Capítulo 3 ASCII Encoding for Hidden Information Transport in RGB Digital Images

Chapter 3 Codificación ASCII para el transporte de información oculta en imágenes digitales RGB

CABALLERO-HERNANDEZ, Hector†*, GIL-ANTONIO, Leopoldo y AMBRIZ-POLO, Juan C.

Tecnologico de Estudios Superiores de Jocotitlan, División de Ingeniería en Sistemas Computacionales

ID 1st Author: *Hector, Caballero-Hernandez* / **ORC ID**: 0000-0002-2790-833X, **CVU CONACYT ID**: 445998

ID 1st Co-author: *Leopoldo, Gil-Antonio* / **ORC ID**: 0000-0002-7445-9426

ID 2nd Co-author: *Juan C., Ambriz-Polo* / **ORC ID**: 0000-0002-0956-8730, **CVU CONACYT ID**: 551103

DOI: 10.35429/H.2021.11.52.61

*hector.caballero@tesjo.edu.mx

A. Ledesma (Coord.) Ciencias de la Ingeniería y Tecnología. Handbooks-TX-©ECORFAN-México, 2021.

Abstract

Coding techniques allow representing a broad spectrum of digital objects to be concealing by steganography methods. The proposed method shows the treatment of digital objects as text, image, audio, video and executable files to be embedded in digital images RGB, by applying compression techniques such as LZ77 and ASCII with base 64 through of LSB. The proposed method uses an encoding based on vectors derived from deterministic fractals by calculating its fractional dimension to raise the embedded message security level. To encode files whose format is not represented as text strings (audio, video, executable files, etc), which must be compressed and then encoded again to embed the object in the cover image. In order to embed an object, the cover image is analysed by finding proper areas for embedding both an object and rules for message recovery. An analysis of the cover image is performing before embedded the object. The cover image analysis consists of finding one zone to embed a message, another one to insert the rules for hidden message recovery. Both zones receive a previous treatment to altering the original order of pixels to add the respective messages and in this way breaking the original injection scheme. The application of the proposed method allowed to obtain large loads of data embedded in stego-images, without visual distortions and without loss of data in the original message to being recovered.

Compression, Fractals, Redistribution

Introduction

Hidden information is generally used to protect and guarantee confidentiality of a message. In general, steganography is the science most applied to hidden information that studies methods of sending data so that it goes unnoticed (Katzenbeisser & Peticolas, 2000). In steganography, there exists a cover object and a stego-object. The first one is the between a message and a cover object. The most important techniques used are modifications in the space domain and those that use techniques in the frequency domain (Singhal & Ratthore, 2015), (Sofloo & Aghayi, 2017). Spatial methods for steganography most used are LSB (Least Significant Bit) method, consists of modifying the bit of least weight of a byte in the cover image (Das, Das, Bandyopadhyay & Sanyal, 2010). PVD (Pixel Value Differencing) method facilitates data insertion, in addition to generate significant changes in the image, the mechanism for information hiding is based on the substitution of values from different blocks of two contiguous pixels in an image (Jung & Lee-Young, 2012). The most important techniques used in the frequency domain are DFT (Discrete Fourier Transformation), DCT (Discrete Cosine Transformation) and DWT (Wavelet Transformation), one advantage of these techniques over spatial domain techniques is that information is less exposed to compression, clipping and image processing tasks (Velasco-Bautista, López-Hernández, Nakano-Miyatake & Pérez-Meana, 2007).

Some recent applications for steganography have combined deterministic fractals properties with spatial techniques within the frequency domain. Benoit Mandelbrot (Geethaa & Thamizhchelvyb. 2016) proposed the fractal term, as a fractional number representing its metric dimension, fractals are expressed as mathematical sets with similar patterns among them. Fractals can be precisely the same on all scales (Abbas & Hamza, 2014). Fractal dimension is calculated by Equation 1.

$$D_{MB} = \lim_{\varepsilon \to 0} \frac{logN(\varepsilon)}{log\frac{l}{\varepsilon}} \quad (1)$$

Were

D is the Euclidean dimension
MB is the Minkowski-Bouligand dimension,
l is the dimension number,
N is the number of self-similar objects
ε is the linear side.

