A New Steganography Approach for Image Encryption Exchange by Using the Least Significant Bit Insertion

Mohammad Ali Bani Younes 1† and Aman Jantan 2††,

School of Computer Sciences, Universiti Sains Malaysia, Penang, 11800, Malaysia

Summary
A new steganography approach for data hiding is proposed. This approach uses the Least Significant Bits (LSB) insertion to hide data within encrypted image data. The binary representation of the hidden data is used to overwrite the LSB of each byte within the encrypted image randomly. Experimental results show that the correlation and entropy values of the encrypted image before the insertion are similar to the values of correlation and entropy after the insertion. Since the correlation and entropy have not changed, the method offers a good concealment for data in the encrypted image, and reduces the chance of the encrypted image being detected. The hidden data will be used to enable the receiver to reconstruct the same secret transformation table after extracting it and hence the original image can be reproduced by the inverse of the transformation and encryption processes.

Key words:
Image encryption, Image correlation, Decryption, Steganograph, Least Significant Bits.

1. Introduction

Digital images are being exchanged over various types of networks. With the huge growth of computer networks and the latest advances in digital technologies, a huge amount of digital data is being exchanged over various types of networks. It is often true that a large part of this information is confidential, private or both, which increase the demand for stronger encryption techniques [3]. Encryption and steganography are the preferred techniques for protecting the transmitted data [7], as a result, there are various encryption systems to encrypt and decrypt image data, and it can be argued that there is no single encryption algorithm satisfies the different image types [2], [11]. Data exchange is a good example of an application that uses encryption to maintain data confidentiality between the sender and the receiver. In this paper, *steganography* is used to hide information to perform encryption. Steganography techniques are getting significantly more sophisticated and have been widely used. The Steganography techniques are the perfect supplement for encryption that allows a user to hide large amounts of information within an image. Thus, it is often used in conjunction with cryptography so that the information is doubly protected; first it is encrypted and then hidden so that an adversary has to first find the hidden information before decryption take place [4],[6],[8]. The problem with cryptography is that the encrypted message is obvious. This means that anyone who observes an encrypted message in transit can reasonably assume that the sender of the message does not want it to be read by casual observers. This makes it possible to deduce the valuable information. Thus, if the sensitive information will be transmitted over unsecured channel such as the internet, steganography technique can be used to provide an additional protection on a secret message [14]. When hiding information inside images the LSB (Least Significant Bit) method is usually used. While the cryptography tries to convert an image to another one that is hard to understand, steganography involves hiding information so it appears that no information is hidden at all. Therefore, the person will not attempt to decrypt the information [11]. For example, an alteration of the least significant digit for the color value of some pixels in an image will not affect the quality of the image and thus, enabling a message to be sent within an image using these bits [15]. In this paper, steganography technique will be used to send the secret information along with an encrypted image. A number of horizontal and vertical blocks at the sender side will be generated, and then mixed with the encrypted image before transmitting it to the receiver. The receiver will need this information to reconstruct the same secret transformation table after extracting the secret information from the encrypted image. Instead of sending the whole secret transformation table, which is usually big, only the secret information is sent. In this approach, the binary representation of the hidden data is used to overwrite the LSB of each byte within the encrypted image randomly. This method will be expected to spread hidden information within encrypted image data randomly based on the secret key before transmission. The values of the correlation and entropy before and after the insertion process are expected to be the same. Thus, it will be used to reduce the chance of the encrypted image being detected and then enhance the security level of the encrypted images. Furthermore, this information will be used to enable the receiver to reconstruct the same secret

Manuscript received June 5, 2008
Manuscript revised June 20, 2008

transformation table after extracting hidden data and hence the original image can be reproduced by the inverse of the transformation and encryption processes. The rest of this paper is organized as follows. Section 2 gives a background about the current image steganography schemes. In Section 3, the description of the proposed steganography algorithms is presented. Section 4 presents the experimental results and discussion. Finally, Section 5 concludes the paper.

