I. J. Computer Network and Information Security, 2019, 11, 21-27
Published Online November 2019 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2019.11.03

Enhancement of Capacity, Detectability and Distortion of BMP, GIF and JPEG images with Distributed Steganography

Istteffanny I. Araujo
London Metropolitan University/School of Computing and Digital Media, London, N7 8DB, United Kingdom
E-mail: i.araujo@londonmet.ac.uk

Hassan Kazemian
London Metropolitan University/School of Computing and Digital Media, London, N7 8DB, United Kingdom
E-mail: h.kazemian@londonmet.ac.uk

Received: 06 October 2019; Accepted: 16 October 2019; Published: 08 November 2019

Abstract—The advance of Big Data and Internet growth has driven the need for more abundant storage to hold and share data. People are sending more messages to one another and paying attention to the aspects of privacy and security as opposed to previous decades. One of the types of files that are widely shared and instantaneous available over the web are images. They can become available as soon as a shot is taken and keep this closely related to the owner; it is not easy. It has been proposed here to use Steganography to embed information of the author, image description, license of usage and any other secrete information related to it. Thinking of this, an analysis of the best file types, considering capacity, detectability, and distortion was necessary to determine the best solution to tackle current algorithm weaknesses. The performance of BMP, GIF, and JPEG initialises the process of addressing current weaknesses of Steganographic algorithms. The main weaknesses are capacity, detectability and distortion to secure copyright images. Distributed Steganography technique also plays a crucial part in this experiment. It enhances all the file formats analysed. It provided better capacity and less detectability and distortion, especially with BMP. BMP has found to be the better image file format. The unique combination of Distributed Steganography and the use of the best file format approach to address the weaknesses of previous algorithms, especially increasing the capacity. It will undoubtedly be beneficial for the day to day user, social media creators and artists looking to protect their work with copyright.

Index Terms—Steganography, Security, Copyright, Privacy, Images.

I. Introduction

Image is a series of pixels that show or symbolise something in time. These pixels hold RGB (Red, Green, Blue) components ranging from 0 to 255 in 8 bits blocks. Image Steganography refers to hiding secrete content inside a carrier image to protect it, so it is not visible to anybody in plain sight [1]. This content can be text, another image or any other file format such as a spreadsheet, a word document and a PDF. The pixels from the image can be modified to embed data altering the file and providing different quality output [4]. The visible distortion and vulnerability to detection via statistical analysis will be considerably more if the hidden data is in less pixels as possible. The detection of the content will also be higher on all the image format as BMP, GIF and JPEG. Current algorithms experience low capacity and high detectability. Improve capacity and diminish detectability and distortion are the main objectives here.

Therefore, an analyse and use of the best carrier (image file format) and the combination of Distributed Steganography where is an innovation to tackle all the issues. The Distributed Steganography method proposed, embeds the private data over several different images generated from the original carrier of stego-image. This technique provides a successful steganography tool to share images on the web without detaching the author copyrights [15].

II. Capacity, Detectability and Distortion

The capacity of Steganography algorithms measures how much data is possible to embed in a carrier file. With the specific limitations of the given algorithm, in our case, our carrier file is an image [5]. Capacity also helps to identify steganography on an image if the bits are modified massively. Capacity is also known as the primary measurement for Steganography algorithms. Algorithms with higher capacity tend to be weaker on security detectability [16]. They are not very proficient with Steganalysis, e.g. statistical analysis. High capacity algorithms also present more distortion on a steganographic file. The idea of improving capacity with minimal impact on detectability and distortion is an excellent topic for research and contribution to knowledge

as they usually have an enormous impact on each other. On the image, a message is added via Encryption where all or part of the message can be encrypted and displayed on the file in an unreadable format. Watermark is used to add a visible copyright message and Encryption is used to an imperceptible to secure it. Improvement of capacity without significant effect on detectability and distortion is very challenging. Steganalysis investigation is needed to endeavour detectability tests of the algorithm [12]. The simplest method to detect steganography is observing distortion. If there is visible distortion, there is no need to apply mathematical algorithms to guess that there is content hidden inside the file. However, some algorithms may have minimum distortion and still have a great detectability by statistical analysis [16]. One example is the Least Significant Bits algorithm (LSB). It does not always have distortion, depending on how much data is hidden. As per its name, the secret content goes in the least significant bits of our carrier file. A mathematical Steganalysis algorithm to identify the LSB capacity (1) can be applied to find the hidden data.

