Journal of Computer Science 3 (4): 223-232, 2007
ISSN 1549-3636

Hiding a Large Amount of Data with High Security Using Steganography Algorithm

Nameer N. EL-Emam
Applied Computer Science Department, Faculty of Information Technology
Philadelphia University, Jordan

Abstract: This study deals with constructing and implementing new algorithm based on hiding a large amount of data (image, audio, text) file into color BMP image. We have been used adaptive image filtering and adaptive image segmentation with bits replacement on the appropriate pixels. These pixels are selected randomly rather than sequentially by using new concept defined by main cases with their sub cases for each byte in one pixel. This concept based on both visual and statistical. According to the steps of design, we have been concluded 16 main cases with their sub cases that cover all aspects of the input data into color bitmap image. High security layers have been proposed through three layers to make it difficult to break through the encryption of the input data and confuse steganalysis too. Our results against statistical and visual attacks are discussed and make comparison with the previous Steganography algorithms like S-Tools. We show that our algorithm can embed efficiently a large amount of data that has been reached to 75% of the image size with high quality of the output.

Key words: Hiding with high security, hiding with high capacity, adaptive image segmentation, steganography

INTRODUCTION

Steganography is an art and science of information hiding and invisible communication. It's unlike cryptography, where the goal is to secure communications from an eavesdropper by make the data not understood, steganography techniques strive to hide the very presence of the message itself from an observer so there is no knowledge of the existence of the message in the first place. In some situations, sending encrypted information will arouse suspicion while invisible information will not do so[1]. Both sciences can be combined to produce better protection of the information. In this case, when the steganography fails and the message can not be detected if a cryptography technique is used too. Hiding information inside images is a popular technique nowadays. An image with a secret message inside can easily be spread over the World Wide Web or in a news groups.

To hide a message inside an image without changing its visible properties, the cover source can be altered in "noisy" areas with many colour variations, so less attention will be drawn to the modifications. The most common methods to make these alterations involve the usage of the least-significant bit (LSB) developed by Chandramouli *et al.*[2] , masking, filtering and transformations on the cover image. Dumitrescu *et al.*[3], construct an algorithm for detecting LSB Steganography. Von Ahn *et al.*[4] are the information theoretic scientistes; they construct mathematical frameworks based on complex-theoretic view to seek the limits of steganography. Other construction is used by HweeHwa Pang [5]; his scheme used hash value obtained from a file name and password and a position of header of hidden file is located. This approach is used by the present work with new modifications. The next interesting application of steganography is developed by Miroslav Dobsicek[6], where the content is encrypted with one key and can be decrypted with several other keys, the relative entropy between encrypt and one specific decrypt key corresponds to the amount of information. Because of the continual changes at the cutting edge of steganography and the large amount of data involved, steganalysists have suggested using machine learning techniques to characterize images as suspicious or non-suspicious developed by Mittal *et al.*[7]. Pavan *et al.*[8] used entropy based technique for detecting the suitable areas in the document image where data can be embedded with minimum distortion.

When using a 24 bit colour image, a bit of each of the red, green and blue colour components can be used, so a total of 3 bits can be stored in each pixel. Thus, an 800 × 600 pixel image can contain a total amount of

Corresponding Author: Nameer N. EL-Emam, Applied Computer Science Department, Faculty of Information Technology, Philadelphia University, Jordan

1.440.000 bits (180.000 bytes) of secret data. But using just 3 bit from this huge size of bytes is wasting in size. So the main objective of the present work is how to insert more than one bit at each byte in one pixel of the cover-image and give us results like the LSB[2,3] (message to be imperceptible). This objective is satisfied by building new steganography algorithm to hide large amount of any type of information through bitmap image [4, 6] by using maximum number of bits per byte at each pixel. We discuss two types of attacks to be sure that our process for embedded data is worked efficiently. The first attack concerns to work against visual attacks[7,9] to make the ability of humans is unclearly discern between noise and visual patterns, and the second attack concerns to work against statistical attacks[3,8,10,11] to make it much difficult to automate.

New steganography algorithm with high security: New Steganography algorithm by using three layers of security has been constructed. These layers are developed from previous works[1,10] to acquire high security and they work independently to provide unbreakable security wall Fig. 1.

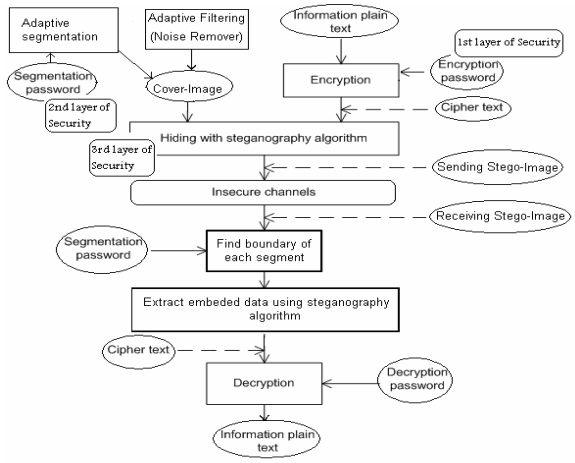


Fig. 1: Steganography with security layers

The encryption mechanism is the first layer of security for data protection by using DES or AES algorithm[1,5]. This layer is used before hiding input data and it gives us powerful Steganography algorithm. Before describing the present algorithm, we need to show the following concepts:

Adaptive segmentation of the cover-image: We introduce new concept of image segmentation by using adaptive segment to support the second layer of security. The cover-image from type bitmap is segmented into random number of uniform[1] or non-uniform segments according to the value of password or any other key as in [2, 4, 5] that is provided by the data owner (Fig. 2). At section four of this work, we have shown how to calculate a number of segments by using non-uniform which is more secure than uniform segments to carry the input data. In addition, to avoid sending files of this enormous size, a compression scheme has been employed on BMP Stego image what is known as *lossless* compression, a scheme that allows the software to exactly reconstruct the original image.

