Algorithm for Text Hiding in Digital Image for Information Security

Ahmed Ibrahim
Irbid National University

Arwa Zabian
Jadara University

Farehan Nizar Esteteya

Amani Khalid Al padawy

Abstract
The mass diffusion of digital communication needs the special means of security. Cryptography concentrates on rendering the message unreadable to any unauthorized persons who might intercept them. In contrast, steganography is a method of concealing the existence of message to allow a secure communication in a complete undetectable manner. Digital image is the most common type of carrier used for steganography. In this paper, we have proposed a steganography mechanism, that reads the message and converts it to a colored image , that is transmitted to the authorized destination, and from which is extracted the information needed to reconstruct the original message. Our encryption approach is based on analyzing the plain text and extracting from it its component's letters, which are encrypted in the colored image. Our results show that for a message size in the range (x_1, x_2) bytes, the size of the resultant image will be fixed and consequently the encryption and decryption time are fixed.

Key words:
Text Hiding, Digital Image, Security

1. Introduction

With the development of the Internet, information processing technologies and the rapid development of communication, it is necessary to share information resources, and the network has becoming the main means of communication. Nevertheless, the Internet is an open environment so; information security has becoming increasingly important. Today, information security technology has two main branches are cryptography and information hiding. Cryptography is concerned on concealing the content of the message, so it becomes difficult to understand. Information hiding is divided into steganography and digital watermarking. Digital watermarking protects unauthorized change of the contents and assures legal user for its copyright (no encryption data embedded).

Steganography is a technique to conceal information into digital context files for purposes such as secret communication and covert channel. Steganography, originally designed for cover or hidden communications. Most existing steganographic tools can provide perceptually invisible data hiding, the stochastic visibility or unauthorized detect ability of hidden data still remains a challenging task. Stochastic visibility can be considered as the possibility of unauthorized detection to differentiate between the cover and the host data based on hypothesis testing. The steganographic system should satisfy a set of requirements; the main requirement consists in providing statistical indistinguishability between the cover data and the host data.

Information hiding can be obtained into four phases are: preliminary phase in which an encryption technique is applied. Embedded phase in which is used an algorithm for information hiding. The transmission phase and finally the extraction phase[1]. A security issue must be used in each step. Information hiding can be used in different applications include military, E- commerce, confidential communication, copyright protection, copy control, authentication, digital elections. In these fields, hiding information better than ciphering because in the former, nobody can notice that there is a message hides behind an image.

In steganography carrier medium is defined as the object that carries the hidden information. Stego-object is the resultant production of steganography that is transmitted to the destination. Stego-key is defined as the key used to extract the hidden data from the stego- object.

There is a variety of digital carriers where data can be hidden. Data may be embedded in: audio file where data can be hidden in form of echoes or slightly modification in signal amplitude, or embedded in an audio file or noise. Information can be hidden in document by manipulating the position of the lines or the words. Data can be hidden in the unused space in file headers like TCP/IP packets. Data can be hidden in design or image by manipulating the properties of image like luminescence, or contrast and colors.

Manuscript received June 5, 2009
Manuscript revised June 20, 2009

A digital image is the most common type carrier used for steganography. An image can be described as an array of numbers that represent high intensively at various point of colors are called pixel. The size of an image can be given in pixels. Pixels are indexed by x and y coordinates with x and y having integer values. Each pixel is generally stored as 24 bit or 8 bit. A 24-bit image are spread over three bytes and each bytes represents red, green, and blue respectively. Colors are obtained by mixing red, green, and blue light in different proportions.

In this paper, we propose a security mechanism that hides a message text into a digital image, by converting the text to colored image, and then the colored image is transmitted over the network to the destination. The decryption process at the destination is done in a conversely manner to the encryption process. Our proposed algorithm (TOPAZ) applies the four steps for information hiding described above that are:

The preliminary phase: in which the plain text is prepared to be encrypted. In the embedded phase the plain text is ciphered in colors in a manner that to each letter is associated a color and the resultant stego object is a colored image. In the transmission phase the previous colored image is transmitted over the network to the destination. The extraction phase consists on using an authentication method to authenticate the destination authorized to extract the information embedded in the image to reconstruct the original text.

