACSIT)
Print ISSN: 2393-9907; Online ISSN: 2393-9915; Volume 2, Number 1; January-March, 2015 pp. 43-47

http://www.krishisanskriti.org/acsit.html

Secure Message Transmission with Watermarking using Image Processing

Shivi Garg[1] and Manoj Kumar[2]

[1]M.Tech Scholar, Delhi Technological University Delhi
[2]Delhi Technological University Delhi
E-mail: shivi1989@gmail.com, 2mkg1109@rediffmail.com

Abstract—*This paper presents a system that allows the users to securely transfer the messages by hiding them in the digital images. To accommodate the messages the original cover image is slightly modified by the embedding algorithm to obtain the stego image. The system uses the scheme of watermarking along with the concepts of Image processing. Here 1-bit information is hidden in the pixels of an image. On changing this image by some image attributes like inverting, gray scale, contrast, brightness, cropping, resizing and color filtering by varied degree in the pixels of an image, the proposed system tries to compare the percentage matching in the watermark so that a proper threshold can be set to detect the presence of a watermark.*

1. INTRODUCTION

Techniques for hiding information have existed since ancient times. Earlier methods include communication via invisible inks, covert channels, microdots, and spread spectrum channels.[1] Invisible ink is invisible either on application or soon thereafter, and which later on can be made visible by some means. Invisible ink is applied to a writing surface with specialty purpose stylus, stamp, fountain pen, toothpick, calligraphy pen or even a finger dipped in the liquid. Microdots are, fundamentally, a steganographic approach to message protection. A microdot is text or an image substantially reduced in size onto a small disc to prevent detection by unintended recipients. Various techniques explored by the authors involved embedding information within digital media, specifically digital images. Data can be hidden in image files by manipulating color values of the pixel. Another digital media other than images, which can be used for steganography are video files. AVI files are created out of couple streams. Because of existence of those streams, it is possible to hide data not only in file's frames but also in mentioned audio stream.

2. DIGITAL WATERMARKING TECHNIQUE

A watermarking algorithm embeds watermark in different kinds of data like image, text, audio, video etc. The embedding process is done by using a private key which maps the locations within the multimedia object (image) where the watermark would be embedded. Once the watermark is embedded, several attacks can happen because the online object can be digitally processed. The attacks are unintentional. Hence the watermark has to be very robust against all attacks which are possible. When the owner wants to check the watermarks in the attacked and damage multimedia object, she/he depends on the private key that was used to embed the watermark. Using the secrete key, the embedded watermark can be detected. This detected watermark may or may not combine the original watermark because the image might have been attacked. Hence to validate the existence of watermark, the original data is used to compare and extract the watermark signal (non-blind watermarking) or a correlation method is used to detect the strength of the watermark signal from the extracted watermark (blind watermarking). In the correlation, detected watermark from the original data is compared with the extracted watermark.

3. PROPOSED SYSTEM

This System extends the concept of watermarking in the field of image processing[2]. This system tries to hide 1 bit information in the pixels of an image. On changing this image by some image attributes like inverting, gray scale, contrast, brightness, cropping, resizing and color filtering by varied degree in the pixels of an image, system tries to compare percentage matching in the watermark so that a proper threshold can be set to detect the presence of a watermark[3].

4. EMBEDDING ALGORITHM

i. In this an image is taken for which (x, y) pixel pairs are obtained. Secret key is concatenated with the pixel pair.
ii. Compute the hash of the concatenated bit pattern.
iii. Compute the mod with the shifting parameter ‘α’. This will give the position for the pixels.
iv. If the mod result is 0, then compute the position of the LSB bit by taking the mod with the position parameter ‘β’.

v. Change the 0 bit to 1 bit and count the number of pixels changed.

vi. Change the image by modifying the pixels by 10 %, 20% or by applying different image processing methods like contrast, color filter and match the number of pixels with the changed pixels.

vii. If the matched pixel is greater than the set threshold, then the watermark is detected else rejected.

