International Journal of Security and Its Applications
Vol.8, No.4 (2014), pp.25-36
http://dx.doi.org/10.14257/ijsia.2014.8.4.03

Protecting Seaport Communication System by Steganography Based Procedures

Yair Wiseman

Computer Science Department
Bar-Ilan University
Ramat-Gan 52900
Israel
wiseman@cs.biu.ac.il

Abstract

Rival companies compete against each other in business and therefore they try to obtain data that is usually held in the hands of seaports. If a company succeeds to damage one of its rival companies, the company will increase in value. Therefore there is risk for any data in the hands of a port; in spite of this, there is no cybersecurity standard for the maritime industry or for enforcement agencies; despite the fact that the need of secure communication is indispensable. Still, using encrypted messages have a noteworthy disadvantage. Each person who sees the message recognizes that there is a cryptic message. This paper introduces a technique of Steganography, that is to say - transmitting encrypted messages in images compressed by the well known JPEG format. In point of fact, the system modifies the pictures a bit, but this modification is totally unnoticeable. The image seems to be an immaculate picture, while truly the picture does include some extra information.

Keywords: *We would like to encourage you to list your keywords in this section*

1. Introduction

There is threat for any data in the hands of a port from vessel manifests to vessel traffic positions, particularly if it is used to perform automated tasks such as payroll, electronic alerts, bank deposits or infrastructure operations including locking gates or turning on floodlights. There is also threat for any computer-controlled functions of the port's lifelines such as water, sewer, fire or other emergency reactions. In addition, ports share data with authorities and the private sector. Storing or sending another part's data creates additional threats and have a need of an extra attention. There are no cybersecurity standards for the maritime industry or for enforcement agencies; however, there are some efforts to develop a uniform overall security system for the multifaceted connection of seaport security subsystems, *e. g.*, [1, 2].

One of security main issues is encrypting messages in order to pass a financial data [3]. Our aim is showing how such a message can be smuggled in a JPEG image. This smuggling is commonly called Steganography [4, 5]. Hiding a message can be very efficient in a financial or a martial community. A third person, who listens to the transmission, can think that just an innocent picture was transfer. This third person will not even try to break the code, because he will think there is no secret message in this message.

JPEG [6, 7] images are saved as sets of frequencies. These frequencies are not accurate. They use to be saved as approximate values. Actually, some numbers divide the values. Then, the values are rounded, so the values can be saved in a less space. This approximation gives JPEG the ability of compressing images efficiently. These inaccurate values are very similar to the original values. It is very hard to notice where did JPEG change the original data. Taking one more step is using the least significant bits of the frequency values in order to save an encrypted message. The sequence values will be changed slightly, but again this change will be very hard to be distinguished.

The new sent message will be most likely much longer. Apparently, this is a disadvantage of this method [8]. However, such a cost may be worth, if being hidden is very important for the message. Nowadays, when communication lines become much faster, this disadvantage will become much less significant [9]. Downloading JPEG images from the web is a very common action. We can see that such an action does not consume a long time; so sending a JPEG image will not be a deficiency [10].

2. Background for JPEG Images

The first step transforms the image color into a suitable color space. There are several methods to transform the image into a color space [11, 12]. The most common methods are the split into YUV components [13] or the split into RGB components [14]. These components are interleaved together within the compressed data. The ratio between these components is usually not one to one. When YUV components are used, usually the Y component will have a four times weight. The human eye is less sensitive to the frequency of chrominance information than to the frequency of luminance information which is represented by the Y component in the YUV format. Hence, the Y component gets a higher weight [15].

JPEG employs Chroma subsampling which is a technique of encoding images by using less resolution for chrominance information than for luminance information, taking advantage of the human eye's lower sensitiveness for color differences than for luminance differences. JPEG supports the obvious 4:1:1 chroma subsampling which denotes the color resolution is quartered, compared to the luminance information i.e. for each sampled element as in Figure 1, there is 4 numbers for luminance and just one number for chrominance; however the default chroma subsampling of JPEG is 4:2:0 which denotes the horizontal sampling is doubled compared to 4:1:1, but as the U and V components are only sampled on each alternate line.