2. Background

Besides cryptography, steganography can be employed to secure information. Steganography is a technique of hiding information in digital media. Cryptography and steganography techniques of digital images are widely used to prevent and frustrate opponents' attacks from unauthorized access. Thus, reliable image encryption techniques are of utmost importance for the protection of data from counterfeiting, tampering, and unauthorized access [9]. While cryptography and steganography are related, there is a difference between the two. Cryptography is used to scramble messages so that they cannot be understood. It does not hide the fact that the message exists. On the other hand, steganography conceals the fact that the message exists by hiding the actual message in another [12]. There are many different steganographic methods that have been proposed over the last few years. Most of them are simple techniques that can be broken by careful analysis of statistical properties of the channel's noise [5]. Recently, several steganographic algorithms for two-color binary images have been proposed [4]. Example of steganography can be found in invisible inks, microdots, character arrangement, digital signatures, covert channels, and spread spectrum communications Johnson and Sushil**.** [13] explain steganography and provide a brief history, describe how steganography is applied in hiding information in images, and survey a few steganography software applications for processing images. Tsung-Yuan and Wen-Hsiang. [10] proposed a new steganographic method for data hiding in Microsoft Word documents by a change tracking technique. Sinha and Singh. [1] proposed a technique to encrypt an image for secure transmission using the digital signature of the image. Digital signatures enable the recipient of a message to authenticate the sender of a message and verify that the message is intact. Kisik et al. [4] proposed a steganographic algorithm which embeds a secret message into bitmap images and palette-based images; the proposed algorithm divides a bitmap image into bit-plane images from LSB-plane to MSB-plane for each pixel, and considers each bit-plane image as a binary one. The description of the proposed algorithms are presented below, the first one will be used to generate a mixed data before transmission to the receiver, and the second will be conducted by the receiver to extract the secret information from the encrypted image before decryption take place and thus obtaining the original image.

3. The Proposed Technique

We apply this method to exchange the information between the sender and the target receiver. This method is applied simply to hide information in the encrypted image and then send the encrypted image to the intended receiver to extract the hidden information to be used for reproducing the same secret transformation table, and then enable the receiver to recover the original image precisely. This method is used to be as a supplement encryption process to prevent the unauthorized access from being detected. The following objectives are the description of the parts of the steganography process; the secret key, the encrypted image represents the transformed image after encryption and the numbers of horizontal and vertical blocks represent the hidden information. The result is the mixed data; encrypted image that includes the mixing data. In this process, we will apply these objectives in order to investigate the effectiveness of the proposed method in the encrypted images with and without this method. Fig.1 and Fig.2 represent an overview model of the proposed technique of the sender's and the receiver's side respectivily.

3.1 Data Mixing (Sender side)

The initial information prepared by the sender (a number of horizontal and vertical blocks) is then mixed with the encrypted image prior to transmission. The insertion positions of this information will be randomly selected depending on the key. This enables the user who knows the key to extract the information. The following algorithm explains how this information will be mixed with the ciphered image data.

ALGORITHM MIXING_DATA

1: Load ciphered image
2: Input key
3: Convert HNB into HNB_bitArray
4: Convert VNB into VNB_bitArray
5: seed = |Hash Function (Key)|
6: Randomize using seed
   /* Replace bits of the HNB_bitArray with least
   significant bits of randomly selected pixels */
7: For i = 0 to length of HNB_bitArray – 1
   7.1. r = rnd // r between 1 and # of image pixels -1
   7.2. least bit of pixel r = HNB_bitArray(i)

/* Replacement of the i^{th} bit of HNB_bitArray with the least significant bit of the r^{th} pixel of the image */
/* Replace bits of the VNB_bitArray with least significant bits of randomly selected pixels */
8: For i = 0 to length of VNB_bitArray – 1
8.1. r = rnd // r between 1 and # of image pixels -1
8.2. least bit of pixel r = VNB_bitArray(i)
/* Replacement of the i^{th} bit of VNB_bitArray with the least significant bit of the r^{th} pixel of the image */
END MIXING_DATA
Input: Ciphered image, Key
Output: Mixed data