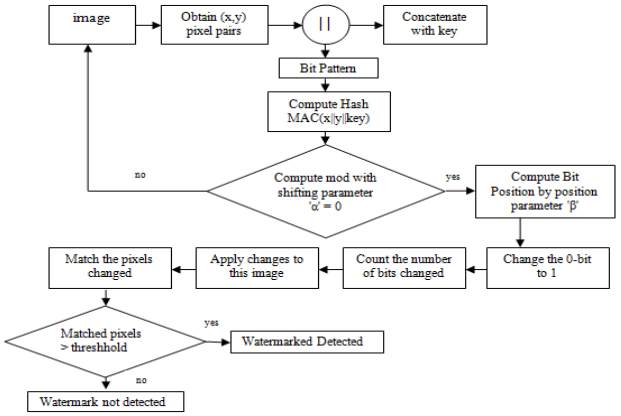
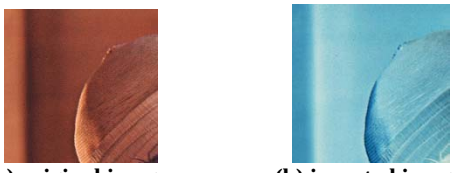


Fig. 1: Flowchart of proposed scheme

5. RESULTS AND ANALYSIS

5.1. Invert an Image

It simply inverts a bitmap, meaning that each pixel value is subtracted from 255. The Invert command inverts all the pixel colors and brightness values in the current layer, as if the image were converted into a negative. Dark areas become bright and bright areas become dark. Hues are replaced by their complementary colors[4].



(a) original image **(b) inverted image**

Fig. 5.1 (a) Original image (b) inverted image

Table 5.1: Percentage matching for the inverted image

Total bits changed	Matching bits	Percentage matched
5158	1804	34.97%

5.2. Gray Scale

Gray scale filtering is in reference to the color mode of a particular image. A gray scale image would be a black and white image; any other color would not be included in it. Basically, it's a black and white image; the colors in that image, if any, will be converted to the corresponding shade of gray (mid tones between black and white) thus making each bit of the image still differentiable[5].

(a). Original image **(b) Gray image**

Fig. 5.2 (a) Original image (b) Gray image

Table 5.2: Percentage matching for the Gray image

Total bits changed	Matching bits	Percentage matched
5158	3342	64.79%

5.3. Contrast (values between -100 and 100)

Contrast refers to the amount of color or gray scale differentiation that exists between various image features in digital images. Images having a higher contrast level generally display a greater degree of color or gray scale variation than those of lower contrast[6].

Table 5.3: Percentage matching for the Contrast image

Contrast Value	Total bits changed	Matching bits	Percentage matched	Contrast Value	Total bits changed	Matching bits	Percentage matched
+10	5158	3956	76.69%	-10	5158	3267	63.33%
+20	5158	3091	59.92%	-20	5158	3239	62.79%
+30	5158	1381	26.77%	-30	5158	3619	69.98%
+40	5158	1169	22.66%	-40	5158	2722	52.77%
+50	5158	457	8.86%	-50	5158	4867	94.35%
+60	5158	160	3.10%	-60	5158	1858	36.02%
+70	5158	147	2.84%	-70	5158	3662	70.99%

Images with positive contrast values: As the contrast value increases, the percentage matching of the pixels is reduced. Images with the positive contrast value is shown in the Fig. 5.3(i).

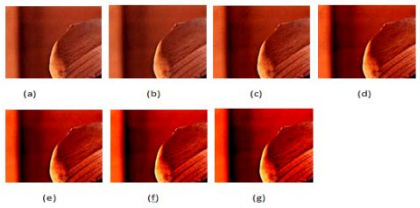


Fig. 5.3(i): Positive Contrast Image: (a) Contrast +10 (b) Contrast +20 (c) Contrast +30 (d) Contrast +40 (e) Contrast +50 (f) Contrast +60 (g) Contrast +70

Images with negative contrast values: As the contrast value is reduced, the pixel matching shows an absurd behavior of up

and down values. Images with the negative contrast is shown in the Fig. 5.3(ii).