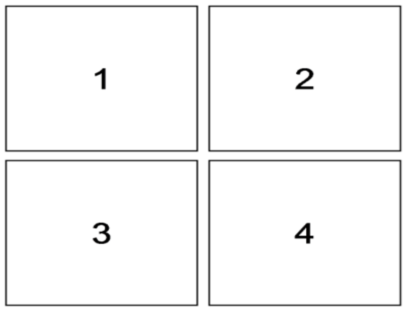

Figure 1. One JPEG Sampled Element

JPEG allows samples of 8 bits or 12 bits. In the original JPEG all values contained by the same source image must have the same precision. The values are shifted from unsigned integers with range $[0, 2^{p-1}]$ to signed integers with range $[-2^{p-1}, 2^{p-1}-1]$, by reducing 2^{p-1} from the original values, where p can be either 8 or 12. These biased values are then sent to the next step.

The second step groups the pixels into blocks of 8X8 pixels. The order of the blocks is line by line and each line is read from left to right. After the group into blocks, JPEG transforms each block through a Forward Discrete Cosine Transform (FDCT) [16]. The DCT gives a frequency map, with 8X8 or 64 elements. The transformation keeps the low frequency information which a human eye is sensitive to. In each block the DCT coefficients are composed of: A single Direct Current (DC) coefficient number, which represents the average intensity level value in each block and the remaining 63 are named Alternating Current (AC) coefficients. They reflect the frequency information of their row and column. These coefficients indicate the amplitude of a specific frequency component of the input array. The frequency content of the sample set at each frequency is calculated by taking a weighted sum of the entire set.

These AC values of one row are the results of this formula:

$$F(u) = \frac{1}{2} \sum_{x=0}^{7} f(x) C(u) \cos[(2x+1) u \pi / 16]$$

Where:

x is the index of value.

f(x) is the value itself.

u is the index of result.

$C(u) = 1/\sqrt{2}$ If u=0; C (u)=1 otherwise.

F (u) is the result itself.

In reality, the computer, the digital camera or any other device that generate JPEG images do not compute the cosine values over and over again. It would be an unnecessary waste of CPU cycles, so they use this formula:

$$F(u) = \frac{1}{2} \sum_{x=0}^{7} f(x) T(u, x)$$

Where T is the matrix described in Table 1:

Table 1. Values of Weight for One Row in an 8x8 Matrix

Index of value / Index of result	0	1	2	3	4	5	6	7
0	+0.707	+0.707	+0.707	+0.707	+0.707	+0.707	+0.707	+0.707
1	+0.981	+0.831	+0.556	+0.195	-0.195	-0.556	-0.831	-0.981
2	+0.924	+0.383	-0.383	-0.924	-0.924	-0.383	+0.383	+0.924
3	+0.831	-0.195	-0.981	-0.556	+0.556	+0.981	+0.195	-0.831
4	+0.707	-0.707	-0.707	+0.707	+0.707	-0.707	-0.707	+0.707
5	+0.556	-0.981	+0.195	+0.831	-0.831	-0.195	+0.981	-0.556

6	+0.383	-0.924	+0.924	-0.383	-0.383	+0.924	-0.924	+0.383
7	+0.195	-0.556	+0.831	-0.981	+0.981	-0.831	+0.556	-0.195

If f(x) (the intensity of each pixel) is equal in the entire row, each F(u) which holds u>0, will be zero. F(0) will be the sum of the row's values divided by $2\sqrt{2}$. This explains why JPEG is better pictures which do not have sharp changes. Pictures which have sharp changes like cartoons will have larger values and hence will be compressed not as good as the pictures without the sharp changes [17, 18].

As was mention above, the blocks are of 8X8, so one-dimensional DCT is enough for only one row. The DCT for an entire block is done by applying one-dimensional DCT separately for each row of eight pixels. The result will be eight rows of frequency coefficients. Then, these 64 coefficients are taken as eight columns. The first column will contain all DC coefficients; the second column will contain the first AC coefficient from each row, and so on. At that time, JPEG apply one-dimensional DCT for each of these columns.

In reality the one-dimensional DCT is not used and the entire calculation is done by one formula that reflects this calculation:

$$F(u,v) = \frac{1}{4}\sum_{y=0}^{7}\sum_{x=0}^{7} f(x,y)C(u)\cos[\frac{(2x+1)u\pi}{16}]C(v)\cos[\frac{(2y+1)v\pi}{16}]$$

Where:

x,y are the indices of value.

f(x,y) is the value itself.

u,v are the indices of result.

$C(v) = 1/\sqrt{2}$ if u=0; C(u)=1 otherwise.

$C(u) = 1/\sqrt{2}$ if v=0; C(v)=1 otherwise.

F(u,v) is the result itself.

It should be noticed that index 0, 0 contains the DC of the DCs. This value is called the DC of the 8x8 block.