Information compression allows to reduce the size of a message to facilitate its transmission. Compression techniques can be with data loss, where the process of original message reconstruction is irreversible. Those methods are usually applied when it is not necessary to keep all object's properties (audio, video, among others).

Compression techniques without loss preserves all objects' features while taking advantage of redundant zones, some of these are Length encoding, RLE encoding (Run Length Encoding), Human coding, LZW coding (Lempel-Ziv-Welch) and others (Salomon & Motta, 2010). In steganography, compression techniques allow to increase embedding capacity.

Literature review

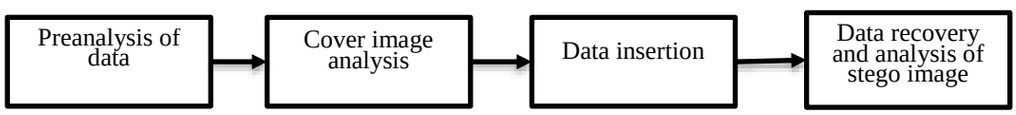
Chuang & Chang (2006) proposed a steganography method for compressed images with BTC. In the proposed scheme, a dynamic programming strategy was used to find the optimal solution of the bijective mapping function for the replacement of LSB with three bits to obtain low distortion in the stego-image. Sun, Lu, Wen, Yu & Shen (2013) presented a modification on absolute moment block coding (AMBTC), in this study they proposed a steganography method in which they use matrix inlays with Hamming code to insert a secret message in the compressed AMBTC bit stream showing acceptable results in embedding capacity, bit rate and hiding efficiency. Eswari, Leelavathy & Rani (1993) an LSB version was presented, after they applied Zhang's algorithm to embed information within fractal images (Zhang, Hu, Wang & Zhang, 2011), in addition to combining RSA technique. Desai & Desai (2014) used cryptography and Watermarking; they used the Mandelbrot fractal for image compression before embedding. An image is divided into sections according to the proposed fractal equation and then inserted by DTC. Tests were performed in grayscale and RGB images.

Stoyanova & Tasheva (2015) an LSB modification was suggested, while using a cryptographic key, which allows controlling data insertion and recovery through Rijndael system (Daemen & Rijmen, 2001). The experimental results indicated that the proposed scheme obtained a greater hiding capacity and efficiency than the other four existing schemes, in addition to guarantee stego-image quality. Ouyang, Park & Kau (2016). combined LSB with XOR, they obtained outstanding results in 512x512 images by embedding images of 25% size with respect to the cover image. Nehete & Bhide (2014) applied texture analysis and color segmentation in images, taking advantage of skin tone variants of a set of human faces, the applied steganography technique is DWT on the YCbCr model. Hussain et al. Hussain & Rafat (2016), proposed as a steganographic LSB mechanism and cryptography using XOR operations and SHA 256-bit method, they also added a pseudo-random number and Hashing techniques to hide information. Geethaa & Thamizhchelvyb (2016) presented an analysis application of chaos theory and fractal theory for steganography, in which the scope for compression methods through fractals is observed.

Proposed Scheme

The proposed method is focused on the insertion of hidden messages in RGB digital images. It has four important stages; first stage is pre-analysis of data, in which reduction of information and removal of symbols that an affect the steganography process are prioritized insertions, second stage is cover image analysis for distribution of the message data to hide, the third stage is data insertion, the fourth stage is data recovery and stego-image analysis. Figure 1 represents a general diagram for to proposed method.