2. Related works

Data hiding represents a class of processes used to embed data in a manner that data should be invisible. The techniques used for data hiding vary depending on the quality of data being hidden and the required invariance of those data to manipulation. Data hiding should be capable of embedding data with the following conditions: An observer does not notice the presence of the data, the embedded data should be directly encoded into the media and the data remain intact across varying data file formats. And finally, the embedded data should be immune to modifications ranging from international and intelligent attempts at removal to anticipated manipulations like (printing, scanning, analog to digital converter).

In [2], Scott explains how to hide messages in music scores each note corresponds to a letter. Scott has use "Ave Maria" code in steganographic, where he used forty tables, each of which contains 24 entries (one for each letter) each letter in the plaintext is replaced by the word that appears in the corresponding table entry and the plaintext look-like a prayer. In 1998, Reeds have deciphered the tables proposed by Scott [3]. Camouflage technique it consists on using masking algorithms for example audio masking that is a phenomenon in which one sound interferes with other sound. That means is based on the properties of the human perceptual system. In [4], Kahn used the acrostic (is a manner in which the first letter of each line form a word or a message). He puts his lover's name in the first letters of successive chapters. In addition, he tells the prisoners of war how hide messages using the dots and dashes. In [5], John Wilkins has explained how one can hide secretly a message into geometric drawing using points, lines or triangles, and he used an invisible ink to print very small dots.

Data hiding in images has some problems due to the way the Human Visual System (HVS). However, images are subject to operations ranging from transformation to filtering and lossy compression. There are many attributes of the HVS that are potential candidates for exploitation in a data hiding system including our varying sensitivity to contrast as a function of spatial frequency and the masking effect of edges (both luminance and chrominance). The HVS has low sensitivity to small changes in luminance, being able to perceive changes of no less than one part in 30 for random patterns. However, in uniform regions of an image the HVS is more sensitive to the change of the luminance approximately one part in 240. Another HVS hole is our relative insensitivity to low spatial frequencies such as continuous change in brightness across an image.

Data hiding in an image can be done by modifying the least significant bit (LSB). In [6], is proposed two techniques for information hiding based on LSB. One that replaces LSB by Pseudo-noise (PN) sequence and the second add a PN sequence to the LSB. In[7,8], another LSB data hiding method is proposed called "PatchWork" that chooses n pairs (ai , bi) of points in an image and increases the brightness of ai by one unit while simultaneously decreasing the brightness of bi . The idea on which is based PatchWork is that given two point A,B chosen randomly from an image, consider a the brightness of A and b the brightness of B so S=a-b, must be equal to zero. In addition, the standard deviation to S on its expected value is 0. Repeating the calculation of s different times must give the same results. If it varies by more than few standard deviations, that means this did not happens by chance but indicates a high degree of certainty the presence of encoding. In PatchWork a set of artificial modifications are done to encode the information in a manner that the brightness of a number of nodes chosen randomly (ai, bi) does not deviate from the expected value 0. The Patch shape used for information encoding can be rectilinear, hexagonal or random in a manner that the energy distributed in the image informally. The restriction on PatchWork algorithms is that is low embedded data rate. In[9], the authors present a data hiding mechanism that embeds information about three images in one image based on Least Significant Bit (LSB). In which the two least significant bits of each pixel in the image are modified, each one contains the result after applying the

edge detection filter, before and after gray scale level connectivity. One pixel contains information for three different images. For data embedding, the LSB of each pixel is replaced by value zero for the non edge pixel[1] , or one for edge pixel. This is done by using the logical operators. So, the LSB contains the indication for the existence of edge pixel. The extraction of the LSB can be implemented by checking the odd and even pixel values. The prior LSB is used to contain the binary edged image after pixel connectivity implementation. In this case the original image occupies the remaining six bits.

In[10],is proposed a steganographic mechanism that divides the cover image into blocks of equal size and then it is embedded the message in the edge of the block depending on the number of ones in left four bits of the pixel. The embedding process is done as follow: first is selected a set of pixels, which would be used for hiding the data. Then the gray level values of the selected pixels is modified to make them even, and that will represent 0. To represent 1, is decrementing the gray level of the appropriate pixel by one. The embedding algorithm starts by splitting each pixel into two equal parts, it counts the number of 1's in the most part, and it embedded a secret message in the least part according to the corresponding number of bits.