The next step is the quantization. The 63 AC coefficients are ordered into a zig-zag sequence. Firstly JPEG puts the coefficient that its row index plus column index is 0, *i.e.,* the DC. Then JPEG puts the coefficients that their row index plus column index is 1. Next, JPEG puts the coefficients that their row index plus column index is 2 and JPEG continues until putting the last coefficient that its row index plus column index is 14. This zig-zag algorithm arranges the coefficients into one dimensional array as described in Figure 2.

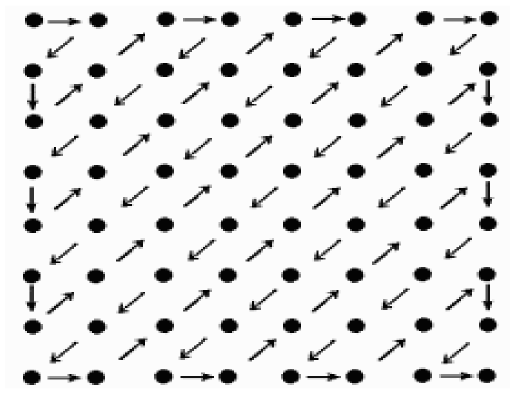

Figure 2. Zig-Zag Order of an 8x8 Block

In each block, each of the 64 coefficients is divided by a separate "quantization coefficient". The quantization coefficients are set according to the desired image quality. In point of fact, the image creator decides about a quality level (which is a number amongst 1 to 100) and according to this number the quantization coefficients are set. There is no standard method how to translate this quality level number into the quantization coefficients, so the quantization coefficients themselves are stored in the compressed image.

The results of the division are rounded to integers. This step loses some information because of the rounding. Furthermore, it can be noted that even if the quantization coefficient is 1, some information will be lost, because typically the DCT coefficients are real numbers.

The last step uses a traditional compression method with the aim of reducing the size of the data. The dictionary methods like the version of Lempel-Ziv methods is more suitable for text compression because it looks for an exact reappearance of strings which is less likely to occur in pictures, so JPEG encodes the reduced coefficients using either Huffman or Arithmetic coding. Usually a strong correlation appears between DC coefficients of adjacent 8X8 blocks. Therefore, JPEG encodes the difference between each pair of adjacent DC coefficient. Baseline JPEG model uses two different Huffman trees to encode any given image, one Huffman tree for the DC coefficients' length and the other Huffman tree for the AC coefficients' length. The tree for the DC values encodes the lengths in bits (1 to 11) of the binary representations of the values in the DC fields.

The second tree encodes information about the sequence of AC coefficients. As many of them are zero, and most of the non-zero values are often concentrated in the upper left part of the 8 × 8 block, the AC coefficients are scanned in the zig-zag order specified by the quantization step, *i.e.*, processing elements on a diagonal close to the upper left corner before those on such diagonals further away from that corner, that is the order is given by (0, 1), (1, 0), (2, 0), (1, 1), (0, 2), (0, 3), (1, 2), *etc.*, The second Huffman tree encodes pairs of the form (n, l), where n (limited to 0 to 15) is the number of elements that are zero, preceding a non-zero element in the given order and l is the length in bits (1 to 10) of the binary representation of the non-zero quantized AC value. The second tree also includes code words for EOB, which is used when no non-zero elements are left in the scanning order and for a sequence of 16 consecutive zeros in the AC sequence (ZRL), necessary to encode zero-runs that are longer than 15. The Huffman trees used in baseline JPEG are well-known and fixed. The trees can be found in [19]. There is also an option to use a specific tree; however this option is only on the odd occasion used.

Each 8X8 block is then encoded by an alternating sequence of Huffman code words and binary integers (except that the codeword for ZRL is not followed by any integer), the first codeword belonging to the first tree and relating to the DC value, the other code words encoding the (n, l) pairs for the AC values, with the last codeword in each block representing EOB.

3. Steganography System Using JPEG

The DCT-based compression can be viewing the FDCT as harmonic analyzer and the IDCT as harmonic synthesizer. In the case of JPEG format each 8X8 block of the source image samples is effectively a 64-point discrete signal which is a function of the 2D dimensional space x, and y. The FDCT takes such a signal as its input and decomposes it into 64 orthogonal basis signals. The output of the FDCT is the set of unique 64 basis signal amplitudes, which can be regarded as the relative amount of the 2D spatial frequency contained in the 64-point input signal [20, 21].