Figure 1 General diagram of the proposed steganography method



Source: Own Elaboration

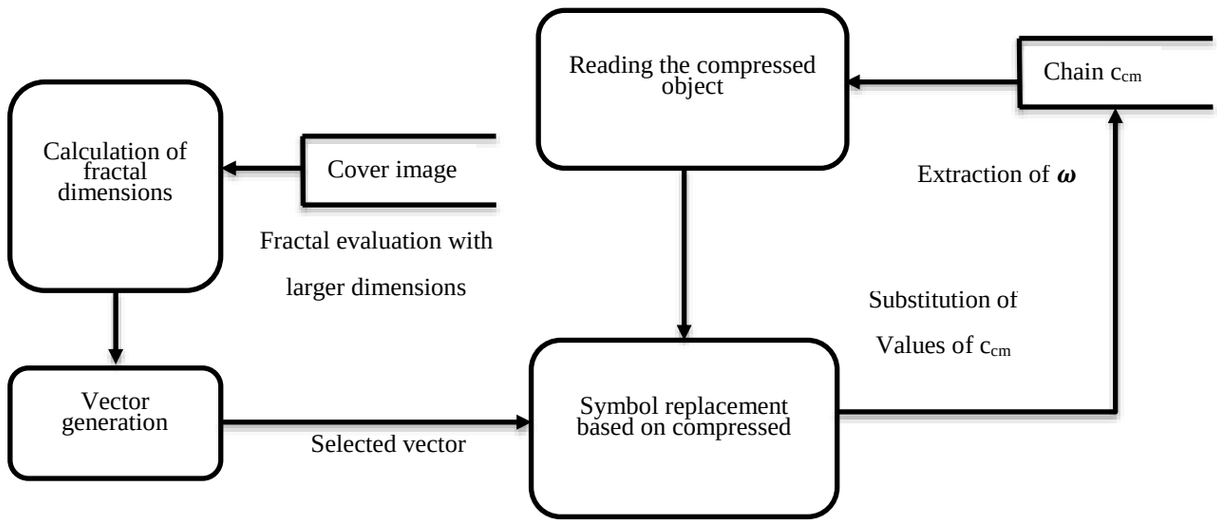
Pre-analysis of data. It determines if a message is possibly embedded in a cover image, also decide whether it is pure text, image, audio, video or executable file. The message reading process obtains its characteristics, and later it is analyzed to know what type of object it is, the next process consists in compressing and coding the message to be stored in a new file. For this stage, the following steps are followed.

Step 1: If the message is plain text, determine its size in bytes to verify if it is possible to insert in the cover image, otherwise:
Step 2: If the message is audio, image, video or executable convert into text string encoding and determine if its insertion into the cover image is possible, otherwise:
Step 3: Choose another cover image.
Step 4: Get the text string and compress it with LZ77 and store it in cm.
Step 5: Code ASCII 64 c_m and store it in c_{mc}.
Step 6: Calculate size of c_{mc} and calculate compression ratio.

Cover image analysis. In this stage, it is necessary to generate a check on the amount of data to be inserted in the cover image, the number of bytes to add are a maximum of 60% in relation with the first 3 bits per pixel in the cover image, to ensure that the quality of the stego-image is not an affected. When c_{cm} has finished being stored, it is necessary to replace its original alphabet with a new one, to increase safety in the steganography process. This process has been developing through the generation of vectors derived from equations of fractal dimensions. The c_{cm} chain should extract the symbols that make up its structure (ω), to be replaced by different symbols.

Figure 2 shows the data ow when substitution of c_{cm} symbols is generated, general steps are presented below. When a fractal dimension with floating point high precision (equal to 50 digits before the decimal point) is calculated, there are infinitesimal variations between the sequences of its obtained dimensions, therefore different fractional dimensions of a fractal can be obtained to generate a new alphabet.

Figure 2 Representation diagram of coding based on fractals



Source: Own elaboration

The vector ω_r represents the new symbol vector to replace ω, when this step is carried out, the symbols contained in c_{cm} are replaced. In this process, the relationship between substituted symbols and the fractal used for substituting symbols generation is stored.

Step 1: Generate a list of deterministic fractals.
Step 2: Calculate the number of alphabet symbols for c_{cm} string.
Step 3: Calculate a fractal dimension of all fractals stored in list.
Step 4: Obtain a fractal with the greatest number of variations in its dimension.
Step 5: Obtain the formed vector (v_f) by the fractal with the greatest number of variations.
Step 6: Associate v_f vector with ω of c_{cm} and replace symbols.
Step 7: Replace symbols of ω with ω_r.