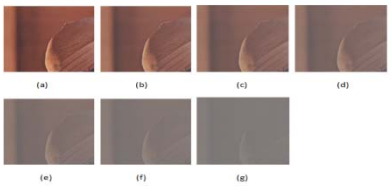


Fig. 5.3(ii): Negative Contrast Image:(a) Contrast-10 (b) Contrast -20 (c) Contrast -30 (d) Contrast -40 (e) Contrast -50 (f) Contrast -60 (g) Contrast -70

5.4. Brightness (values between -255 and 255)

Brightness refers to the overall lightness or darkness of the image. The Brightness filter adds a value to each pixel, and if we go over 255 or below 0 the value is adjusted accordingly and so the difference between pixels that have been moved to a boundary is discarded. Doing a Brightness filter of 100, and then of -100 will not result in the original image - we will lose contrast. The reason for that is that the values are clamped[7].

Table 5.4: Percentage matching for the Brightness value of an image

Brightness Value	Total bits changed	Matching bits	Percentage matched	Brightness Value	Total bits changed	Matching bits	Percentage matched
-25	5158	3398	65.87%	+25	5158	3389	65.70%
-50	5158	1985	38.48%	+50	5158	5158	100%
-75	5158	165	3.19%	+75	5158	1715	33.24%
-100	5158	25	0.48%	+100	5158	3258	63.16%
-125	5158	14	0.27%	+125	5158	2567	49.76%
-150	5158	9	0.17%	+150	5158	1814	35.17%
-175	5158	5	0.09%	+175	5158	4864	94.3%
-200	5158	0	0%	+200	5158	1891	36.66%
-225	5158	0	0%	+225	5158	14	0.27%

Image results with positive brightness are shown in the Fig. 4(i).

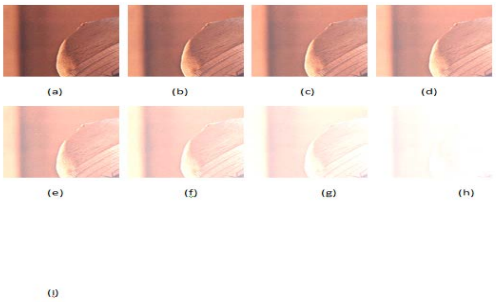


Fig. 5.4(i): Image Brightness Positive: (a) Brightness +25 (b) Brightness +50 (c) Brightness +75 (d) Brightness +100 (e) Brightness +125 (f) Brightness +150 (g) Brightness +175 (h) Brightness +200 (i) Brightness +225

Image results with negative brightness shown as in Fig. 5.4(ii).

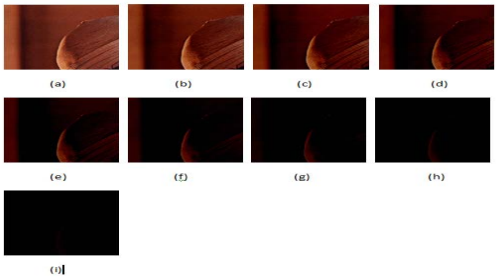


Fig. 5.4(ii): Image Brightness Negative: (a) Brightness -25 (b) Brightness -50 (c) Brightness -75 (d) Brightness -100 (e) Brightness -125 (f) Brightness -150 (g) Brightness -175 (h) Brightness -200 (i) Brightness -225

5.5. Gamma (values between 0.2 and 5 for RGB)

A gamma filter works by creating an array of 256 values called a gamma ramp for each value of the red, blue and green components [8]. The gamma value must be between 0.2 and 5.

The formula for calculating the gamma ramp is

255 * (i / 255)1/gamma + 0.5.