In [11], is proposed a technique for hiding determined set of data in an image in the following manner, the image is considered as a matrix, and the data to be hide is considered as a secret key that is a matrix of size m*n. The resultant hidden data is embedded in the previous image given a set of operations of sum and bitwise AND between the two matrices (image and secret key). The weakness of this technique is to use of binary AND and the modification must occur in the locations in which the secret key has a value of 1. That can be easily notice the modification location. In [12], is proposed a data hiding technique that given an image of size m*n can hide as many as ($\lfloor \log 2\ (mn+1) \rfloor$) bits of data in the image by changing at most 2bits in the image. This technique uses another binary operator XOR to protect the secret key from being compromised and uses in addition a weight matrix to increase the data-hiding rate while remaining high quality of the host image. Therefore, in this technique a set of XOR operations are done between the original image, the secret key, and the weight matrix to produce the final image with hidden data. In [13], Lippmann hides data in the chrominance channel of the National Television Standard Committee (NTSC) Television signal by exploiting the temporal over sampling of color in such signals. This method encodes a large amount of data but the data are lost to most recording, compression and transcoding processes. In[14], is proposed a data decomposition mechanism for data hiding in which , the message to be hide is divided into blocks and is used six images to hide these blocks.

[1] An edge is a set of connected pixels that lie on the boundary between two regions.

3. TOPAZ Algorithm

TOPAZ is a steganography mechanism that hides text in an image file (picture) to allow only the authorized destination to read the information embedded in the image. The algorithm works on two parts. The first is the authorized source (sender) and the authorized destination, where the sender performs the preliminary, encryption and transmission phase and the destination performs the decryption phase. For that the encryption and decryption phases are related to each other in a manner that the authorized destination does not require a lot of time to extract the hiding message. TOPAZ is scalable, where we can increase the size of the embedded data without increasing significantly the size of the resultant image or the encryption time.

3.1 Algorithm Description:

Our algorithm works in four phases are: the preliminary phase, the embedded phase, the transmission phase and the extraction phase.

3.1.1 Preliminary phase: in this phase, the algorithm prepares the plain text to be easy to encrypt. Where to each letter is assigned an array of size initially zero. Then a set of comparisons between the plain text and the letters are done. For each repeated letter, the size of the correspondent array is incremented by one. The image is considered as a matrix of pixels (x, y) where x takes the values from 0 to 36 (that corresponds to 24 letters and the numbers from 0-9). Y takes values from 0 to the plain text length (if the size of the message is 100 bytes, y takes values from 0-100). In the column (0, 0) is inserted the letters of the plain text corresponding to their location. So, the size of the resultant image will be $37 \times y$.

3.1.2 Embedded phase: in this phase is chosen a set of 36 colors are in the same scale of colors in RGB scale (it is possible to use different colors with different scales) to represent the letters from A to Z and the numbers from 0 to 9. The columns from x=1 to 36 are reserved to store the colors associated to the letters. In the cells x=1 to 36, y=0 is insert the letters, the order of letters in these columns can be the English alphabet order or can be inserted randomly. The encryption process starts by reading the plain text and making a set of comparisons, if the letter is presented in the plain text the correspondent column will

colored, with the correspondent color. If the letter is not represented, the column is colored in black. In this manner our resultant image includes the letters of the plain text and their locations and the correspondent colors. The colors are used as mask where reading the first column implies the existence of other information in the image. However only the authorized destination can read the plain text.

3.1.2.1 Example :

Figure. 1, represents an interface to the steganography of the following text: "hiding a text in an image is a steganography''. The text is inserted in the first text box in the figure. In the second text box is evident the letters components of the plain text and finally the image represents the encryption of letters extracted in the previous phase. In the bottom right of the figure it is evident the encryption time. The embedding process is done given the table.1.