For Example, assume that the binary stream in Fig.3 is part of a ciphered image data of a (*28* pixels). Assume also that the horizontal number of blocks (HNB) is *150* and the vertical number of blocks (VNB) is *100*, then the stream "**001**0010110 & 0001100100", the binary representation of 150 and 100 respectively, will be hidden in the ciphered image data. Each of those bits will replace the LSB of a pixel value. Those pixels will be selected randomly depending on the key**.** Fig.4 shows the same part of the image data with the least significant bits of pixels number *6*, *17*, and 27 changed with the first *3* bits of the stream data that represent the number of blocks. Observe that the least significant bit of pixel *6* has been changed from 1 to 0; the same happened to pixel *17*; while the least significant bit of pixel *27* doesn't changed since its value and the new value are the same.
The bits in **bold** in Fig.3 and Fig. 4 are the ones that will be impacted by the insertion process. The change in these bits is small will be unclear and unrecognizable by the human eye.

3.2 Data Extracting (Receiver side)

On the other side, before the decryption of the ciphered image take place, the mixed data should be extracted from the ciphered one. The number of blocks (horizontally and vertically) were mixed with the ciphered image should be extracted as explained in the following algorithm. A user knows the key is the only one who can extract this information since the insertion positions depend on the secret key.

ALGORITHM SPLITTING_DATA
1: Load mixed data
2: Input key
3: seed = |Hash Function (Key)|
4: Randomize using seed
/* Extract bits of the HNB_bitArray */
5: For i = 0 to length of HNB_bitArray – 1
5.1. r = rnd // r between 1 and # of image pixels -1
5.2. HNB_bitArray(i) = least bit of pixel r /* Extract the i^{th} bit of HNB_bitArray from the least significant bit of the $(r)^{th}$ pixel of the image */
6: Convert HNB_bitArray into HNB
/* Extract bits of the VNB_bitArray */
7: For i = 0 to length of VNB_bitArray – 1
7.1. r = rnd // r between 1 and # of image pixels -1
7.2. VNB_bitArray(i) = least bit of pixel r
/* Extract the i^{th} bit of VNB_bitArray from the least significant bit of the $(r)^{th}$ pixel of the image */
8: Convert VNB_bitArray into VNB
END SPLITTING _DATA
Input: Mixed data, Key
Output: Ciphered image, HNB, VNB

4. Experimental Results and Discussion

The method used to evaluate the proposed technique is described in Fig.1. The algorithm was applied on a bit mapped (bmp) image that has the size of 300 pixels ×300 pixels with 256 colors. To evaluate the impact of the insertion process on the encrypted images, three different cases were tested. The number of blocks and the binary code for each case are shown in Table 1.

4.1 Data Mixing and Extracting

The horizontal number of blocks and the vertical number of blocks is 1024 blocks. Therefore, the number of bits that need to be sent within the encrypted image will be 20 bits; 10 bits for horizontal number of blocks and 10 bits for vertical number of blocks. These 20 bits will be inserted in the image data randomly based on the secret key by using the LSB insertion. Test results of the encrypted images with and without insertion position of data are shown in Figure 5. The correlation and entropy results of all cases before and after the insertion process are summarized in Table 2. Table 2 shows that the values of the correlation and entropy before and after the insertion process are the same. This means that the correlation and entropy are not affected with insertion data.

Case 1: image1
In case1 we used a 8-byte key (8ualmpur). In this case, NHB = 30 and the VNB = 30. The results of insertion process for case1 image1 are presented in Table 3. Table 3 and Fig. 6 show that the highlighted fifteen pixels still maintain the same color values as were before the insertion.