Data insertion. Assignment of rules allows recovery of embedded data. This stage is achieved using regular grammars through the association of rules that represent certain symbols by a certain number of bits. In this stage, we propose grammars to allow de fining recognition rules of embedded message symbols. A formal grammar is a fourfold $G = (N, T, P, S)$ where N is a finite set of non-terminal symbols, T is a finite set of terminal symbols $N \cap T = \emptyset, P$ is a set of finite productions and S is a distinguished symbol $S \in (N \cap T)$ (Hopcroft & Ullman, 1993). Association rules specify fractal data chosen for the final coding phase of the message, the equivalence of modified message symbols, mechanism pixel distribution, among others. Once space preparation for insertion of recovery rules is completed, the number of pixels is modified and analyzed, so the analysis is carried out on the pixels that must be selected to insert the hidden data of the object. Therefore, an analysis must be carried out on pixels that must be selected to add hidden data of selected pixels. A corresponding zone of smaller size should be considered for rules storage. For the above, the next steps are executed.

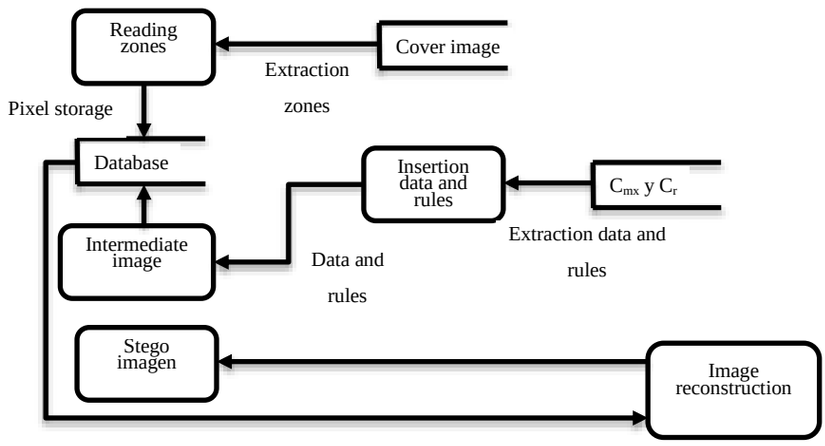
Step 1: Determine the number of pixels that should be occupied by c_{cm}.
Step 2: Determine the number of pixels that r_c recovery rules should store.
Step 3: Separate pixels destined for c_{cm} storage and those destined for r_c.
Step 4: Store the original pixels position on a database.

Finishing previous steps, the number of pixels destined for rules and for message has been determined, it is necessary to generate two images named I_{ob} and I_r (image for object and image for rules), on these images it is necessary to apply a new pixel distribution by means of the following steps.

Step 1: Decompose I_{ob} and G_o images on three RGB channels.
Step 2: Store each channel in a database with its respective channels table, saving their original coordinates, pixels values and a unique identifier.
Step 3: Apply a reordering function on R, G and B channels, which favors a heterogeneous distribution of them.
Step 4: Store in different tables channels that have undergone reordering.
Step 5: Generate a new image from reordered channels.

A new image is obtained (intermediate image) with reordered pixels, the message is inserted in I_{ob}, while recovery rules are inserted in I_r image. The intermediate image is the product of applying redistribution equations to original pixels positions, without losing the reference of its original position. When the message is stored along with recovery rules, images are then merged I_{ob} and I_r through records stored in the database with their original positions to form the final image (stego-image). For this purpose, it has been proposed to assign an identification number for each pixel in every channel. Figure 3 shows the data diagram of the cover image analysis process.

Figure 3 Redistribution of pixels and data insertion



Relationship between original and modified image

Source: Own Elaboration

Data recovery and analysis. Upon completion of insertion of the message and recovery rules of it from the cover image, it is necessary to recover data and to complete the analysis with quality metrics to verify if data is recovered correctly and if a visual test can be approved.