Fig. 2: Adaptive segmentation on a cover-image using uniform or non-uniform segmentations

Pixel selection style: This approach is the third layer of security in this work, as shown in Fig. 1. We perform random selection of the proper byte at each pixel of the cover-image according to the color characteristic to embed secret data. This type of selection is implemented by using new concepts which are called main cases and sub cases that have been described in the next section. These new concepts are strength to reduce the noise of a stego-image.

Descriptions of the steganography algorithm: In the present Steganography algorithm, two parts (data hiding at the sender side and data extracting at the receiver side). These parts are constructed and implemented to satisfy the following requirements:

1. The algorithm must reduce the chances of statistical detection.
2. The algorithm must provide robustness against a variety of image manipulation attacks.
3. The stego-image must not have any distortion artifacts.
4. The algorithm must not sacrifice embedding capacity in order to achieve the above requirements.

The first part is used to hide data file inside Bitmap according to the following actions:

* Accept encryption password from the sender.
* Find a maximum size (number of bytes) that is accepted by the cover-image.
* Perform compression on secret data file to increase the amount of hiding secret data.
* Perform encryption on secret data file.

* Perform the following processes on the cover-image from type bitmap:
  - Adaptive segmentation according to the password.
  - Adaptive filtering (noise remover) by using Minimum Mean-Square Error (MMSE)[2] equation (1) to reduce number of colors at the cover-image.

$$\underset{\forall i,j}{MMSE} = C_{ij} - \frac{\sigma_n^2}{\sigma_l^2}\left(C_{ij} - m_{ij}^{l}\right). \quad (1)$$

Where
C_{ij} = color byte at the Cover-image
σ_n^2 = noise variance.
σ_l^2 = local variance (in the segment under consideration).
m^l = local mean (average in segments under consideration) .

* Perform scanning to select suitable pixels on each segment by extracting image characteristics. The candidate pixels are used to embed secret data.
* Perform hiding of the secret data into bitmap images according to color characteristics.

While the second part is used to extract data from bitmap image at the receiver side in conformity with the following actions:

* Extracting password.
* Scanning segment's pixels according to the password.
* Extracting data file.
* Decrypt the extracting data file by using the password.

The present Steganography algorithm is used to hide any type of input data within a bitmap image which includes 24– bits (3 bytes of RGB colors), each byte is separated into two nibbles (four bits). The left nibble contains the highest value in the byte while the right nibble contains the lowest value in the byte. Therefore, any changes in the right nibble will cause a minimal change in a byte value. We can specify the byte effective according to the left nibble value. The nibble value is fixed by the interval [0, 15], so that we conclude that we have 16 levels of a priority, each one represents one main case (MC) out of 16. In the present work we use the following formula equation (2) to determine the index of MC that must be implemented to perform hiding in a bitmap cover-image.

$$MC = Int\left(\frac{ByteColor}{16}\right) + 1. \quad (2)$$

where ByteColor∈{ByteRed, ByteGreen, ByteBlue} represents the value of Color in decimal notation. To hide large amount of data, all three colors of one pixel are checked by the present Steganography algorithm to perform hiding data by depending on new concept which is called sub-case (SC). This concept is used to organize the pixel architect. We have defined 6 SCs that are specified according to the following:

Let us define MC^{color} = index, where 1≤ index ≤ 16 and color∈{R, G, B} where R, G, B are pixel's color equal to ByteRed, ByteGreen and ByteBlue respectively. Assume that the MC of the current color is C and X,Y are the MCs of the rest colors at the same pixel, then we can define the index of SC according to the following conditions:

$$Index\ of\ SC = \begin{cases} 1 & \Leftrightarrow & X,Y < C \\ 2 & \Leftrightarrow & (X = C\ \&\ Y < C)\,|\,(X < C\ \&\ Y = C) \\ 3 & \Leftrightarrow & X,Y = C \\ 4 & \Leftrightarrow & (X > C\ \&\ Y < C)\,|\,(X < C\ \&\ Y > C) \\ 5 & \Leftrightarrow & (X > C\ \&\ Y = C)\,|\,(X = C\ \&\ Y > C) \\ 6 & \Leftrightarrow & X,Y > C \end{cases} \quad (3)$$

To embed data in the proper pixels, it is necessary to use the priority value Pr(MC) which is calculated as follows:

$$Pr(MC) = |MC - 17| \quad (4)$$

where the priority level depends on the subscript of MC as the following formula:

$$Pr(MC_i) > Pr(MC_{i+1}) \quad \forall i \ni 1 \le i \le 15 \quad (5)$$

Before we describe the present Steganography algorithm, let us define the following notations:

Type of pixels

1- CP: - Is the Current pixel that includes the set of colors {R, G, and B}.
2- NP: - Is the next pixel of the CP.