(1) Capacity Definition Algorithm

$$N_{\gamma}^{*} = \left\{ \max\ N\ s.t \left| \beta_{D}^{(N)} - \alpha_{D}^{(N)} \right| \le \gamma D \right\}$$

Every specific algorithm constraint needs to be taken into consideration to measure the capacity on steganography algorithm. For the same carrier, different algorithms and a capacity measure, e.g. using LSB requires finding the least significant bits only. Another algorithm may apply LSB plus looking for a specific colour shade like the blue less noticeable for the human eye [6], restricting the capacity, and so on. Therefore, the steganographic capacity definition in the above equation where N is the number of message-carrying symbols, a is the false alarm and D is detection probability.

III. Image Files

A. Graphics Interchange Format

The Graphic Interchange Format images, mostly known as GIF, is used widely over the internet. It can be stored in low resolution to save computer space as it applies a compressed lossless technique to a reduced size file. When downloaded, the images will still display excellent visibility for its Multibit graphics across different platforms [2]. As other image formats, this reads as a continuous data stream pixel-based file, and it has support for transparency, animation features. Allowing 8-bit colour indexing only, which is in practice 256 distinct shades compared to the 16.777.216 possibilities, it has limitations [11]. As per its limitations, the capacity for hiding data is significantly smaller than the BMP format. An attempt to embed a more substantial amount of data into a GIF file would result in significant distortion. However, it can still be used to store more straightforward messages, and this is likely to be done with simpler algorithms as LSB. The LSB usage would be the first guess if applying Steganalysis to this file type [16].

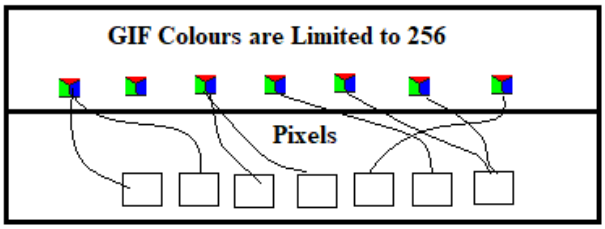

Fig.1. GIF Limited Colours to Pixels

B. Bitmap

The Bitmap (BMP) file format is a file generated from a map of bits, known explicitly in computing as a bit array. Most of the image files are similar; however, BMP has its unique charm since all bytes are entirely present in the file. Other formats use compression from the start. An example is JPEG; a compression applied to digitalised images. The file structure of BMP helps to program and explore all means of this format to improve our results. When referencing BMP, it implies one bit to one pixel of the image. These are not compressed and identify the depth colour of a digital image [6]. There are other variations of BMP that use compressions named differently. BMP file structure starts with a header of 14 bytes describing the file, followed by a DIB header detailing the bitmap and its pixel format as illustrated in Fig. 2. There are optional functions such as the Extra bitmasks to be used in conjunction with the compressed version of BMP [6]. A colour table is also aggregated into this optional structure bringing the colour pixel array, followed by the Gap1 block defining the alignment structure of the file. The Pixel array structure is not optional, every BMP file will have this to stipulate the specific value of each pixel, and this varies in size. The next two structures are also optional, Gap2 and ICC colour Profile varies in size and helps to manage the file colours.

C. Joint Photographic Expert Group

JPEG (Joint Photographic Expert Group) is not originally a file format. It is a lossy compression into byte of stream technique. It uses the codec specification of Discrete Cosine Transform (DCT). It produces a spatial frequency domain of 64 patterns, followed by application of the Discrete Cosine Transform (DCT) mathematical algorithm and Quantization [17]. It outputs transform coefficients generating the Entropy Encoder. They are tokens with values changed from positive to a number close to 0, subtracting the middle point (128) from every entry of the 0-255 RGB components, resulting in an easier to compress the signal [14].