To retrieve the message of the stego-image is necessary to recover the rules that allowed pixels distribution of the cover image and generate the intermediate image. The rules zone is read from the intermediate image. The vector obtained from the fractal is read to apply the first phase of data decoding. The fractal creates a relation of the alphabet with the ASCII 64 base original symbols for encoding. Data is extracted, and the first decoding is generated, later the message is decoded in 64 base and finally decompressed with LZ77. Stego-images resulting from the steganography process are evaluated through MSE, PSNR and SSIM (Wang, Simoncelli & Bovik, 2003) metrics to determine if they present visual deterioration concerning the cover image.

Experimental Results and Discussion

The proposed steganography method is assessed through the insertion of different types of files, which are text, image, audio, video and an executable Windows le, with characteristics shown in Table 1.

Table 1 Characteristics of files inserted in cover images

Type of file	Size in bytes	Compression ratio
Text file	524,237	232.836 %
Image PNG file	181,737	-69.939 %
Audio ogg file, 54 seconds	183,238	-76.039 %
Video 3gp file, 18 seconds	183,238	-76.039 %
Executable Win32 file	190,976	101.690 %

Source: Own Elaboration

Data files shown in Table 1, should be inserted into RGB images digital with bmp format and 512x512 pixels dimension. Tested images are Lenna, peppers, Barbara, airplane and Goldhill. Five copies are created per image; each copy will store a different file (text file, ogg audio file, video 3gp file, executable Win32 file and image PNG. It can be seen in Table 1; compression ratio is positive for non-compressed data such as text file and executable le. While compression ratio is negative when applied to objects whose structure has been compressed previously, this comes as a disadvantage, but it should be noted that Mac OS Roman coding allows easily to manipulate this type of objects without having different extracted files components such as audio and video.

Figure 4 Cover images used for assessment

a) Lenna b) Peppers c) Barbara d) Airplane e) Goldhill

Source: Dataset USC-SIPI (USC-SIPI, 2017)

Fractals (Hausdorf dimension) used for the final alphabet generation in the last coding phase of the message inserted in cover images are Koch Snowflake, Cantor Set, Contour of Gosper Island, Border of the curve terdragon, Two-dimensional Cantor, Vicsek fractal, Quadratic Curve Koch 1, Quadratic Curve Koch 2, Tree of three branches, Pascal's triangle module 3, Pascal's triangle, Hexacop, Pentacop, Sierpinski rug, Greek cross 2D fractal and Hilbert 3D curve (Kenneth, 2003).

The proposed method is coded in Python 2.7, with OpenCV library for Mac OS X version 10.13.6. Redistribution equations for R channels is $abs\,(x \times id \times 2) - (x \times tan \times id) \times abs((tan \times x)/4)$ and for channels G and B is $abs((id \times y) \times (id \times x)) \times -8$, where id is an unambiguous identifier number for each pixel x, y are pixel coordinates and abs is an absolute function value. Quality metrics with which stego-images have been measured are MSE, PSNR and SSIM in MATLAB version 2016. The database engine that has been used is SQLite 3 for data temporary storage. The injection technique chosen is 3 bits LSB.

Table 2 contains results obtained when applying quality metrics. As it can be seen in Table 2, in stego-images with text file inserted, they exceeded 40.000 dB, stego-image that showed greater stability is Lenna, because it exceeds 41.332 dB of PSRN, concerning MSE scores, it goes from 4.779 to 5.332 points. SSIM metric shows a score that bread from 0.962 to 0.998, with Lenna being the best evaluated image in all tests.

For audio data, stego-images present a more significant deterioration because of the amount of data now of inserting pixels numeric values has been somewhat higher impact than embedding pure text. Results are competitive, because PSNR ranged from 39.700 to 40.400 dB, in the case of MSE it is between 6.200 points to 6.890 points, and for SSIM did not fall below 0.950 points, with the best evaluated image being Lenna. The third object inserted was a video le, this presents similar results to the audio file because its size in bytes is technically the same.

To the executable les, the results are better than audio and video les. Finding a higher number of redundant zones and PSNR results oscillate between 48.900 dB and 49.000 dB, meanwhile, MSE oscillates between 0.700 to 0.800 points and SSIM goes from 0.999 to 0.994. Finally, now of inserting Lenna image, lower scores are achieved because the final number of bytes is higher than other testing objects. Even though PSNR levels remain above 38.000 dB, MSE has been higher than 8.600 points and SSIM never dropped below 0.940 points. In all stego-images, 100 % of the embedded data is recovered.