If this value is greater than 255, then it is clamped to 255. It is possible to have a different gamma value for each of the 3 color components. Then for each pixel in the image, we can substitute the value in this array for the original value of that component at that pixel.

Table 5.5: percentage matching for the Gamma filtered image

RGB Values	Total bits changed	Matching bits	Percentage matched
(0.2,0.2,0.2)	5158	29	0.56%
(0.5,0.5,0.5)	5158	3419	66.28%
(0.8,0.8,0.8)	5158	4050	78.51%
(1,1,1)	5158	5158	100%
(3,3,3)	5158	4757	92.22%
(5,5,5)	5158	3475	67.37

Image results for the Gamma filter as shown in the below Fig. 5.5

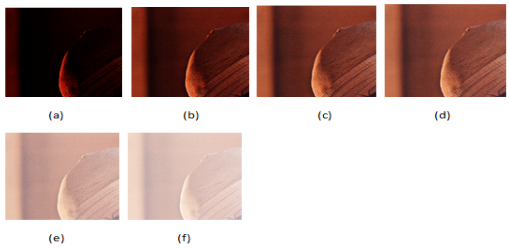


Fig. 5.5: Gamma Filter: (a) Gamma 0.2 (b) Gamma 0.5 (c) Gamma 0.8(d) Gamma 1 (e) Gamma 3 (f) Gamma 5

5.6. Color Filter

Color filters are sometimes classified according to their type of spectral absorption: short-wavelength pass, long-wavelength pass, or band-pass; diffuse or sharp-cutting; monochromatic or conversion. The short-wavelength pass transmits all wavelengths up to the specified one and then absorbs. The long-wavelength pass is the opposite. Every filter is a band-pass filter when considered generally [9].

It just adds or subtracts a value to each color. The most useful thing to do with this filter is to set two colors to -255 in order to strip them and see one color component of an image. For example, for red filter, keep the red component as it is and just subtract 255 from the green component and blue component.

Table 5.6: Percentage matching for the Color Filtered image

Filter	Total bits changed	Matching bits	Percentage matched
RED Filter	5158	0	0%
GREEN Filter	5158	0	0%
BLUE Filter	5158	5158	100%

Image results

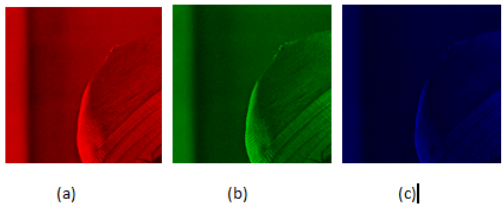


Fig. 5.6: Color Filter:
(a) RED Filter (b) GREEN Filter (c) BLUE Filter

5.7. Resize an Image

When image is resized, total number of the pixels are reduced.

Original image Dimensions: 202 X 202

Table 5.7: Percentage matching for the Resized image

Resized Dimensions	Total bits changed	Matching bits	Percentage matched
10 X 10	5158	7	0.135%
30 X 30	5158	85	1.65%
50 X 50	5158	226	4.38%
70 X 70	5158	469	9.09%
90 X 90	5158	708	13.72%
110 X 110	5158	1107	21.46%
130 X 130	5158	1464	28.38%
150 X 150	5158	2000	38.77%
170 X 170	5158	2544	49.32%
190 X 190	5158	3280	63.59%
200 X 200	5158	3634	70.45%

Image results: As the image size reduces, the pixels matching will reduce because bit information is lost will resizing it to a smaller dimension. Image results are shown as in Fig. 5.7.