Bitmap File Header BITMAPFILEHEADER	
Signature	
File size	
Reserved1	Reserved2
File Offset to PixelArray	
DIB Header BITMAPINFOHEADER	
DIB Header Size	
Image Width (w)	
Image Height (h)	
Planes	Bits per Pixel
Compression = BI_BITFIELDS	
Image Size	
X Pixels Per Meter	
Y Pixels Per Meter	
Colours in Colour table	
Important Colour Count	
Red Channel bitmask	
Green Channel bitmask	
Blue Channel bitmask	

Fig.2. BMP File Structure

IV. Discrete Cosine Transform

The Discrete Cosine Transform method divides images into sub-bands; small high-frequency components are deleted and use only real numbers for JPEG compression. When embedding, there are four steps before applying DCT (2) [10].

(2) Discrete Cosine Transform

$$c(u) = a(u)\sum_{x=0}^{N-1} f(x)\cos\left[\frac{\pi(2x+1)u}{2N}\right]$$

for u = 0,1,2, ..., N − 1. Inversing, then:
for x = 0,1, 2, …, N − 1. α(u) denotes:

$$f(x) = \sum_{u=0}^{N-1} a(u)C(u)\cos\left[\frac{\pi(2x+1)u}{2N}\right]$$

The Cosine Transform algorithm code simulated breaks the figure into partitions of an 8x8 block [14] as an example, and reassembled it as per steps below:

1) DCT Object Created.
2) Use the 8*8 matrix to compress, illustrated in Fig. 3.
3) Use dequantitizeImage () method.
4) Use inverseDCT () method.

There is a quantisation (signal processing like rounding and truncating) matrix set up. The temporary matrix multiplication N * N uses cosine to re-multiply and generate the final matrix [3]. It quantises and round the value to an integer. In the inverse operation part, the N*N matrix will output 0 to 255 values for the pixels. Fig. 4 shows the concept of Least Significant Bits and Most Significant bits.

Fig.3. 8x8 Base Matrix of DCT

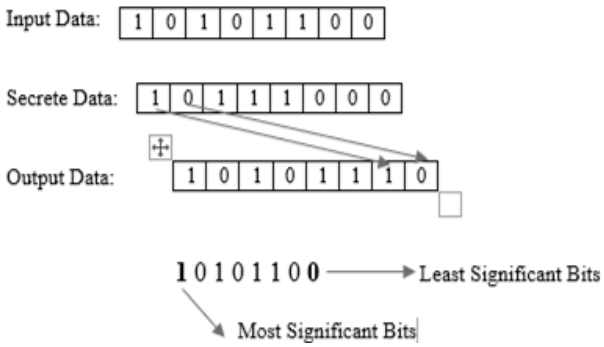

Fig.4. LSB insertion example

V. Experiments and Results

Detectability is one of the main issues in Steganography algorithms. The initial experiment to highlight how a small hidden message can alter the size of an image, DCT (Compression method) diminishes the scale between size of stego-image and size of Original Image. It makes this vulnerability less noticeable in different file formats.

Table 1 shows the results of image Steganography experiment using the algorithm where the stego-message has only 4.096 bytes. It demonstrates the results of the experiment carried with images of four different formats. The detectability is measured, taking into consideration the size of the image after embedding the data and distortion is analysed both visually per amount of data embedded in each file. The closer the output file is from the original file, the better the security will be as it is less detectable analysing the image size on Table 1:

Table 1. Comparing BMP to another formats experiment

Image	Original Size (bytes)	Stego-Size (bytes)	Distortion
PNG1.png	860.160	864.256	Low
GIF1.png	114.688	167.936	Low
BMP1.png	1163.264	1056.768	Low
JPEG1.png	262.114	1179.648	Low
		File Increase	Detectability
PNG1.png	860.160	0.47619%	Low
GIF1.png	114.688	46.4285%	Medium
BMP1.png	1163.264	-9.5489%	Low
JPEG1.png	262.114	349.999%	High

BMP is the best solution to Steganography on Image files because the format does not always use compression. It uses bitmap where all bits are present in the file as opposed to other file formats that use techniques of compression by standard. BMP is also loss-less, where the final stego-image pixel values modified are only the ones gathered to embed the data. The final file format of an original BMP image changes after embedding the secret information. There could be a smaller file size and not a more prominent image as a result of the extra information added, which generally implies Steganography addition addition. Consider detectability as the main vulnerability on LSB, not on the increased File Size, but on the "Least Significant Bits embedment" that trigger less security when using Statistical Analyses. Observe what happens with an image characterised with the binary bits below with the following 8 bits 10101010, where line one is the original image, while line number two shows the stego-image.