Pixels properties:

1. Selected color:
$$Sel_{Color} = \underset{\forall Color \in CP}{\arg\min}\,(MC_{Color}) \quad (6)$$
2. Main case of the selected color in the current pixel:
$$Cmc = MC_{Sel_{Color}} \ni Color \in CP \quad (7)$$
3. Main case of the rest colors in the current pixel:
$$MC_{RC} = \underset{\forall\, Color \in RC}{arg\,min}\ (MC_{Color}) \ni Color \in CP\ \&\ RC \subset CP \quad (8)$$
4. One of the rest colors:
$$E = RC_i \quad \forall i = 1,2 \quad \& \quad E \in RC \quad (9)$$
5. Select one of the rest colors in the current pixel whose its main case is greater than main case of the selected color (Sel_c) :
$$Pr(MC_H) < Pr(MC_{Sel_C}) \ni H \in RC, C \notin RC \quad (10)$$
6. R, G and B are the red, green, and blue colors in one pixel respectively.

Journal of Computer Science 3 (4): 223-232, 2007
ISSN 1549-3636

Table 1: MC with their corresponding SC

MC	The Corresponding SC	Set Description of SC
1	$\{SC_3, SC_5, SC_6\}$	$\{\forall\ SC\ \ \exists\ C = Sel_{Color}\ \in CP\ \ \&\ RC_i\ \in CP\ \ \forall\ i = 1,2\ \ \ni\ \ C \le RC_i\ ,\}$
2-15	$\{SC_1, SC_3, SC_5, SC_6\}$	$\{\forall\ SC\ \ \exists\ C = Sel_{Color}\ \in CP\ \ and\ \ RC_i \in CP\ \forall\ i = 1,2\ \ \ni\ \ C \le RC_i \mid C \ge RC_i\}$
16	$\{SC_1, SC_3\}$	$\{\forall\ SC\ \ \exists\ C = Sel_{Color}\ \in CP\ \ and\ \ RC_i\ \in CP\ \ \forall\ i = 1,2\ \ \ni\ C \ge RC_i\}$

Properties of embedding secret message: Create Object (DataBits) which holds information about the embedded message (the content and the length of the secret message).

Properties of MC: Assume that every MC have a number of SCs, Table 1 illustrates the MC with their SCs. Practically, we conclude that SC_2 and SC_4 are neglected from any MC. This restriction is used to avoid sudden change of the colour in the stego-image.

The following pseudo code shows step by step how the present steganography algorithm can selecte the suitable pixel efficiently to embed data to be imperceptible from both visual and statistical attacks.

```
// Read data file bits , Perform Searching on all MC for each segment starting from MC=1,…16
Foreach(MCi , i=1,…,16 ) {
 Foreach(segment in the Cover-image){
  Foreach(Sel(Color)∈ Cmc)  {
    If (Cmc >= 2 AND SC=1)  {
// Check the MC of rest colors
  If ( RC = 1)  {
// Check if all current pixel property is equal to the next pixel property
  If ( CP.property = NP.property ) {
 // Hide 2 data bits in the selected color from the current pixel
    Hide(CP.C, DataBits(2));
 // Hide 2 data bits in the selected color byte from the next pixel
    Hide(NP.C, DataBits(2));
   }
   }
   }
Else if (Cmc >= 1 AND Cmc <= 16 AND SC=3 ) {
// Hide 1 data bit in the Red color byte
  Hide(CP.R, DataBits(1) );
// Hide 1 data bits in the Green color byte
  Hide( CP.G, DataBits(1) );
// Hide 2 data bits in the Blue color byte
  Hide( CP.B, DataBits(2) );
 }
Else if(Cmc >= 1 AND Cmc <= 15 AND SC=5 ) {
// Hide 2 data bits in the selected color byte from the current pixel
  Hide( CP.C, DataBits(2) );
// Hide data bits in the greater or equal MC byte for the selected color min case from the rest pixel bytes
   Hide( CP.E, DataBits(2) );
  }
Else if(Cmc >= 1 AND CP.Cmc <= 15 AND SC=6 ) {
// Hide 2 data bits in the selected color byte from the current pixel
   Hide( CP.C, DataBits(2) );
// Hide data bits in the greater or equal MC byte for the selected color min case from the rest pixel byte's
  Hide( CP.H, DataBits(2) );
 }
 }// end foreach Sel(color)
 } //end foreach segment
}// end foreach MC
```

Implementation of the present algorithm: Assume that we have a cover-image which contains three types of MC: MC1, MC_2 and MC_6 and we have three types of pixels: MC_1 with SC_3, MC_2 with SC_5 and MC_6 with SC_1. Now, we try to hide 2- bytes 01010101, 01010101. Before we perform hiding, we must compute the number of segments in the Cover-image through the following steps:

1. Let S be a number of characters in the input password (PS).
2. Find $N = Int\left(\frac{S}{2}\right)$ (11)
3. Find a number of segments on the vertical and horizontal directions $(Seg_v\ ,\ Seg_h)$ by using the following formulas:

$$Seg_v = \sum_{i=1}^{N} Val(PS_i) \quad (12)$$

$$Seg_h = \sum_{i=N+1}^{2N} Val(PS_i) \quad (13)$$

where, $Val(PS_i)$ represents the value of the i^{th} character at the PS.

4. Find the size of non-uniform segments on both directions:
   * Find number of pixels (Length) for each segment with respect to both directions:

$$\begin{aligned} Length_i &= \underset{1 \le i \le Seg_v}{Val(PS_i)} \\ k_j &= j\ mod\ s\ ;\ j = 1,...,s \\ Length_i &= \underset{1 \le i \le Seg_h}{Val(PS_i)} \\ k_j &= j\ mod\ s\ ;\ j = s,...,1 \end{aligned} \quad (14)$$

Corresponding Author: Nameer N. EL-Emam, Applied Computer Science Department, Faculty of Information Technology, Philadelphia University, Jordan

5. Perform segmentation by using column wise indexing on the cover-image into *(Seg_v x Seg_h)* segments through non-uniform size of segments. The present algorithm performs hiding into each segment separately according to row wise scanning as in Fig. 3.