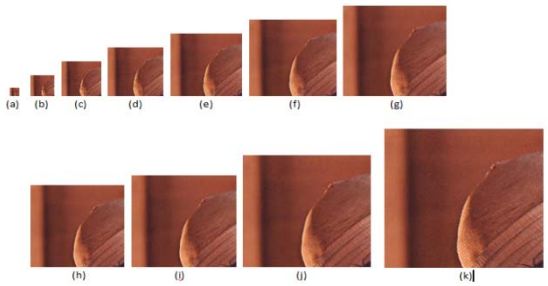


Fig. 5.7: Resize Image: (a) 10 X 10 (b) 30 X 30 (c) 50 X 50 (d) 70 X 70 (e) 90 X 90 (f) 110 X 110 (g) 130 X 130 (h) 150 X 150 (i) 170 X 170(j) 190 X 190 (k) 200 X 200

5.8. Crop an Image

When image is cropped, some material from the edges is trimmed to show a smaller area.

Table 5.8: percentage matching for the cropped image

XY Coordinates	Total bits changed	Matching bits	Percentage matched
(10 , 10)	5158	3095	60.00%
(30 ,30)	5158	3041	58.95%
(40 , 40)	5158	2150	41.68%
(50 , 50)	5158	2028	39.31%
(70 , 70)	5158	1450	28.11%
(100 , 100)	5158	855	16.57%
(120 , 120)	5158	694	13.45%
(150 , 150)	5158	267	5.17%

As the cropping area increases, the percentage matching in the pixels reduces the large amount of pixels are modified resulting in the loss of bit information thereby difficult to detect the watermark.

Image Results are shown in the Fig. 5.8 given below:

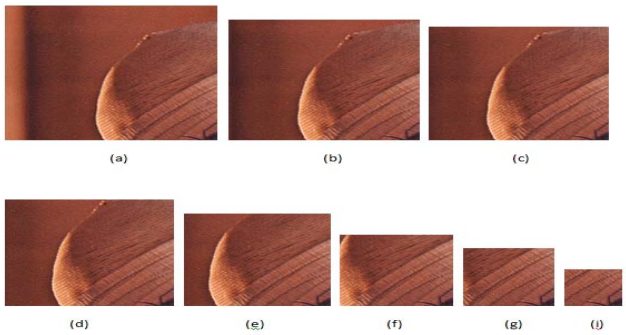


Fig. 5.8: Cropped Image: (a) (10, 10) (b) (30, 30) (c) (40, 40) (d) (50, 50) (e) (70, 70) (f) (100, 100) (g) (120, 120) (h) (150, 150)

6. CONCLUSION

The proposed scheme is robust to all the attacks .In this 1-bit information is hidden in an image based on the position parameter 'α' and hiding parameter 'β'. 'α' gives the row of the pixel where this bit information is hidden. And 'β' gives the last four LSB bits where this bit is to be hidden. Then this image is modified resulting the change in the pixel value. Then percentage of matching pixels is calculated and based on that a threshold can be set in the future.

REFERENCES

[1] Information Hiding: Steganography and Watermarking - Attacks and Countermeasures Neil F. Johnson, Zoran Duric, Sushil Jajodia, 3rd Edition, Kluwer Academic Publisher, 2003.
[2] Sangeet Saha, Chandrajit pal, Rourab paul, Satyabrata Maity, Suman Sau A brief experience on journey through hardware developments for image processing and it's applications on Cryptography University Of Calcutta, Kolkata, India.
[3] Feng Bao, Robert H. Deng, Beng Chin Ooi, Yanjiang Yang, Tailored Reversible Watermarking Schemes for Authentication of Electronic Clinical Atlas, National University of Singapore.
[4] docs.gimp.org/en/gimp-layer-invert.html, webpage.
[5] www.codeproject.com/Articles/33838/Gray/Image-Processing-using-C.
[6] www.codeproject.com/Articles/33838/Contrast/Image-Processing-using-C.
[7] www.codeproject.com/Articles/33838/Brightness/Image-Processing-using-C.
[8] www.smokycogs.com/blog/image-processing-in-c-sharp-adjusting-the-gamma.
[9] www.workspaces.codeproject.com/saleth-prakash/image-processing-using-matrices-in-csharp.