10001101 10011111 11110001 10001111
10000001 10000000 11111110 10101011

10001101 10011110 11110001 10001110
10000001 10000000 11111111 10101010

The Least Significant Bits has a secret message hidden in it, and by doing so, it demonstrates how statistical analysis algorithms could identify the hidden message rapidly. It scans these least significant bits (LSB) is the base of most Steganographic algorithms [8]. Adding the compression step using Discrete Cosine Transform takes it to another level. It still includes LSB, but it uses matrix complexity and quantisation (3).

(3) DCT Matrix Representation

$$Tij = \frac{1}{\sqrt{N}} \sqrt{\frac{2}{N}} \cos\left[\frac{(2j+1)}{2}\frac{i\pi}{N}\right] if\ i \geq 0$$

Apart from putting the file formats into detection testing, the frequency power/agility was tested on the file formats to measure security in PSNR (4), which outputs dB. Results of experiments made with different file formats are in Table 2. The higher is the PSNR, the higher is the security on the algorithm and the larger the image, the higher is the capacity [7]. The difference between the smaller file and the larger one is 1048.576 bytes

(4) Measuring Security on Image Steganography Formula

$$PSNR = 10 \log_{10} \frac{255^2}{MSE} dB$$

File 1 is 1.352380952380952 smaller than the larger file.

File 2 is 10.14285714285714 smaller than the larger file.

File 3 is our larger file.

File 4 is 4.4375 smaller than the larger file.

Table 2. Power Measurement per File Format

Image	Power in dB	Security	Capacity
PNG1.png	9.34dB	Low	Medium
GIF1.gif	10.86dB	Medium	Low
BMP1.bmp	65dB	High	High
JPEG1.jpg	8.99dB	Low	Low

The bigger the image, the higher is the security as there is more space to hide the data. BMP has one of the best performances between other file types, including for capacity and security, achieving better results than Nidhi in 2014 [13] and Chia-Chen-Lin in 2010 [9]. Therefore, the algorithm is settled to transform input files into BMP to increase the capacity. It diminished the file size after the application of Steganography.

JPG is the one that performed least to use, as using the same algorithm and same stego-message, the file size has increased by approximately 350% after Steganography.

The below plain JPEG of 2028x1520 pixels example demonstrates the approach. The image name is Canary (C:\Users\istteffanny.araujo\Desktop\canary.JPG). The sample message "I am at university today to provide an update of my project" will be hidden in it. The 2x2 matrix illustrated in Fig. 5. is below to break the image for the distribution of the secret data.

Fig.5. Canary.JPG broken into a 2x2 matrix

There is also the option to choose between two encryption algorithms before transforming the result in binary to distribute later the stegano files generated by the application. RC4 is quicker, occupies less space as the encryption key is shorter than RSA. The RSA is slower because the encryption key is more significant, hence securer. Let us see the detailed output example of how the converted/encrypted data is on the system.

Plain Message:

I am at university today to provide an update of my project

Plain Message in Binary:

01001001 00100000 01100001 01101101 00100000
01100001 01110100 00100000 01110101 01101110
01101001 01110110 01100101 01110010 01110011
01101001 01110100 01111001 00100000 01110100
01101111 01100100 01100001 01111001 00100000
01110100 01101111 00100000 01110000 01110010
01101111 01110110 01101001 01100100 01100101
00100000 01100001 01101110 00100000 01110101
01110000 01100100 01100001 01110100 01100101
00100000 01101111 01100110 00100000 01101101
01111001 00100000 01110000 01110010 01101111
01101010 01100101 01100011 01110100

The message below demonstrates the output of the message encrypted in RC4 encryption which outputs a smaller message.