Figure.1: Text Encryption Example3.1.2.1

Figure.2, 3 show an example (letter A) of the pseucode of the preparation and encryption phase written in C# language. Where the maximum size of the text can be embedded in the image is 100 bytes. The statement "Color PixelColor_a =" is used to assign color to char a or A and so on…… "Count_A=" is used to calculate the number of repeated value of A or a and so on…. "Int[]aa=new int[100]" is used to define an array to character a to save his position in the plain text. 100 is the size of the plain text in the example. "Comp_Pt" is a function used to prepare the text and has one parameter (array= plain text). The loop statement is used to read one character from the array of the plain text then switch to match the character and then save his position on his array.

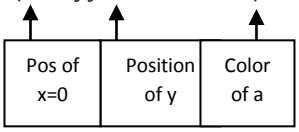
Table.1: The color representation of the letter in the example.1

letter	Color representation In RGB	Letter	Color representation In RGB	Letter	Color representation In RGB	Letter	Color representation In RGB
A	9.0.0	H	21.7.0	O	17.9.9	T	6.3.0
D	0.0.20	I	17.17.0	P	4.0.0	X	13.13.0
E	13.6.0	M	23.0.12	R	9.0.0	y	4.4.9
G	26.0.0	N	11.21.21	S	0.0.6		

```
namespace Start_Page_security_algo
{
    public partial class Encryption : Form
    {
        public Encryption()
        {
            InitializeComponent();
            textBox2.Visible = false;
            Encryptionbutton.Visible = false;
            pictureBox2.Visible = false;
        }

        int Count_A = 0, Count_B = 0, Count_C = 0;

   Color newColor_A; Color newColor_B; Color newColor_C;

//Color pixelColor;
Color pixelColora; Color pixelColorb; Color pixelColorc;

int[] aa = new int[100];

public void comp_pt(char[] a3)
        {


            for (int i = 0; i < plain.Length; i++)
                switch (a3[i])
                {
                    case 'A':
                    case 'a':
                        aa[i] = i + 1;
                        break;
                    case 'B':
                    case 'b':
                        ab[i] = i + 1;
                        break;

}

if (Count_A != 0)
                {
                    textBox2.Text = "A";
                }
```

Figure 2: the pseuocode of the preparation phase.

In figure.3, the Read function is used to assign color to each character. "Color.fromARGB" is used to assign number of color using integer values from 0-255. The encryption function is used to draw color by using (get, set) pixel in the image. "For (int yy =0)" is used to draw position of each character on the image. Image1.setpexil(0,aa[yy],newColor_A).



3.1.3 Transmission phase: in this phase the resultant image is transmitted over the network to the destination, an important issue in our work were taken in consideration is to maintain the size of the image small as possible that can be transmitted quickly and does not require a high bandwidth for the download .

3.1.4 Decryption phase: the operations done in this phase are the opposite operations done in the encryption phase. Where in the encryption phase the algorithm read the plain text as letters and to each letter is assigned a color from the RGB maintaining the letters locations. In the decryption phase, the authorized destination must do a set of authentication before arriving to the message that is an image. So, the decryption process starts by reading colors and finding the correspondent letter. If a color in the cells x=1, to 37 is exist that means the corresponding letter exist in the plain text. For determining the repetitions and the locations of the letters the algorithm reads the first column and reconstructs the original message.

Figure.4, represents the decryption process of the example 3.1.2.1. In the figure it is evident that the received message is in the form of image (first textbox), then the decryption phase generate the sequence of letters components of the text and finally the reconstruction of the original text. In the right part of the figure is evident the decryption time for level 1 and 2.

```
public void read_pt(char[] a2)
        {
            if (textBox1.Text == "")
                MessageBox.Show("Error please Insert plain text first", "",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            else
                for (int i = 0; i < plain.Length; i++)
                switch (a2[i])
                {
                    case 'A':
                    case 'a':
                        Count_A++;
                        newColor_A = Color.FromArgb(9,0,0);
                        break;
                    case 'B':
                    case 'b':
                        Count_B++;
                        newColor_B = Color.FromArgb(0,5,0);
                        break;
                    case 'C':
                    case 'c':
                        Count_C++;
                        newColor_C = Color.FromArgb(4, 4, 0);
                        break;

}
public void encreption(
{

            image1 = new Bitmap(@"C:\Documents and Settings\...."  , true);

            int x; int y;

            // Loop through the images pixels to reset color.
            for (x = 0; x < (image1.Width) ; x++)
            {
                for (y = 0; y < image1.Height; y++)
                {
                    pixelColora = image1.GetPixel(x, y);
                    image1.SetPixel((1), y, newColor_A);

                    pixelColorb = image1.GetPixel(x, y);
                    image1.SetPixel((2), y, newColor_B);

                    pixelColorc = image1.GetPixel(x, y);
                    image1.SetPixel((3), y, newColor_C);

}
int xx = image1.Width - 11;
            for (int yy = 0; yy < (image1.Height)-1;yy++)
            {
                if (Count_A > 0)
                {
                    image1.GetPixel(xx, yy);
                    image1.SetPixel(0, aa[yy], newColor_A);
                }
```