Case 2: image2
In case2 we used a 16-byte key *(softhma5ins9ecur*). In this case the HNB = 60 and the VNB = 60. The results of the insertion process for case2 are presented in Table 4.

Table 4 and Fig. 7 show that the highlighted seven pixels still maintain the same color values as were before insertion process. This means that this insertion process is simple and strong and then offered a good concealment for the data within the encrypted image.

Case 3: image3

In case2 we used a 32-byte key *(4yisbecxomingrandmoreimportant3k*). In this case the HNB = 100 and the VNB = 100. The results of the insertion process for case3 are presented in Table 5.

Table 5 and Fig. 6 show that the highlighted nine pixels still have the same color values as were before insertion.

5. Conclusion

A steganography method is proposed to embed information within an encrypted image data randomly. This method will be expected to spread hidden information within encrypted image data randomly based on the secret key before transmission. Thus, this information appears to be nothing out of the usual and should be available to the receiver to rebuild the same secret transformation table, which is needed to rebuild the transformed image, and then recover the original image. The insertion positions of this information will be randomly selected. Experimental results of this method show that the correlation and entropy values before and after insertion process were the same, offering a simple and strong way to conceal the data in the encrypted image. Thus, it will be used to reduce the chance of the encrypted image being detected and then enhance the security level of the encrypted images.