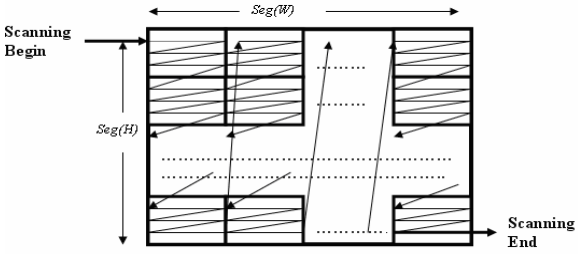


Fig. 3: Scanning pixels on the adaptive image's segments

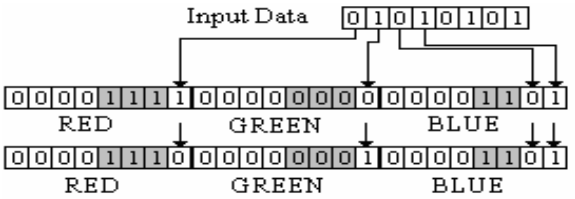


Fig. 4: Data Hiding Using MC 1 with SC 3

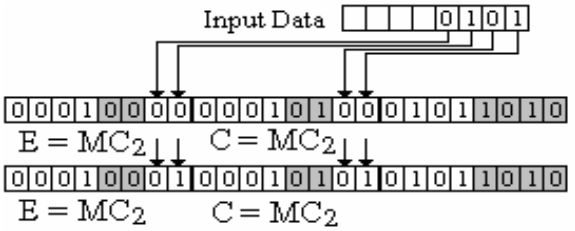


Fig. 5: Data Hiding Using MC_2 with SC_5

The algorithm starts hiding data inside the highest priority $Pr(MC_1)$, assume that CP includes the following values of colors (R=15,G=0,B=13), the MC of CP is MC_1 with SC_3. In the present Steganography algorithm, we can hide the data "01010101", as the following:

We assign the left most bit of the input data into the right most bit of the R-byte, the bit before the left most bit is assigned to the right most bit of the G-byte and we assign the rest two bits (the right nibble of input data) into the right most two bits of the B-byte as it is shown in Fig. 4.

After hiding data in all, the MC_1, it will start hiding inside the next highest priority which is MC_2 in this example. Let us say that the first pixel from MC_2 was in SC_5 and its value was of colors (R=16, G=20, B=90), it will hide the data as the following, the least 2 bit will be modified in the current selected color and in the color which it's MC equal to the current selected color MC, as it is shown in Fig. 5.

And then after hiding the data in all the MC 2 it will start hiding inside the next high priority in this example which is MC 6, let we say that the first pixel from MC_6 was in SC_1 and it's value was (17, 21, 90), it

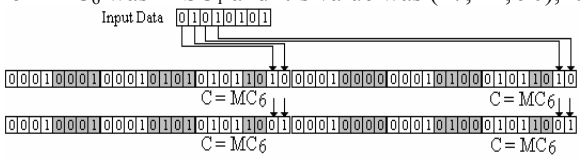


Fig. 6: Data Hiding Using MC_6 with SC_1

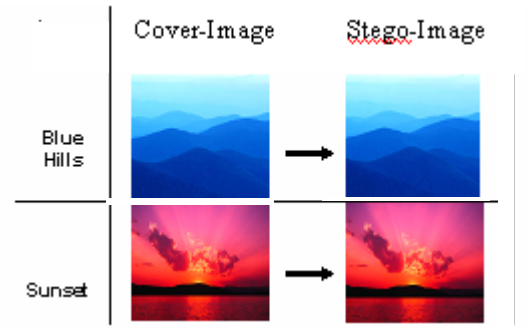


Fig. 7: Blue hills and sunset bitmap images before and after data hiding

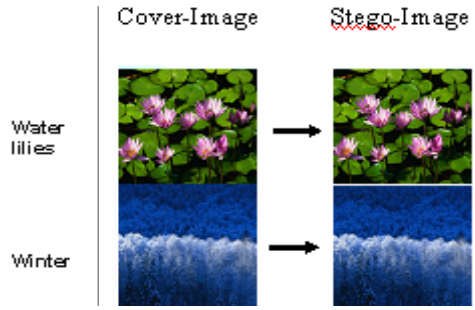


Fig. 8: Water lilies and winter bitmap images before and after data hiding

will hide the data as the following, the least 2 bit will be modified in the current selected color for the current selected pixel and in the current selected color for the next pixel, as it is shown in Fig. 6.

RESULTS AND DISCUSSION

Confidence in the present results is gained by comparing of the results obtained from the present Steganography algorithm with those previously published in the literatures[1-3, 5, 6, 9,11].

As we mention before, specifications of a good Steganography algorithms on a bitmap image are depend on the following points:

1. Comparison between the present work and the previous work.
2. A Large amount of hiding data (Avoid Statistical attack): using one cover-image as a carrier to hold a maximum number of bytes instead of a sequence of carriers.
3. Human vision scale (Avoid visual attack): Steganography message can be embedded into digital images in ways that are imperceptible to the human eye. In other words, a stego-image that is generated by the present algorithm has to be normal for human vision and cannot be detected.

Comparison with previous work: We use more than 50 BMP images to test the present algorithm and to be sure that the aim of data embedding is satisfied. In this work, it appears that the efficiency of embedding data is very high when we perform comparison with previous works. We have been shown the comparison results of four BMP images with fix size 400x400 pixels Fig. 7-8 with S-Tools algorithm[1]. The results illustrate the level of efficiency according to the following concepts:

* The amount of hidden data.
* The amount of noise detections.