As was described in the previous section, the results of these mathematic formulas are treated as integers. Moreover, some numbers divide these integers and the results again are accommodated in integers. Obviously, rounding floating-point numbers to integers causes some loss of data, so the values of JPEG's coefficients are not accurate. The Steganography algorithm exploits the fact that the values are not accurate anyway. It changes the least significant bits of the values so they will not contain a data of an image. These bits will contain an encrypted message.

The Steganography algorithm takes into account some or all of the following considerations:

- **The quality of a picture**

JPEG standard stipulates that a user can set the quality of the picture when he creates it. More qualitative image consumes more disk space [22]. The quality value is a number not less than 100. This value sets the numbers which divide the AC coefficients. When a smaller number divides an AC coefficient, more data will remain, so when reconstructing the image, the data will be more accurate and a better picture will be seen.

When using more qualitative image, the Steganography algorithm can insert more bits [23]. Almost every time a high quality image is used, the added data is redundant [24]. The Steganography algorithm can take advantage of this redundant data. It can replace the redundant data by an encrypted message.

- **Coefficients might be changed.**

Human's eye is more sensitive to first coefficients. Indeed, smaller numbers will divide the first coefficients, when JPEG performs the quantization step. On the other hand, because smaller numbers divide the first coefficients, they may have more data, which can be used by the Steganography algorithm.

The Steganography algorithm must take into account the size of the compressed data. JPEG treats the data as sequences. Each one of these sequences contains a sequence of zeros and another value at the end. When there are just zeros left in the end of a block, JPEG outputs an EOB. It is very common to find a block from a JPEG image with an EOB after about 20 coefficients or so [25,26]. The rest of the coefficients after the EOB are zeros, so they are not coded. If one of latter coefficients will be chosen and a non-zero number will be placed in it, a long sequence of zero will be added before this coefficient. The entropy encoder will recognize an irregular situation, which will yield a long code. Obviously, such codes will enlarge the space of an image.

- **The number of bits in each coefficient**

When using more bits in order to accommodate the encrypted message, fewer bits are left for the original value of the coefficient. The quality of the picture can be harmed. A poor quality picture might be suspicious. Someone might suspect that the picture passed some processing. This is exactly what we do not like to come about.

The Steganography algorithm can be summarized as:

- while there are more bits in the encrypted message
    - read N bits.
    - read one JPEG block into B.
    - decompress B using the entropy decoder.
    - replace N chosen bits from B by N bits from the encrypted message.
    - compress B using the entropy encoder.
    - write block B.

4. Experiments

The method was tested on an image from a seaport, which is shown in Figure 2. The first lines of the image have both a uniform color area and an area, which has sharp changes. This image was chosen, in order to check the sensitivity of the picture to both sorts of data. This image has a 75% quality. This quality is very common in use and it used as a default value in many applications.

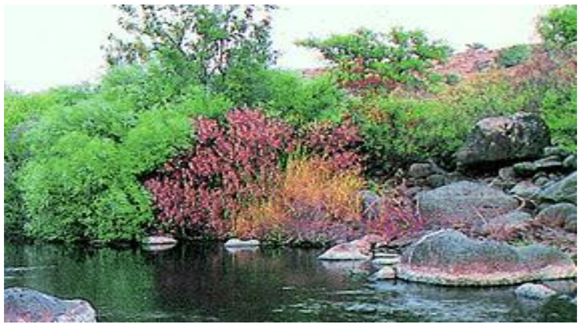

Figure 2. Original Tested Image

The sequence of bits which was inserted into this image is:

11110000110100100110000101110001010110000111101100000010111011101110000011 0
1000000111111001111011000011110100000100010100001111110101001010110010110101 10
11111011101001101000111011101010101111100000100110111001110010000010010110001 0
01000111111010100011111000100001111001010100010101001001011010011000100000110 0
01100011100001100011010010001001001100011101101000110101011001010110010110000 1
00101110001111010111010001000010000010011101011100000010001000111000001010111 1
01010000000111000011110100001110110000100010100011110110000011001101011110011 1
11010110000111001001000000010010110010011000000010111001000110000010111110101 1
01001001000000111000111100010001110010111010001000110000011101001001011010111 0
10110001101110111110110101010101000111011100000111000100101000001110010111011 0
0100010000001101110100101000110000011000110100110011010010101000011110 1

The pictures in Figure 3, were created by insert just one bit into the:

- first AC coefficient
- 21th coefficient
- 41th coefficient
- 61th coefficient

in each JPEG block.