Figure 3: the pseuocode of the encryption phase

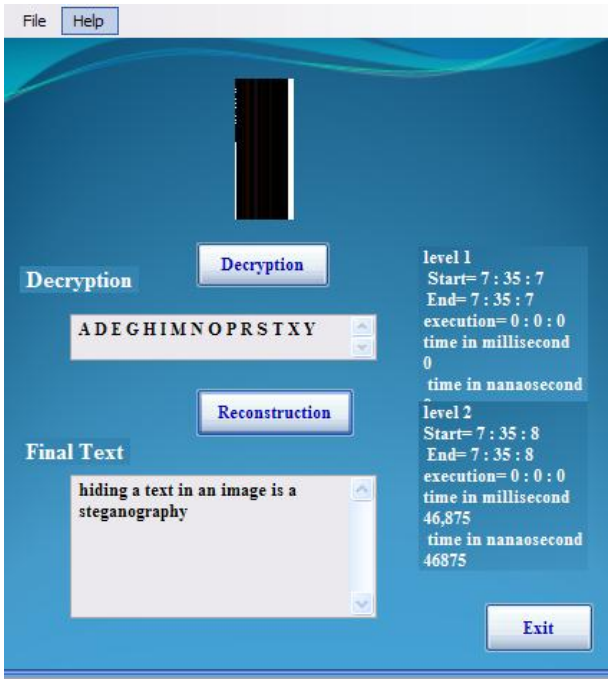


Figure 4: The decryption phase of example 3.1.2.1

4. Results

We have implemented our algorithm in C# and developed on .NET framework. To see the performance of our algorithm on different message size we have varied the size of the plaintext embedded in the image. Our results show that increasing the text size has increased the time needed for the encryption and decryption processes. However the size of the resultant image is still acceptable. Our algorithm performs better for small text size (less than 100 bytes) where we can fix the size of the resultant image and consecutively the encryption time will be fixed.

Our work were directed in two directions, the first one is to fix the size of the resultant image to 12.2 Kbytes and to vary the size of the data that can be embedded without degradation of the final text quality (reconstruction must be correct). The results presented in table 2 show that for text size vary from 87-141 bytes the size of the image is 12.2 Kbytes and the encryption and decryption time is fixed, the algorithm performance is good.

The second direction is to have a variable image size given the change in the text size. The results presented in table 3 show that the encryption and the decryption time is increased linearly with the text size and the size of the resultant image is still acceptable.

Table 2: The variation of encryption and decryption time with the text size fixing the image size

Text size (bytes)	Image size (Kbytes)	Encryption time (ms)	Decryption time (ms)
87	12.2	10	46.875
120	12.2	10	46.875
141	12.2	10	46.875

Table 3: The variation of encryption and decryption time with the text size varying the image size.

Text size (bytes)	Image size (Kbytes)	Encryption time (ms)	Decryption time (ms)
82	12.2	10	46.875
368	49.5	40	250
712	107	88.46	875
953	119	98.60	1109

5. Conclusion

Our goal in this paper is to propose a steganography mechanism that allows the hiding of a large quantity of data as possible in a colored image without increasing the size of the resultant image significantly. We have proposed an algorithm that works in phases in which first of all the text is prepared where is extracted the component letters without repetitions. To each letter is assigned an array with maximum size the size of the plain text, its value initially is zero and the size of the array is incremented for each repetition of the letter in the plain text. Then to each letter is associated a color stored in the image. Our image contains all the information needed about the plain text but this information cannot be read without having the corresponding decryption algorithm that read the colors and associate to each color a letter and then reconstruct the plain text. Our results show the efficiency of our algorithm for a range of data size where we can fix the image size and consecutively the encryption and decryption time will be fixed, and we can increase the size of data to be embedded without degradation of the final text properties. For example text size 100 bytes. However we can obtain the same performance if we fixed the text size between 100-200 bytes, and so on. But if the size of the text is variable, will vary the size of the image and both the encryption and decryption time.