Fig. 9 illustrates the amount of the hidden data on selected images from Fig. 7-8 with fix size (400x400) pixels.

It appears that the present work can hide a large amount of data and it exceeds the capability of the S-Tools. We should emphasize that the "Blue Hills" image in the Fig. 7 can hide the maximum amount of data while the "Water lilies" image in Fig. 8 is in the minimum. This variety of image's capabilities to hide bytes depends on the color distribution and the sequence of the input characters.

Figure 11 illustrate the amount of the noise on 4 images by using two kinds of noise detections, the first one is used to find the average of 4 neighbor pixels in the 3x3 pixels Fig. 10-left around the pixel P_i as in equation (15). While the second one is used to find the average of 12 neighbor pixels in the 5x5 pixels Fig. 10-right around the pixel P_i as in equation (16).

$$Noise_{3x3} = \sum_{i=1}^{mxn} \left(\left| \frac{\left(P_i^S + \sum_{j=1}^{4} P_j^S \right)}{5} - \frac{\left(P_i^C + \sum_{j=1}^{4} P_j^C \right)}{5} \right| \right) \quad (15)$$

$$Noise_{5x5} = \sum_{i=1}^{mxn} \left(\left| \frac{\left(P_i^S + \sum_{j=1}^{12} P_j^S \right)}{13} - \frac{\left(P_i^C + \sum_{j=1}^{12} P_j^C \right)}{13} \right| \right) \quad (16)$$

Where P_i^S and P_i^C are the color values of the pixel i in both stego-image and cover-image respectively, and *(mxn)* is a number of pixels in a bitmap image.

In Fig. 12 we divide a noise frequency into 21 classes, started from the minimum value (- 10) to the maximum value (+10) in the selected bitmaps. We calculate the frequency of damage pixels and then determined the corresponding class for each pixel. We perform sector comparison instead of pixel

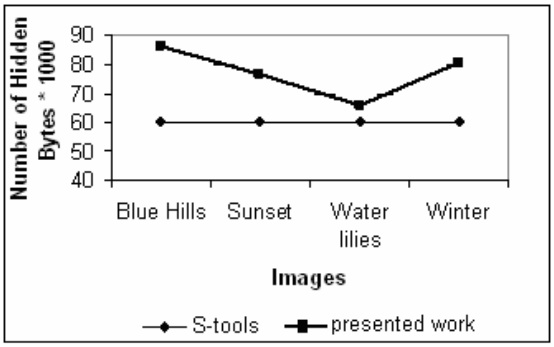


Fig. 9: The amount of hidden byte vs four different bitmap carriers

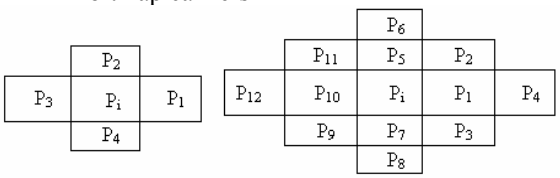


Fig. 10: Noise detections using 4 and 12 neighbor pixels

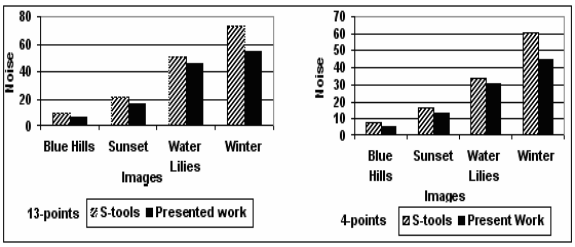


Fig. 11: Noise detection of the present work and previous works vs four deference bitmap carriers

comparison to show the smoothing of color around a specific pixel; this comparison has been implemented by using equation (16). It appears that the present work is less noise frequency than S-Tools algorithm for all images.

Avoids statistical attack: Challenge χ^2 attacks.

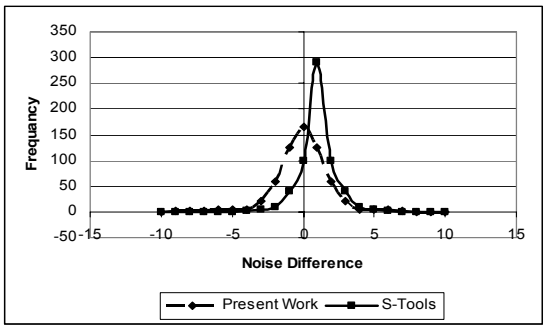
$$\chi^2_{k-1} = \sum_{i=1}^{k} \frac{(n_i - n_i')^2}{n_i'} \quad (17)$$

Where, k-1 is degree of freedom and n_i' is calculated according to the equation (18).

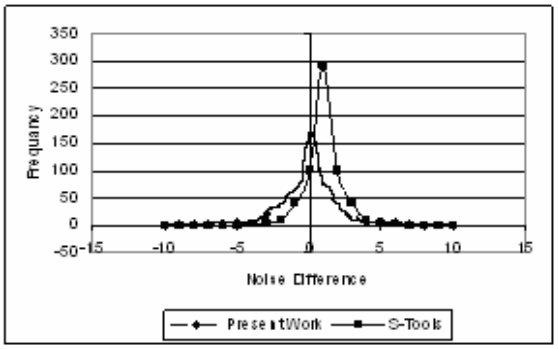
$$n_i' = \frac{\left| Frequancy\ of\ color\ accors\ in\ the\ set\ \{C_{2i}^{\{R,G,B\}}, C_{2i+1}^{\{R,G,B\}}\} \right|}{2} \quad (18)$$

Where, $\{C_{2i}^{\{R,G,B\}}, C_{2i+1}^{\{R,G,B\}}\}$ is the i-th pair of the palette colors $C_0^{\{R,G,B\}}, C_1^{\{R,G,B\}}, \ldots, C_{255}^{\{R,G,B\}}$ for each color {R,G,B}.