Message Encrypted in RC4 has 11 characters:
[B@1b0375b3

The below message demonstrates the same message if encrypted in RSA. RSA encryption has a longer key, so the same message will output a more significant number of junk characters which is intentionally done to mask the secret information.

Message Encrypted in RSA has 245 characters:
//Displays 245 Junk Characters

Even if the original file is not of the BMP type, it calculates how much data should be placed randomly into each file. The division is almost equally, the possible remainders of non-exact divisions could be ignored or pointed to a cluster with similar reference. There is also a possibility of it to be left out. For this example, it was added less data to a random file giving less clue of the content, and this is done programmatically on the code.

The Result of the distributed encryption is the four stegano files with secrete data as per Fig. 6. From here, there are a few proposed approaches, one it is to reassemble the image that has been "cut" and added the secrete data into a single image for storage. Another way is to embed these fragments into the original plain image. It adds extra security, and it keeps the stegano-images hidden. There is no actual need to generate the stegano-images at the start.

Fig.6. Stegano files containing secrete data distributed

The Original file is a 2028 x 1520 JPG file converted to four 1014 x 760 BMP files to increase the embedding capacity. Considering the four available images to embed data, the total number of pixels has increased to 4056 x 3040, representing a 100% x 100% increase in this example.

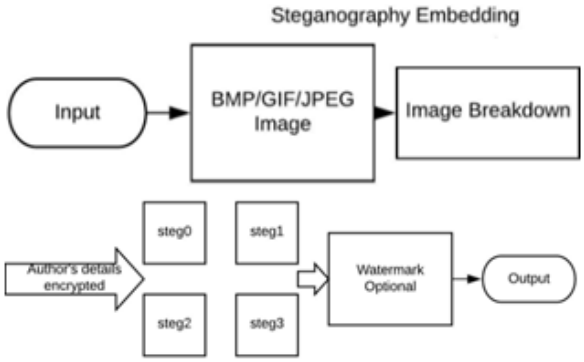

Fig.7. DSoBMP-I Algorithm Explained

The capacity doubled by using BMP and our proposed DSoBMP-I algorithm on a 2 x 2 matrix in comparison to using other file formats on standard steganography. Detectability and distortion is improved as there are more files to distribute the secret data.

More files to embed the data means less to embed into each file. Less embedment in a file means less distortion and less detectability because of extra layers added and the distributed technique. It will be difficult the retrieval of the secrete data without knowing the exact number of matrixes applied to the embedding, the chosen encryption and other optional approaches and random methods.

Comparing its final stegano images also pass the plain sight test where no precise detection of the secret message is appointed, the secret message is not detectable on plain sight, and the final stego-image has no distortion with the secrete data added. Fig. 6 shows the partitioned file without any data added and Fig. 8 shows the result of embedding part of the secret data.

Fig.8. Plain Image x stegano-image

Finally, Fig. 6. Fig. 8. and Fig 9. have the successful use of BMP that is proven the most efficient together with our distributed DSoBMP-I Steganography prototype, denoted as a *New Adaptative Distribution method*. Fig. 10 demonstrates the overall achievement of DSoBMP-I using a 2x2 matrix. Here, the algorithm has the output of the example described in this paper.