Table 2 Results obtained by inserting objects in cover images

Stego image	Embebed file	Average PSNR	MSE	SSIM	Bits per pixel
Lenna	Text	41.332	4.790	0.998	5.332
Peppers	Text	40.807	5.398	0.997	5.332
Barbara	Text	40.861	5.332	0.997	5.332
Airplane	Text	40.856	5.337	0.962	5.332
Goldhill	Text	40.877	5.312	0.984	5.332
Lenna	Audio	40.211	6.200	0.997	1.863
Peppers	Audio	39.744	6.896	0.996	1.863
Barbara	Audio	39.803	6.802	0.991	1.863
Airplane	Audio	39.786	6.829	0.952	1.863
Goldhill	Audio	39.821	6.774	0.981	1.863
Lenna	Video	40.211	6.200	0.997	1.863
Peppers	Video	39.744	6.896	0.996	1.863
Barbara	Video	39.803	6.802	0.991	1.863
Airplane	Video	39.786	6.829	0.952	1.863
Goldhill	Video	39.821	6.774	0.981	1.863
Lenna	Win32	49.564	0.719	0.999	1.942
Peppers	Win32	49.947	0.828	0.999	1.942
Barbara	Win32	48.961	0.825	0.999	1.942
Airplane	Win32	48.949	0.828	0.994	1.942
Goldhill	Win32	48.967	0.824	0.996	1.942
Lenna	Image PNG	39.158	7.895	0.997	1.848
Peppers	Image PNG	38.767	8.637	0.996	1.848
Barbara	Image PNG	38.822	8.526	0.989	1.848
Airplane	Image PNG	38.822	8.528	0.945	1.848
Goldhill	Image PNG	38.842	8.487	0.977	1.848

Source: Own Elaboration

Verifying data provided by authors in Stoyanova & Tasheva (2015), Chuang & Chang (2006) and Sun, Lu, Wen, Yu & Shen (2013) (see Table 3), it is possible to determine that message analysis for concealing with the cover image allows greatly to exceed the amount of embedded data. Coding and compression processes allow to reduce the amount of data in a significant way especially in the case when it is executing on pure text. It is important to point out that when an audio or video file is hidden, it is not possible to reduce its size, because they contain a previous compression treatment which means that it is no longer possible to reduce their size without data loss. On the other hand, processing that has been applied to this type of files has allowed its embedding in the cover image satisfactorily and without loss of data. In Stoyanova & Tasheva (2015) PSNR results are superior to the proposed scheme but represented load doubles inserted information.

Table 3 Comparison between achieved results and works of the literary review

Author	Stego image	Payload in bits	Average PNSR
Sun, Lu, Wen, Yu & Shen (2013)	Lenna	169,250	28.92
Sun, Lu, Wen, Yu & Shen (2013)	Airplane	168,388	28.11
Sun, Lu, Wen, Yu & Shen (2013)	Goldhill	172,771	28.74
Chuang & Chang (2006)	Lenna	114,688	30.71
Chuang & Chang (2006)	Airplane	114,688	30.99
Chuang & Chang (2006)	Goldhill	114,688	30.49
Stoyanova & Tasheva (2015)	Lenna	1,966,080	54.89
Ouyang, Park & Kau (2016)	Lenna	262,144	72.97
Proposed	Lenna	4,193,896	41.33
Proposed	Airplane	4,193,896	40.85
Proposed	Goldhill	4,193,896	40.87

Source: Own Elaboration

Conclusions

In steganography, representation of the hidden message is important, because its size reduction (data without previous compression) together with an image with printable characters such as 64 base ASCII allows its transport and extraction without characteristics loss, facilitating data handling. As it has been shown in other studies, pixels regrouping (intermediated image) reduces visual deformation in the stego-image. Temporary redistribution of pixels and insertion of information in this stage allows protecting the original injection scheme, which helps a classic data recovery technique to avoid obtaining information directly from a stego-image.