6. Tables, Figures

6.1 Tables and Figures

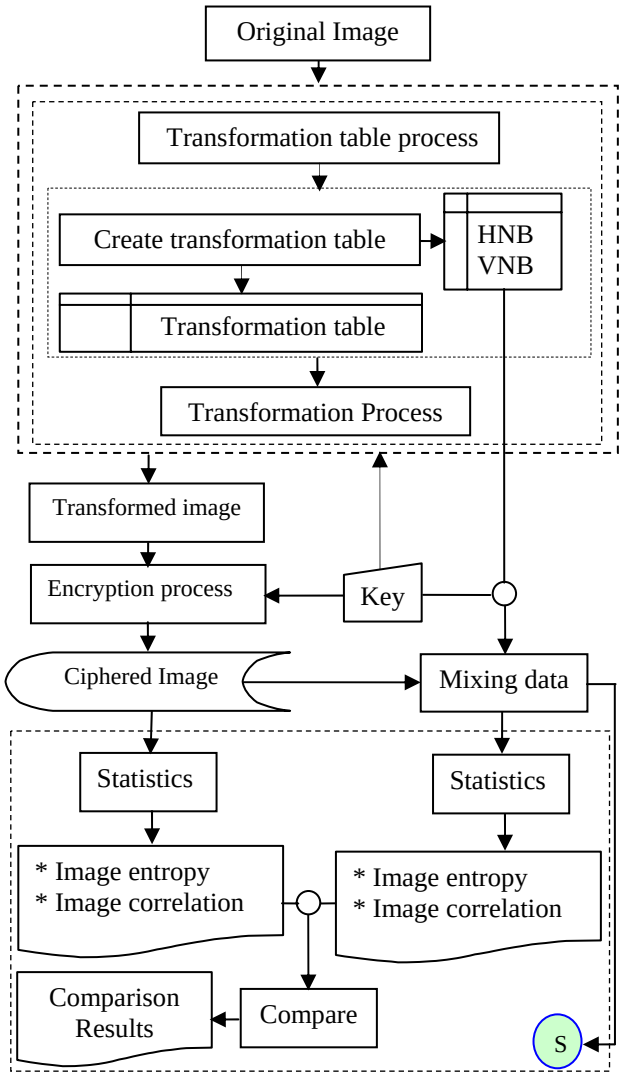

Fig. 1 An overview model of the proposed technique at the sender side

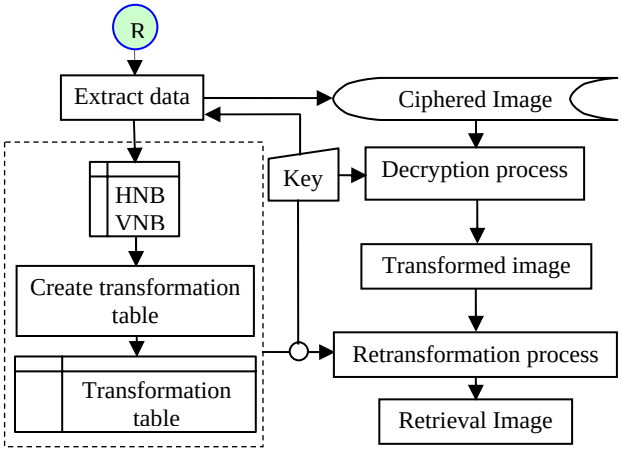

Fig. 2 An overview model of the proposed technique at the receiver side

11000111110011111111111000111101
11111111110011111000111111000000
00000000111001111111111111101000
11111010**1**00000111000111111000000
11000101010111100110100000110000
10011111001111111000111101000111
01111101100111110010001**1**11010000

Fig. 3 Part of a ciphered image data without insertion

11000111110011111111111000111101
11111111110011111**0**000111111000000
00000000111001111111111111101000
11111010**0**00000111000111111000000
11000101010111100110100000110000
10011111001111111000111101000111
01111101100111110010001**1**11010000

Fig. 4 Part of a ciphered image data with insertion

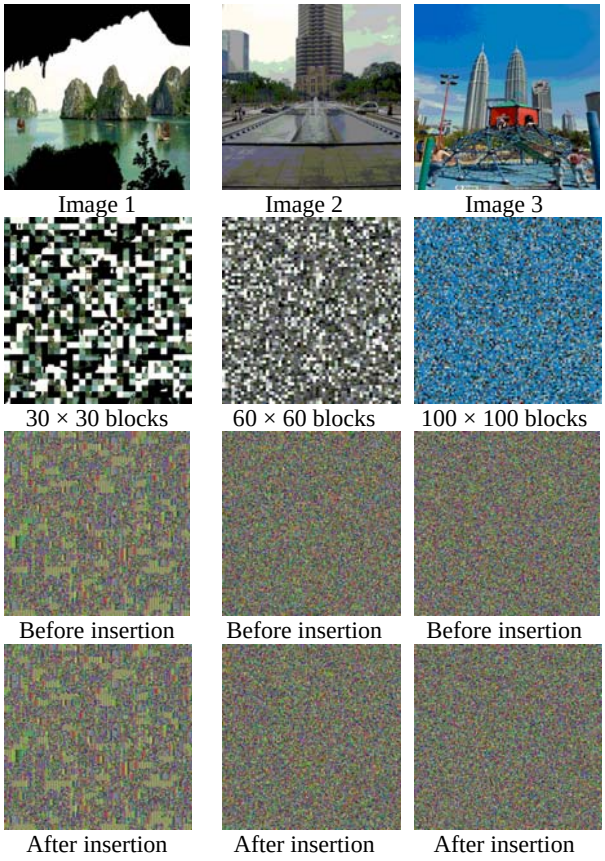

Fig. 5 Results of encrypted images with and without insertion

Table 1: Different cases to test the impact of the insertion data of the encrypted images

Cases	*Key*	*HNB*	*VNB*	*Binary representation*	
				HNB	*VNB*
Image1	8byte	30	30	0000011110	0000011110
Image2	16byte	60	60	0000111100	0000111100
Image3	32bye	100	100	0001001001	0001001001

Table 2 Results of correlation and entropy with and without insertion data

Encrypted Images		*Number of Blocks*	*Correlation*	*Entropy*
Image1	Before insertion	30×30	0.0566	5.2261
	After insertion		0.0566	5.2261
Image2	Before insertion	60×60	0.0032	5.5415
	After insertion		0.0032	5.5415
Image3	Before insertion	100×100	0.0018	5.5417
	After insertion		0.0018	5.5417