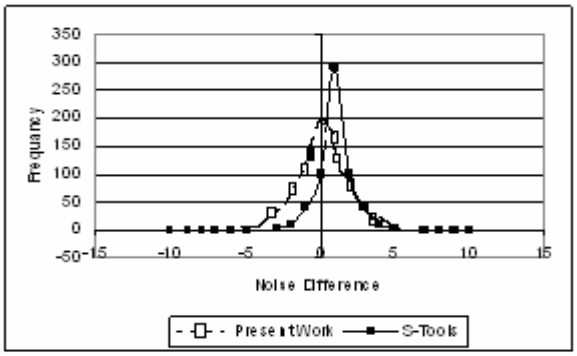
Sunset



Blue Hills



Water lilies



Winter

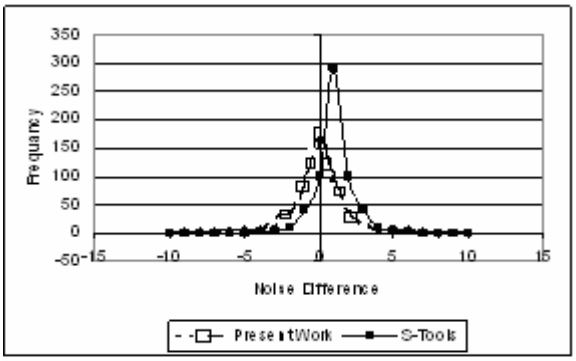
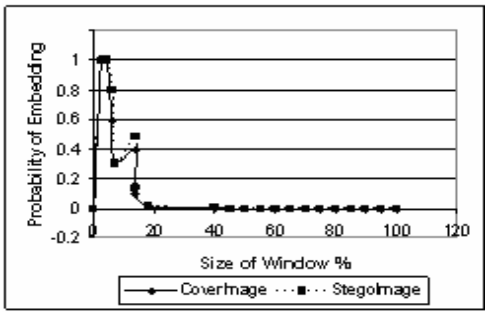
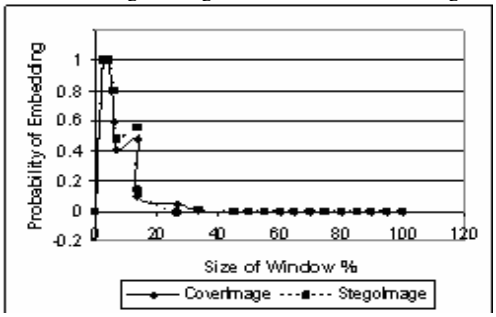


Fig. 12: Frequency of the sector noise

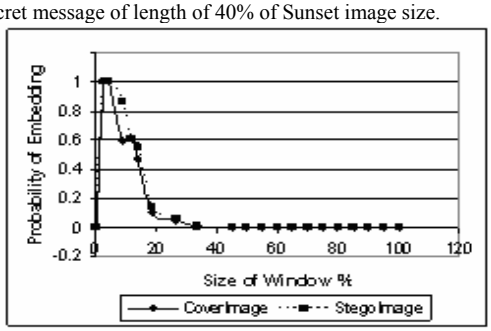
The secret message of length of 40% of Blue Hills image size.



The secret message of length of 75% of Blue Hills image size.



The secret message of length of 40% of Sunset image size.



The secret message of length of 75% of Sunset image size.

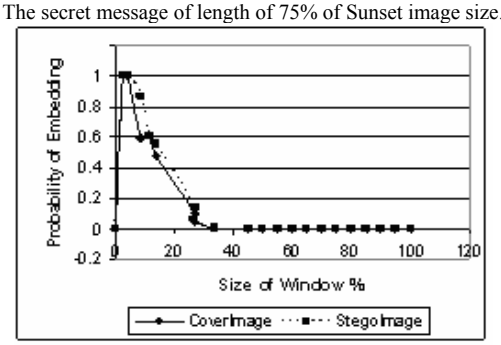
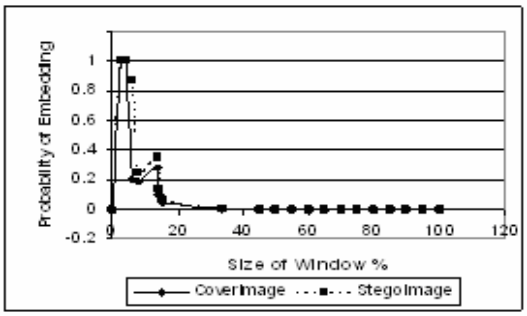
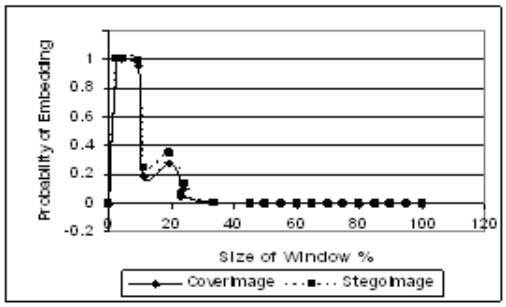


Fig. 13a:Probability of embedding for a secret message of length 40 and 75% of images (Blue Hills, Sunset) size

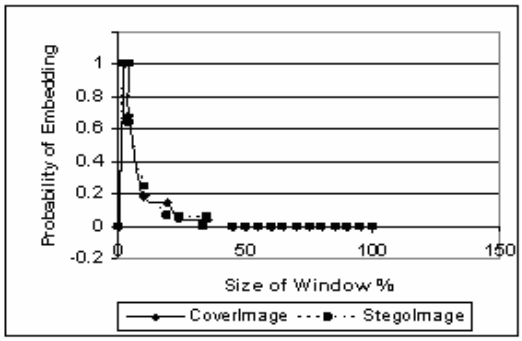
The secret message of length of 40% of Water lilies image size.