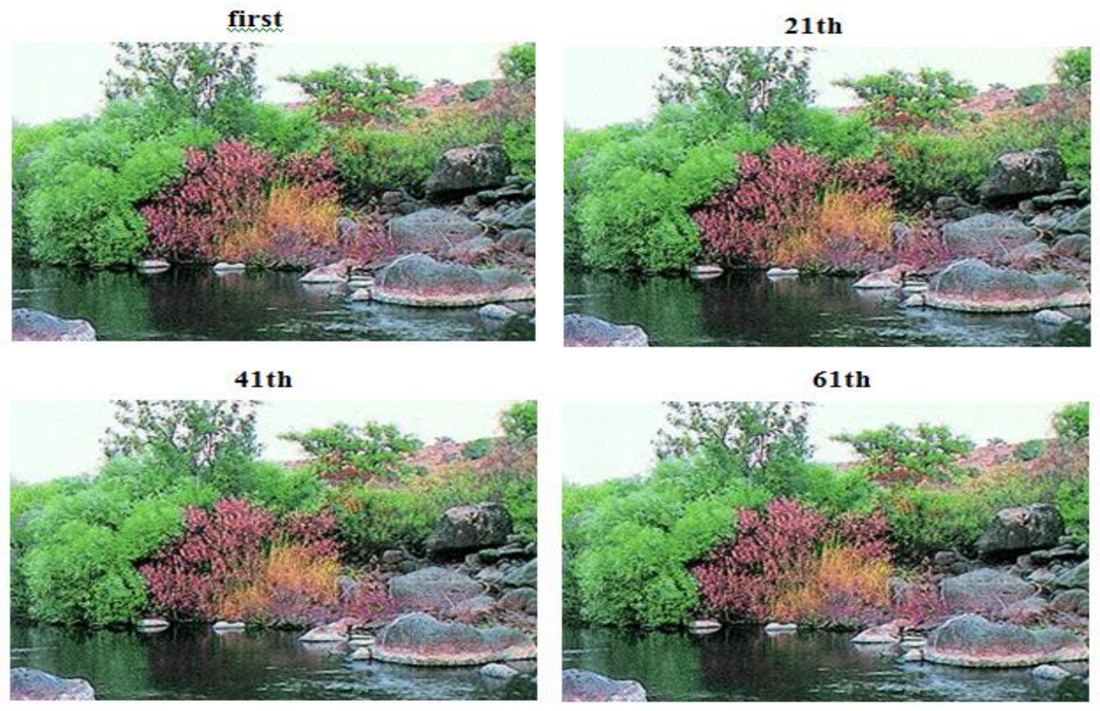

Figure 3. Changing One Bit in Various Coefficients

When trying to change more than one coefficient, this image will loose a significant data. Anyone who sees the last two pictures in Figure 4 can notice that they passed a sort of a treatment. The uniform data is usually more affected. When there are a lot of details, it is hard to notice if something was added or was deleted. When the data is uniform, every addition or deletion is more noticeable. Figure 4 shows changing of 1, 2, 4 and 8 bits in the first AC coefficient.

In order to increase the ability of putting more bits in coefficients, the quality of a picture can be increased. Such an increase gives more bits to each coefficient. Most of these bits are redundant, so changing them might be negligible. Figure 5 shows same picture as was shown in Figure 4, created with a 90% quality. Again, some bits were replaced in the first AC coefficients in the picture. The disadvantage of using high quality image is a larger size of the picture.

Sizes of compressed images were affected by number of non-zero values that were compressed. When we insert more bits, the chance to have a non-zero value will grow. In addition, small numbers are handled more efficiently by the compression method implemented in JPEG [27].

We can see in Figure 6 that when a latter coefficient is changed, the size of the compressed data is larger. Usually, when there is a non-zero value in a latter coefficient, a long sequence of zeros will be before this coefficient. A long sequence of zeros before a non-zero value is a very rare case [28, 29], so unlike dictionary codes [30,31] the entropy encoder will yield for this data, a long sequence of bits. This will increase the size of the compressed image.

The size of a high quality image will be even larger, because JPEG saves more data for each coefficient. The sizes of the images in Figure 5 are 36970 (4 bits) and 37105 (8 bits). The size of the original image with 90% quality is 36940.