Table 3: Results of insertion positions of case1

Insertion positions	*Pixels color before insertion*	*Pixels color after insertion*
72229	146	146
6556	122	122
76040	85	84
64569	195	194
64917	166	166
25438	35	35
51057	209	209
67030	85	85
27500	87	87
70299	100	100
51146	76	76
45253	119	118
89864	6	6
4606	189	188
86528	218	218
86696	197	197
85264	225	225
41719	9	9
17749	81	81
7848	39	38

Table 4: Results of insertion positions of case2

Insertion positions	*Pixels color before insertion*	*Pixels color after insertion*
36968	243	242
30458	86	86
13729	81	80
32070	11	10
44455	185	185
69455	96	97
4862	182	183
26790	132	133
88444	121	120
55548	128	128
74907	91	90
24186	185	184
36160	11	10
55186	160	160
30138	117	117
48566	35	35
45306	206	207
73384	194	195
41132	179	178
21318	128	128

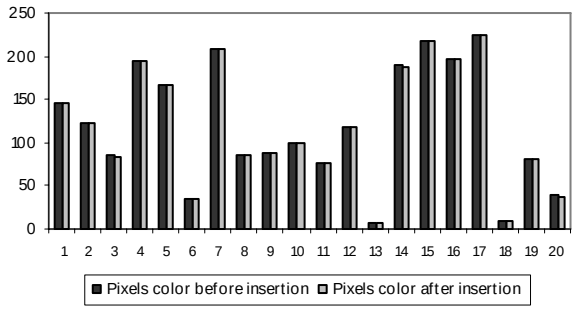

Fig. 6 Color level before and after insertion of case 1

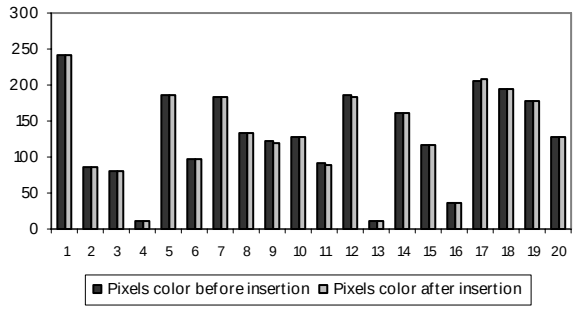

Fig. 7 Color level before and after insertion of case2

Table 5: Results of insertion positions of case3

Insertion positions	*Pixels color before insertion*	*Pixels color after insertion*
61642	30	30
31753	254	254
48268	3	2
10057	206	207
85067	28	29
85779	177	176
75791	122	122
11179	177	177
33118	177	176
49869	11	10
61289	17	16
83889	179	178
19885	200	200
37384	61	61
7797	49	49
89697	71	70
38955	217	216
73182	250	251
3873	218	218
83564	202	202

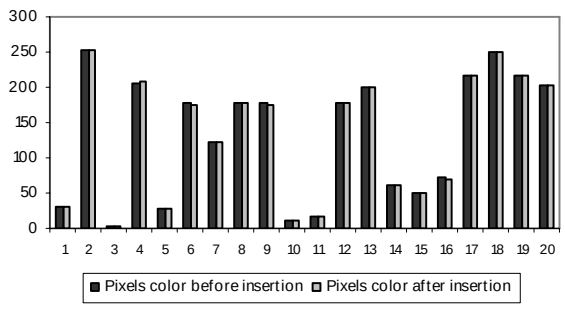

Fig. 8 Color level before and after insertion of case3

References

[1] A. Sinha, K. Singh, "A technique for image encryption using digital signature," Source: Optics Communications, vol.218, no. 4, 2003, pp.229-234.

[2] A.F. Al-Husainy. Mohammed, "Image Encryption Using Genetic Algorithm," Journal of Information Technology, Vol. 5, No. 3, 2006, PP.516-519.