The secret message of length of 75% of Water lilies image size.



The secret message of length of 40% of Winter image size.



The secret message of length of 75% of Winter image size.

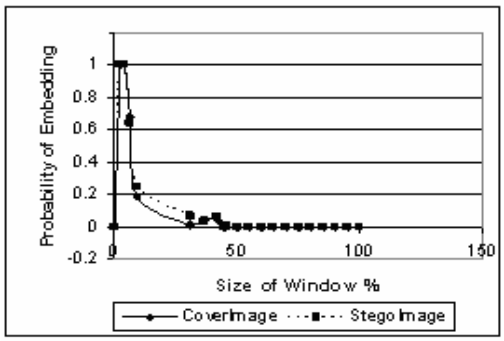
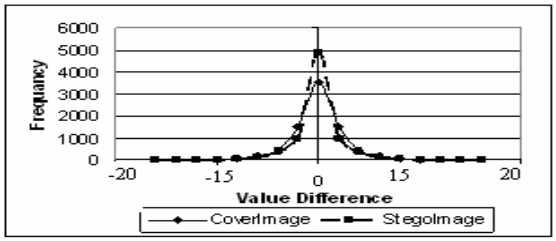


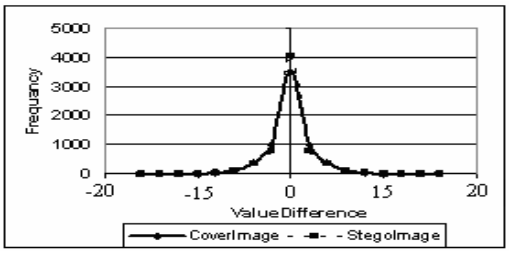
Fig. 13b: Probability of embedding for a secret message of length 40 and 75% of images (Water lilies, Winter) size

Sunset



Water lilies

Blue Hills



Winter

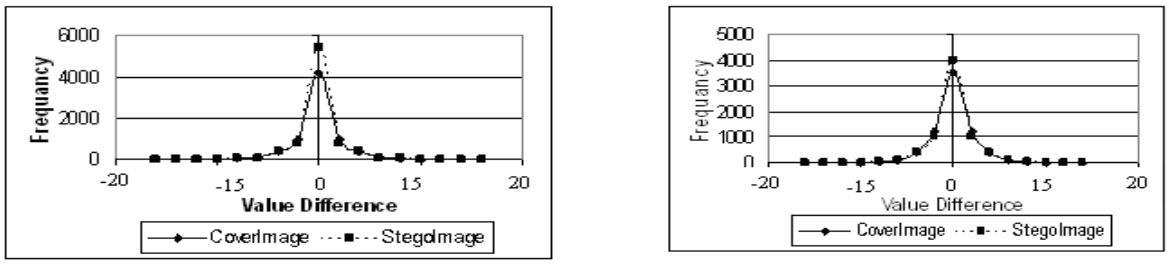


Fig. 14: Value difference on neighboring pixels for both cover and stego images

$$n_i = \left| Frequancy\ of\ color\ accors\ at\ the\ color\ C_{2i} \right| \quad (19)$$

We use equation 19 to express the probability that distributed of n_i' , n_i are equal.

$$p = 1 - \frac{1}{2^{\frac{k-1}{2}}\,\Gamma\left(\frac{k-1}{2}\right)} \int_0^{\chi^2_{k-1}} e^{\frac{x}{2}}\, x^{\frac{k-1}{2}-1}\, dx. \quad (20)$$

While the message-carrying pixels in the image are selected randomly rather than sequentially according to MCs with their SCs concepts, the chi-square test is less effective. It appears that chi-square attack on stego images with randomly scattered messages produces fluctuating *P* values in the beginning and then, as the sample size increases, the p value eventually drops to zero due to the sensitivity of the test. Figures 13a-13b show the comparison between Cover and Stego images, it appears that the present embedded algorithm produce the same behavior of the probability *P* with respect to the length of secret message (40% and 75%) of the image size and any window size using four types of images. In addition we see that the *p* value eventually drops to zero at the window sizes 20 and 35 on any image. We conclude that stegoanalysis can not be detected an embedded data due to the matching of Stego and Cover images curves.