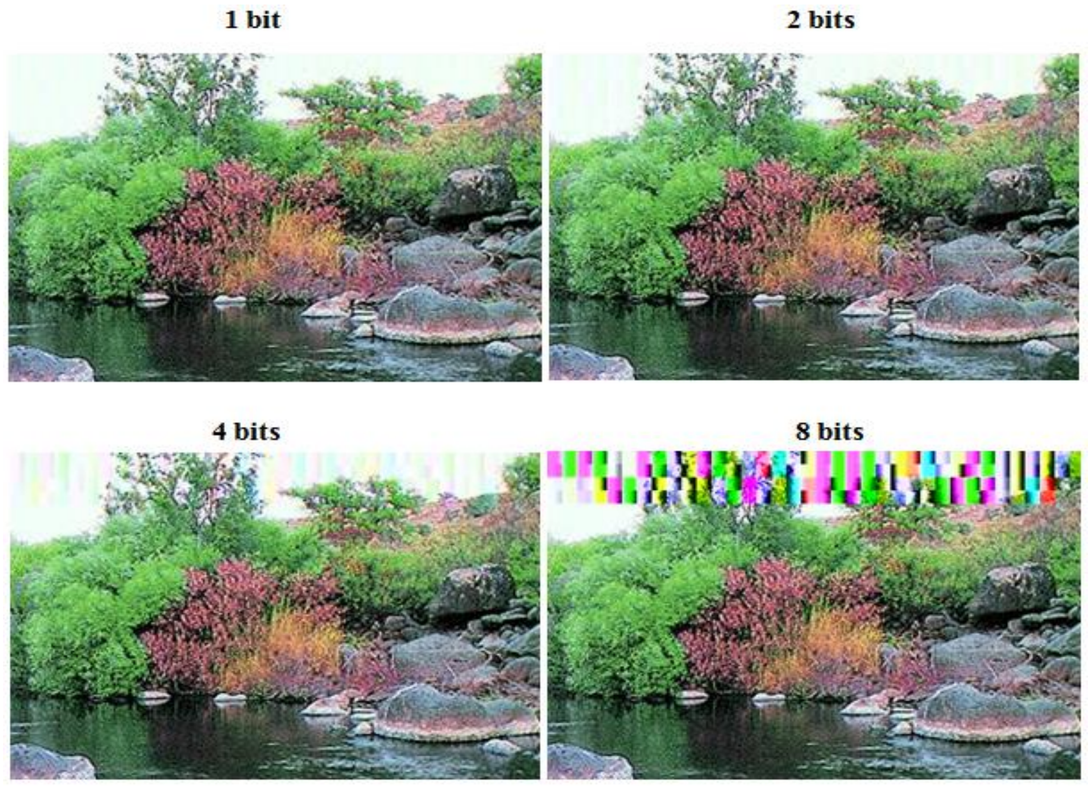

Figure 4. Changing Several Bits in First AC Coefficient

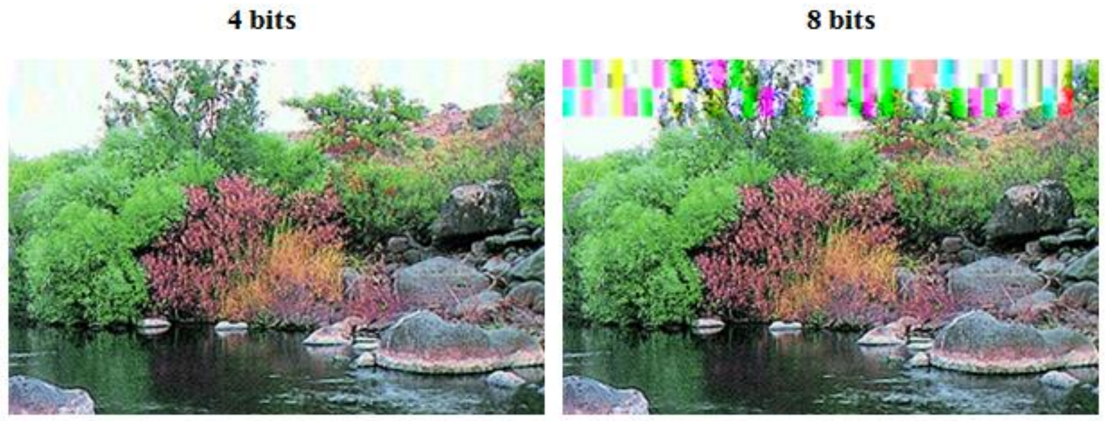

Figure 5. Changing Several Bits in the First AC Coefficient in a 90% Quality Image