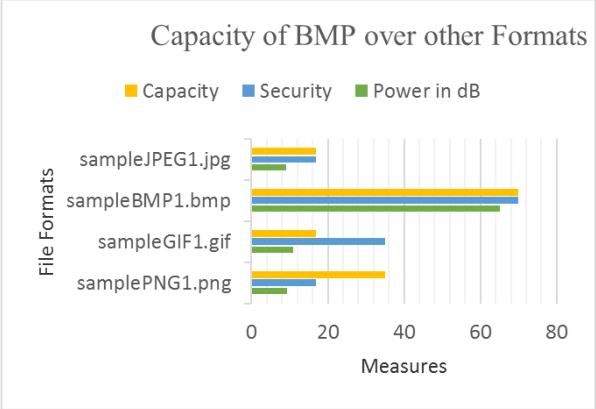

Fig.9. Efficiency of BMP over other file formats

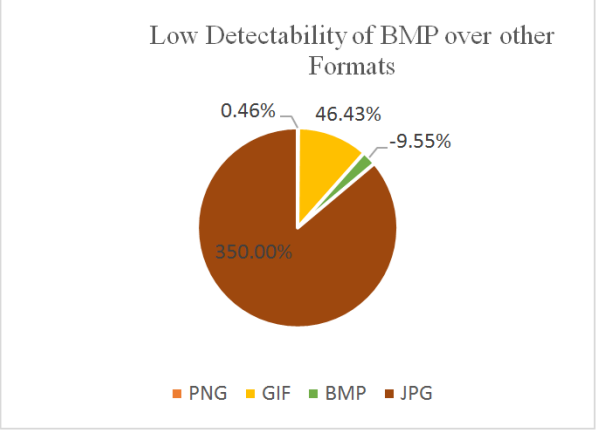

Fig.10. Low Detectability of BMP over other formats per file increment after Steganography

VI. Conclusions

The proposed algorithm improved the overall Steganography performance applied in Security for several years. Identifying the main weaknesses of current techniques, the DSoBMP-I approach proven to tackle the low capacity, high detectability and distortion issues. These are common within most steganographic algorithms used today not always equipped to deal with Big Data. Different experiments with several algorithms and image formats showed that the best image type to embed the data is BMP.

The choice of the two useful encryption algorithms by the agility in case the embedded data or files are broad, and by its complexity, where files are not enormous. The extra security comes from the new method of distributing the secrete data into a set of images that comes from the original file supplied in conjunction with the choice of size of partitions. The approach has proven to increase capacity by 100% compared to other algorithms where file format is not essential and distributed does not apply. All, using a simple 2x2 matrix and this only increases as matrixes increases and has proven to output more power than past papers investigated like Nidhi's [13]. The power of the algorithm is also superior to previous researches as 68db of power was achieved in this method while other researches highlighted the power of 30db in 2010 and 65db in 2015.

Detectability and distortion are also not visible in plain sight. The next approach for our needs to secure Copywrite materials and Big Data can benefit from the DSoBMP-I methodology proposed here. The capacity, distortion and detectability issues are improved significantly using this new methodology. The improvement of data distribution in more than one carrier as opposed to the current algorithms and the application convert and embed the secret message to the best file format to use. Future works include improving the technique to merge the partitioned images back to original. In terms of Image Copyrights, add features to at least password-protect images. The next step is to protect against snipping tool, print and print-screen, which is challenging as there are different operating systems, software and software versions to consider.

References

[1] Pal, S.And Mishra, S.. (2019). An efficient steganography technique for images using chaotic bitstream. *IJCNIS*. 10 (1), p45-50. in press. doi: 10.5815/ijcnis.2019.02.03

[2] Budoor, S., Daniyal, A.And Li, C. (2016). Secret communication on facebook using image steganography: experimental study. International Journal of Computer Science and Information Security (IJCSIS). 14 (1), p428-444. in press.

[3] El-Latif, E., Taha, A. And Zayed, H. (2019). A passive approach for detecting image splicing using deep learning and haar wavelet transform. *IJCNIS*. 5 (1), p28-35. in press. doi: 10.5815/ijcnis.2019.05.04

[4] Hosam, O. (2019). Attacking image watermarking and steganography - a survey. *IJCNIS*. 3 (1), p23-37. in press. doi: 10.5815/ijitcs.2019.03.03

[5] Douglas, M., Bailey, K., Leeney, M. et al. Multimed Tools Appl (2018). An overview of steganography techniques applied to the protection of biometric data, p77. in press. doi: https://doi.org/10.1007/s11042-017-5308-3

[6] Nag, A. (2019). Low-tech steganography for covert operations. *IJCNIS*. 2 (1), p21-27. in press.