Comparing our results with that presented in [10], we can find that in [10], the size of the data embedded is increased linearly with the image size. However, different images for the same size can embed different data size (increasing or decreasing the size of data) because the data hiding mechanism is depend on the image representation and different representation results in different data hiding. In our work, the image size is increased linearly with the size of the message to be hide. However the same image size can hold the same data size because the mechanism used for data hiding is not depend on the image representation but the image is a results of data hiding.

Our current and future work is to distribute the information in the image randomly where a random generator can generate the location of the letter randomly in the matrix. We have obtained very good results for the encryption phase and we still working on the decryption phase.

References

[1] F.A.P.Petitcalas, R.J.Anderson and M.G.Kuhn. "Information Hiding –A Survey " . Proceedings of the IEEE Special issue on Protection of Multimedia Content. 87(7). 1062-1078. July 1999. DOI:10.1109/5.771065

[2] G.Scott. " Schola Steganographic: in Class Octa Distributa". Jobus Hertz Printer. 1680. Bound with Gasparis.

[3] J.Reeds. "Solved : The Ciphers in Book III of Trithemius Steganographic" cryptologia. Vol XXII. No.4. 291-317. OCT 1998. ISSN 0161-1194

[4] D.Kahn. "The Code Breakers- The Story of Secret Writing". New York U.S.A: Scrinber. 1996.ISBN 0-684-83130-9

[5] J.Wilkins. "Mercury: or The Secret and Swift Messenger: Showing how a man may with privacy and speed communicate his thoughts to a friend at any distance". London: printed for rich Baldwin near the oxford- Arms in Warnick-Lane. 2nd ed. 1694 Cambridge University Library.

[6] R.G.Van Schyndel, A.Z.Tirkel, and C.F.Osborne. "A Digital Watermark". In Proceedings of IEEE International conference in image processing 1994. Vol II (Austin,TX). 86-90. DOI: 10.1109/ICIP 1994.413536.

[7] W.Bender, D.Gruhl and N.Morimoto, A.Lu. "Techniques for Data Hiding" Tech.Report. Mit Media Lab.1994.

[8] W.Bender, D.Gruhl, N.Morimoto, A.Lu. "Techniques for Data Hiding". IBM Systems Journal. Vol 35, issue 3-4, 1996. 313-336. ISSN: 0018-8670.

[9] R.H.Alwan, F.J.Kadhim, A.T.Al-Taani. "Data Embedding Based on Better Use of Bits in Image Pixels". International Journal of Signal Processing. Volume. 2, Number. 1, 2005. pp: 104-107. ISSN: 1304-4494.

[10] A.T.Al-Taani, A.M. Al-Issa. " A Novel Steganographic Method for Gray-Level Images". International Journal of computer, Information, and systems Science and Engeering.3:1.2009.pp:5-10 ISSN:2070-3732

[11] Y.Yuan Chen, H.Kuang Pan, and Y.Chee-Tseng. "A Secure Data Hiding Scheme for Two color Images". in Proceedings of the Fifth IEEE Symposium on Computer and Communication ISCC 2000. Page 750. ISBN: 0-7695-0722-0.

[12] M.Y.Wu ad J.H.Lee. "A Novel Data embedding Method for Two Color Facsimile Images". In proceedings of International Symposium on Multimedia Information Processing. Chung-Li Taiwan, R.O.C. 14-16 December 1998.

[13] S.F.William, A.B.Lippman, A.H.Edward, and N.NAran. " A Receiver Compatible Enhanced Definition Television System". U.S.Patent No: Wo/1990/009081.09-08-1990.

[14] X.Zhang, S.Wang and W. Zhang. "Steganography Combining Data Decomposition Mechanism and Stego-coding Method. Informatica 33(1)-2009. pp :41-48. ISSN:1854-3871.