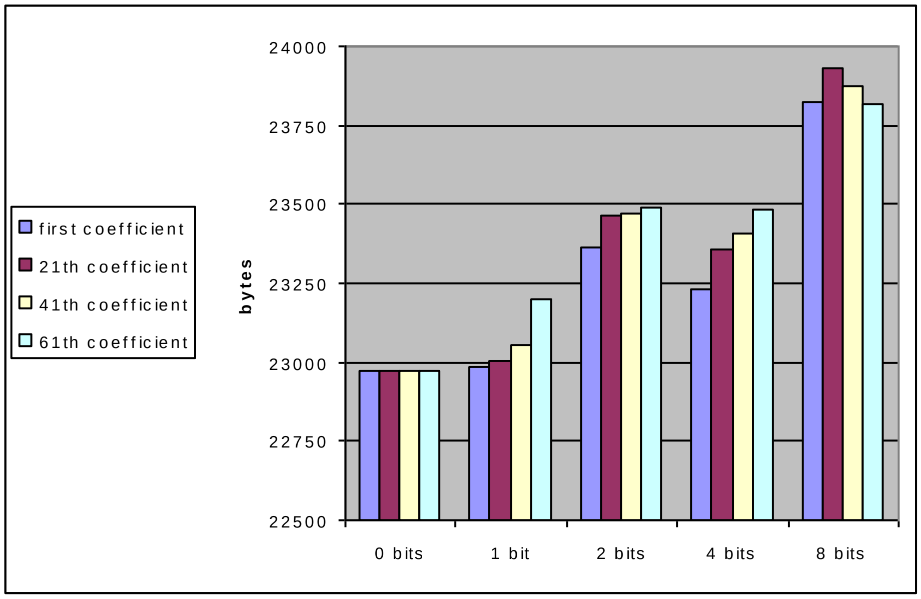

Figure 6. Size of Compressed Image When Some Bits Were Changed

5. Conclusions and Future Work

The task of securing cyberspace has emerged as perhaps the single most important seaport security challenge of the decade [2]. An increasing number of functions are dependent on port computer systems and the Internet whereas security issues become more imperative [32,33].

The results of the experiments are promising. JPEG takes off part of the image data, which seems to be redundant. It seems that more data can be removed; yet it will be hard to notice it. This redundant data can be exploited in order to transfer confidential messages.

A further research could be done on what else might be done with this redundant data in JPEG images.

References

[1] E. Ochin, L. Dobryakova, Z. Pietrzykowski and P. Borkowski, "The application of cryptography and steganography in the integration of seaport security subsystems", Scientific Journals Maritime, Zeszyty Naukowe Akademia Morska w Szczecinie, **(2012)**.
[2] L. Musser, "Securing Seaport Cyberspace", U.S. Department of Homeland Security, **(2013)**.
[3] P. Thiyagarajan, G. Aghila and V. P. Venkatesan, "Stepping up internet banking security using dynamic pattern based image steganography", Advances in Computing and Communications, Springer, **(2011)**, pp. 98-112.
[4] B. Li, J. He, J. Huang and Y. Q. Shi, "A survey on image steganography and steganalysis", Journal of Information Hiding and Multimedia Signal Processing, vol. 2, no. 2, **(2011),** pp. 142-172.
[5] C. Cachin, "Digital steganography", Encyclopedia of Cryptography & Security, **(2011),** pp. 348-352.
[6] Y. Wiseman, "The still image lossy compression standard – JPEG", Encyclopedia of Information and Science Technology, Third Edition, **(2014)**.
[7] G. K. Wallace, "The JPEG Still Picture Compression Standard Communication of the ACM 34", **(1991),** pp. 3-44.
[8] Y. Wiseman, K. Schwan and P. Widener, "Efficient End to End Data Exchange Using Configurable Compression", Operating Systems Review, vol. 39, no. 3, **(2005),** pp. 4-23..
[9] Y. Wiseman, "A Pipeline Chip for Quasi Arithmetic Coding, IEICE Journal – Trans", Fundamentals, Tokyo, Japan, vol. E84-A, no. 4, **(2001),** pp. 1034-1041.
[10] F. Li, X. Zhang, J. Yu and W. Shen, "Adaptive JPEG steganography with new distortion function", Annals of telecommunications-annales des telecommunications, **(2014),** pp. 1-10.