[7] Kim, C. & Yang, CN. Multimed Tools Appl (2015). Watermark with dsa signature using predictive coding, p74: 5189. in press. doi 10.1007/s11042-013-1667-6

[8] Khami, M.H. (2018). Transmitting security enforcement by text encrypting and image hiding technique using combined encrypt/hide keys. *IJEM*. 1 (1), p1-15. in press. doi: 10.5815/ijem.2018.01.01

[9] Lin, Chia-Chen & Shiu, Pei-Feng. (2010). High capacity data hiding scheme for dct-based images. Journal of Information Hiding and Multimedia Signal Processing. 1. in press.

[10] Mao, J. et al. (2016). A method to estimate the steganographic capacity in dct domain based on mcuu model. Wuhan University Journal of Natural Sciences. 21 (4), p283–290. in press. doi: https://doi.org/10.1007/s11859-016-1172-7

[11] Mazurczyk, W., Karaś, M., Szczypiorski, K. et al (2016) YouSkyde: Information hiding for skype video traffic. Multimed Tools Appl, 75: 13521. in press. doi: https://doi.org/10.1007/s11042-015-2740-0

[12] Mazurczyk, W., Szaga, P. & Szczypiorski, K. (2014). Using transcoding for hidden communication in telephony. Multimed Tools Appl, p70: 2139. in press. doi: https://doi.org/10.1007/s11042-012-1224-8

[13] M. Devi and N. Sharma, Improved detection of least significant bit steganography algorithms in colour and grayscale images. 2014 Recent Advances in Engineering and Computational Sciences (RAECS), Chandigarh, 2014, pp. 1-5. in press. doi: 10.1109/raecs.2014.6799507

[14] Rabie, T. And Kamel, I. (2017). High-capacity steganography: a global-adaptive-region discrete cosine transform approach. Multimedia Tools and Applications. 76 (5), p6473–6493. in press. doi: https://doi.org/10.1007/s11042-016-3301-x

[15] SentinelOne. (2019). Hiding code inside images: how malware uses steganography. Available: https://www.sentinelone.com/blog/hiding-code-inside-images-malware-steganography/. Last accessed 06/10/2019. Unpublished.

[16] Ansari, A. et All. (2019). A comparative study of recent steganography techniques for multiple image formats. *IJCNIS*. 1 (1), p11-25. in press. doi: 10.5815/ijcnis.2019.01.02

[17] J. Bhatia and M. Okade, A novel image enhancement technique based on statistical analysis of dct coefficients for jpeg compressed images. (2016) Twenty-Second National Conference on Communication (NCC), Guwahati, 2016, pp. 1-6., in press. doi: 10.1109/NCC.2016.7561122

Authors' Profiles

Istteffanny Isloure Araujo received the B.S.c and M.S.c degrees in Computer Science and Computer Forensics and IT Security from London Metropolitan University in 2013 and 2015, respectively. She is in the Intelligent Systems and Research Centre of the School of Computing and Digital Media. She studies big data, digital security, copyright and privacy using Cryptography while assisting in teaching at the university.

Dr Hassan Kazemian received a B.Sc. in Engineering from Oxford Brookes University, UK in 1985. He received an M.Sc. in Control Systems Engineering from the University of East London, UK in 1987. He followed with a PhD in Learning Fuzzy Controllers from Queen Mary University of London, UK, in 1998. He is currently a professor at London Metropolitan University. He worked for Ravensbourne College University Sector, UK, as a senior lecturer for eight years. Previous lecturing experience includes the University of East London, UK, University of Northampton, UK, and Newham College, UK. Research interests include AI and ML applications to cybersecurity. Prof. Kazemian is a Fellow of the Institution of Engineering and Technology FIET (formerly IEE) UK, Chartered Engineer (C.Eng.) UK, and Fellow of British Computing Society (BCS) UK.

How to cite this paper: Istteffanny I. Araujo, Hassan Kazemian, "Enhancement of Capacity, Detectability and Distortion of BMP, GIF and JPEG images with Distributed Steganography", International Journal of Computer Network and Information Security(IJCNIS), Vol.11, No.11, pp.21-27, 2019. DOI: 10.5815/ijcnis.2019.11.03