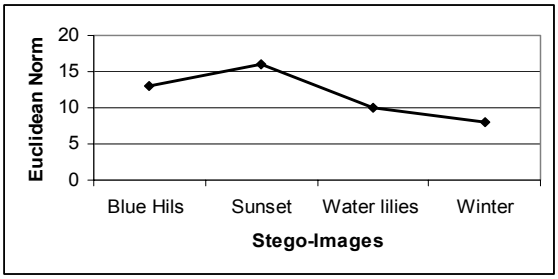


Fig. 15: Euclidean norm evaluations on four stego images

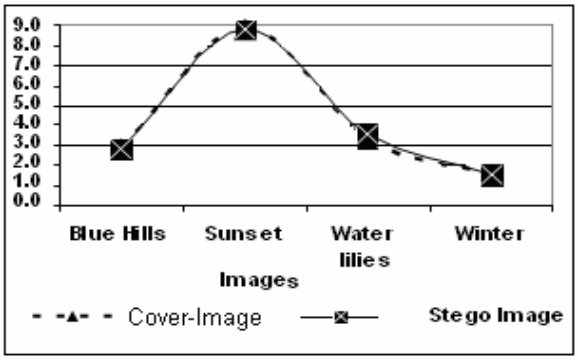


Fig. 16: The length of cover and stego images

Statistical of the value difference between neighboring pixels: In this work we calculate the difference of the horizontally neighboring pairs of the image Fig. 14 and we perform the comparison between Cover and Stego images on four types of images according to the following formula:

$$Val^{C}_{i,j} = \left| P^{C}_{i,j} - P^{C}_{i,j+1} \right|,$$

$$Val^{S}_{i,j} = \left| P^{S}_{i,j} - P^{S}_{i,j+1} \right| \quad \forall i = 1,..,(nxm) - 1 \quad (21)$$

It appears that all values differences are come near to zero for both Cover and Stego images and comparison between them are indeed remarkable similar. We conclude that the smoothing of the color is equivalent for both Stego and Cover images.

Avoids visual attack

The amount of Euclidean norm: Fig. 15 shows the amount of the Euclidean norm to compute the distance between the Cover-images and Stego-image presented at Fig. 7a-7b. This measure is used to find the set of the closest palette color (in Euclidean norm) by using the following equation:

$$d = \sqrt{(R_c - R_s)^2 + (G_c - G_s)^2 + (B_c - B_s)^2} \quad (22)$$

Where the subscript c means the cover-image and the subscript s means the stego-image.

It appears that Sunset image has a maximum norm due to the large area of uniform palette color (Red, Pink), while Winter image has a minimum norm due to

small area of the uniform palette colors (Blue or White or Black).

The amount of brightness information: Figure 16 shows the amount of the length L equation (23) of the RGB vector for both cover and stego images shown in Fig. 6-7. This measure is necessary to use against visual attack to find the brightness information.

$$L = \sqrt{R^2 + G^2 + B^2}. \quad (23)$$

CONCLUSION

In this work, we satisfy the aim that says Steganography is an effective way to obscure data and hide sensitive information. The present algorithm allows an individual to hide data inside other data with hopes that the transfer medium will be so obscure that no one would ever think to examine the contents of the file. The algorithm which is described by pseudo-code is presented and it is possible to implement a Steganography algorithm to hide a large amount of data into carrier bitmap image.

We used three layers of security to secured data by obscuring the context in which it was transferred. With continued research and an improvement in algorithms design, steganography can be taken as a serious way to hide data and the present work appears that it was more efficient than the most familiar algorithm like (S-Tools)[1]. Working against visual and statistical attacks need adaptive algorithm on each step of data embedded. It was found that the present algorithm was attractive and results reached by this algorithm were efficient in the field of data embedding (Steganography). We performed three types of comparison; the first one was used to compare the present algorithm with S-Tools algorithm through the amount of noise and the amount of size. It appears that the present work was less effective of the nose at the pixels and lager amount of embedded data. The second comparison was made upon the statistical attack; it shows that it was difficult to distinguish between Cover and Stego image when chi square and the difference between neighboring pixels were implemented. The last comparison was found that visual attack results indicate that using non uniform color was extremely powerful when we have large amount of embedded data.

REFERENCES

1. S-Tools (http:// digitalforensics. champlain. edu/ download/ s-tools 4.zip).
2. Chandramouli, R. and N. Memon, 2001. Analysis of LSB based image steganography techniques. Proc. of ICIP, Thessaloniki, Greece.
3. Dumitrescu, S., W. Xiaolin and Z. Wang, 2003. Detection of LSB steganography via sample pair analysis. In: LNCS, Vol. 2578, Springer-Verlag, New York, pp: 355–372.
4. Ahn, L.V. and N.J. Hopper, 2004. Public-key steganography. In Lecture Notes in Computer Science. Vol. 3027 / 2004 of Advances in Cryptology - EUROCRYPT 2004, pp: 323–341. Springer-Verlag Heidelberg.
5. Pang, H.H., K.L. Tan and X. Zhou, 2004. Steganographic schemes for file system and b-tree. IEEE Trans. on Knowledge and Data Engineering, 16: 701–713.
6. Dobsicek, M., 2004. Extended steganographic system. In: 8th Intl. Student Conf. on Electrical Engineering. FEE CTU.
7. Mittal, U. and N. Phamdo, 2002. Hybrid digital-analog joint source-channel codes for broadcasting and robust communications. IEEE Trans. on Info. Theory, 48: 1082 –1102.
8. Pavan, S., S. Gangadharpalli and V. Sridhar, 2005. Multivariate entropy detector based hybrid image registration algorithm. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, pp: 18-23.
9. Moulin, P. and J.A. O'Sullivan, 2003. Information-theoretic analysis of information hiding. IEEE Trans. on Info. Theory, 49: 563–593.
10. Amin, P., N. Liu and K. Subbalakshmi, 2005. Statistically secure digital image data hiding. IEEE Multimedia Signal Processing MMSP05, China.
11. Jackson, J., G. Gunsch, R. Claypoole and G. Lamont, 2004. Detecting novel steganography with an anomaly- based strategy. J. Electr. Imag., 13: 860– 870.