[11] D. Hearn and M. P. Baker, “Computer Graphic Prentice Hall”, Englewood Cliffs, NJ, **(1986),** pp. 295-307.
[12] A. K. Jain, “Fundamental of Digital Image Processing Prentice Hall Information and System Sciences Series”, **(1986),** pp. 553-557.
[13] R. W. G. Hunt, “The Reproduction of Colour Fountain”, Press England, **(1995),** pp. 507-511.
[14] P. A. Laplante and A. D. Stoyenko, “Real Time Imaging, Theory, Techniques and Applications”, IEEE Press Inc. NY, **(1996),** pp. 122-124.
[15] G. J. Awcock, “Applied Image Processing McGraw-Hill”, Book Company, **(1996),** pp. 260-261.
[16] K. R. Rao and P. Yip, “Discrete Cosine Transform Algorithms, Advantages”, Applications Academic Press Inc., London, **(1990)**.
[17] Y. Wiseman and E. Fredj, “Contour Extraction of Compressed JPEG Images”, ACM - Journal of Graphic Tools, vol. 6, no. 3, **(2001),** pp. 37-43.
[18] E. Fredj and Y. Wiseman, “An O(n) Algorithm for Edge Detection in Photos Compressed by JPEG Format”, Proc. IASTED International Conference on Signal and Image Processing SIP-2001, Honolulu, Hawaii, **(2001),** pp. 304-308.
[19] “Information Technology Digital Compression and Coding of Continuous-Tone”, Still Images Requirements and Guidelines International Standard ISO/IEC 10918-1, **(1993)**.
[20] Y. Wiseman, “Fuselage Damage Locator System”, Advanced Science and Technology Letters, vol. 37, **(2013),** pp. 1-4, Jeju Island, Korea.
[21] Y. Wiseman, “Device for Detection of Fuselage Defective Parts”, Information Journal, Tokyo, Japan, vol. 17, no. 6, **(2014)**.
[22] Y. Wiseman, “Camera That Takes Pictures of Aircraft and Ground Vehicle Tires Can Save Lives”, Journal of Electronic Imaging, vol. 22, no. 4, **(2013),** pp. 041104.
[23] J. Fridrich, “Effect of cover quantization on steganographic fisher information”, IEEE Transactions on Information Forensics and Security, vol. 8, no. 2, **(2013),** pp. 361-373.
[24] R. K. Toor and R. Kaur, “A Steganographic Method Based Upon Jpeg and Quantization Table Modification”, International Journal of Information Technology, vol. 6, no. 1, **(2012),** pp. 19-21.
[25] Y. Wiseman, "Take a Picture of Your Tire!", Proc. IEEE Conference on Vehicular Electronics and Safety (IEEE ICVES-2010) Qingdao, ShanDong, China, **(2010),** pp. 151-156.
[26] Y. Wiseman, “The Effectiveness of JPEG Images Produced By a Standard Digital Camera to Detect Damaged Tyres”, World Review of Intermodal Transportation Research, vol. 4, no. 1, **(2013),** pp. 23-36.
[27] Y. Wiseman, “Burrows-Wheeler Based JPEG”, Data Science Journal, vol. 6, **(2007),** pp. 19-27.
[28] S. T. Klein and Y. Wiseman, “Parallel Huffman Decoding with Applications to JPEG Files”, The Computer Journal, Oxford University Press, vol. 46, no. 5, **(2003),** pp. 487-497.
[29] S. T. Klein and Y. Wiseman, “Parallel Huffman Decoding, Proc. Data Compression Conference DCC-2000”, Snowbird, Utah, USA, **(2000),** pp. 383-392.
[30] S. T. Klein and Y. Wiseman, “Parallel Lempel Ziv Coding”, Journal of Discrete Applied Mathematics, vol. 146, no. 2, **(2005),** pp. 180-191.
[31] Y. Wiseman, “The Relative Efficiency of LZW and LZSS”, Data Science Journal, vol. 6, **(2007),** pp. 1-6.
[32] Y. Wiseman and Y. Giat, “Multi-modal passenger security in Israel”, Multimodal Security in Passenger and Freight Transportation: Frameworks and Policy Applications, Edward Elgar Publishing Limited, **(2014).**
[33] M. D. Orosz, T. Milind and R. Heather, “Adaptive Real-Time eaport Security (DynaPortSec)-Integrated PortSec”, Game Theory and Adaptive Adversary, **(2013)**.

Author

Dr. Yair Wiseman, got his PhD from Bar-Ilan University and completed two Post-Doc - one at the Hebrew University of Jerusalem and one in Georgia Institute of Technology. Dr. Wiseman's research interests include embedded systems, vehicular systems, intelligent transportation systems, process scheduling, hardware-software codesign, memory management, real-time operating systems. Dr. Wiseman's has been supervising many graduate students and his papers have been published in many venues.