[3] MAB. Younes, A. Jantan, "Image Encryption Using Block-Based Transformation Algorithm," IAENG International Journal of Computer Science, Vol. 35, Issue. 1, 2008, pp.15-23.

[4] EE. Kisik Chang, J. Changho, L. Sangjin, "High Quality Perceptual Steganographic Techniques," Springer, Vol. 2939, 2004, pp.518-531.

[5] Elke, Fraz, "Steganography preserving statistical properties," proceeding of the 5th internationally Workshop on information Hiding, Noordwijkerhout, The Netherlands, October 2002, LNCS 2578, Springer 2003, pp. 278-294.

[6] G. C. Kessler, "Steganography: Hiding Data Within Data," An edited version of this paper with the title "Hiding Data in Data," originally appeared in the April 2002 issue of *Windows & .NET Magazine*. September 2001.

[7] H. El-din H. Ahmed, M. K. Hamdy, and O. S. Farag Allah, "Encryption quality analysis of the RC5 block cipher algorithm for digital images," Optical Engineering, Vol. 45, Issue 10107003, 2006, (7 pages).

[8] H. Kathryn, "A Java Steganography Tool," March 24, 2005. http://diit.sourceforge.net/files/Proposal.pdf

[9] H. Zenon, V. Sviatoslav, R.Yuriy, "Cryptography and steganography of video information in modern communications," 1998, Citeseer.ist.psu.edu/hrytskiv98cryptography.html

[10] L. Tsung-Yuan, T. Wen-Hsiang ,"A New Steganographic Method for Data Hiding in Microsoft Word Documents by a Change Tracking Technique," Information Forensics and Security, IEEE Transactions on Vol. 2, Issue 1, March 2007 Page(s):24 – 30, Digital Object Identifier 10.1109/TIFS.2006.890310.

[11] Li. Shujun, X. Zheng, "Cryptanalysis of a chaotic image encryption method," Inst. of Image Process. Xi'an Jiaotong Univ., Shaanxi, This paper appears in: Circuits and Systems, ISCAS 2002. IEEE International Symposium on Publication Date: 2002, Vol. 2, 2002, pp. 708,711.

[12] M.M. Amin, M. Salleh, S. Ibrahim, M.R Katmin, M.Z.I. Shamsuddin, "Information hiding using steganography," Telecommunication Technology, 2003. NCTT 2003 Proceedings. 4th National Conference on Vol, Issue, 14-15 Jan, 2003, Pp: 21-25 Digital Object Identifier.

[13] N.F. Johnson, J. Suhil, " Exploring Steganography:Seeing the Unseen," Computing practices, 2006, 2006, http://www.jjtc.com/pub/r2026.pdf

[14] N.F. Johnson, J. Suhil, " Exploring Steganography:Seeing the Unseen," Computing practices, 2006, 2006, http://www.jjtc.com/pub/r2026.pdf

[15] S. Stefan, "Steganographic Systems," CSC/MAT 494, .http://www.nku.edu/~mcsc/mat494/uploads/StanevPaper.pdf

[16] W. Stallings, "Cryptography and Network Security," Principles and Practice, Upper Saddle River, NJ: Prentice Hall, 2006, pp. 24-30, 259-262.

Mohammad Ali Bani Younes received BSc in Computer Science from Yarmouk University in Irbid Jordan, MSc from Sudan University of Science and Technology in 1986 and 2003 respectively and currently enrolled in the PhD program in Computer Science in the Universiti Sains Malaysia.

Dr. Aman Jantan is a senior lecturer at the School of Computer Sciences, Universiti Sains Malaysia. He received the BComp.Sc (Hons.) and M.Sc. degree in Computer Science from the Universiti Sains Malaysia in 1993 and 1996 respectively. He obtained his PhD in Computer Science in 2002 from the same university. His research interests are in the fields of AI, Computer and Network Security, E-commerce/web intelligence, Compilers design and development techniques.