Three important points are observed in favor of the proposed scheme. First, when dealing with messages as alphanumeric type chains, manipulation of their elements allows efficiently transporting information without the need of having to manipulate many components of data for embedding, because it is only necessary to manage symbols obtained from its alphabet. Secondly, compression usefulness allows increasing the amount of data that is represented for embedding and aims at two purposes: increasing the amount of data that can be stored in a carrier image, and data is not stored in plain text. Finally, a third point is pixels redistribution, which helps to create a security layer by breaking insertion logic of the steganographic technique, adding to the selection of specifics pixels to decrease the impact of mod

ifications that were made on a stego-image.

Steganography allows people to exchange information in a secure way, because it reduces the probability that possible attackers detect that a process of exchange of confidential information is being generated because the information is transported undercover in a digital object. of which it is difficult to suspect that it carries a hidden message.

As future work, it is desirable to detect redundant areas in the cover images for data insertion, and in this way, the quality of the stego-image would be improved.

Acknowledgement

We are grateful to the Tecnológico de Estudios Superiores de Jocotitlán for the support given in carrying out this research project.

References

Abbas, T.A., & Hamza H. K. (2014). Steganography Using Fractal Images Technique. Computer Science; IOSR Journal of Engineering (IOSRJEN),4, 52-61.

Velasco-Bautista, C. L., López-Hernández, J. C.,Nakano Miyatake, M., & Pérez-Meana, H. M. (2007). Esteganografía en una imagen digital en el dominio DCT. *Científica*, *11*(4), 169-176.

Chuang, J. C., & Chang, C. C. (2006). Using a simple and fast image compression algorithm to hide secret information. *International Journal of Computers and Applications*, *28*(4), 329-333.

Das, S., Das, S., Bandyopadhyay, B., & Sanyal, S. (2011). Steganography and Steganalysis: different approaches. *arXiv preprint arXiv:1111.3758*.

Desai, H. V., & Desai, A. A. (2014). Image steganography using mandelbrot fractal. *International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR)*, *4*(2), 71-79.

Eswari, G. S., Leelavathy, N., & Rani, U. S. (2014). Fractal image steganography using non-linear model. *International Journal of Innovative Research in Computer and Communication Engineering*, *2*(1), 2644-2649.

Geetha, G., & Thamizhchelvyb, K. (2016). Application of chaos and fractals in Image steganography a review. *International Journal of Control Theory and Applications*, *9*(45), 95-106.

Hopcroft, J. E., Motwani, R., Ullman, J. D., & Alfonseca, M. (2007). *Introducción a la teoría de autómatas, lenguajes y computación*. Pearson educación.

Jung, K. H., & Yoo, K. Y. (2014). Three-directional data hiding method for digital images. *Cryptologia*, *38*(2), 178-191.

Katzenbeisser, S. & Peticolas, F.A (2000). Information Hiding Techniques for Steganography and Digital Watermarking. 1st ed. Artech House, 21-24.

Kenneth, F (2003). Fractal Geometry: Mathematical Foundations and Applications. John Wiley & Sons.

Nehete, D., & Bhide, A. (2014). Skin tone based secret data hiding in images. *International Journal of Current Engineering and Technology*, 18-24.

Ouyang, L., Park, J. H., & Kaur, H. (2016). Performance of efficient steganographic methods for image and text. *Journal of Advances in Information Technology Vol*, *7*(1).

Rafat, K. F., & Hussain, M. J. (2016). Secure steganography for digital images. *International Journal of Advanced Computer Science and Applications*, *7*(6), 45-59.

Rijmen, V., & Daemen, J. (2001). Advanced encryption standard. *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, 137-139.

Salomon, D., & Motta, G. (2010). *Handbook of data compression*. London; New York: Springer.

Singhal, S., & Rathore, R. S. (2015). Detailed Review of Image Based Steganographic Techniques. *IJCST*, *6*, 93-95.

Sofloo, A.G., Aghayi, M (2010). Steganography in the last significant bit. Journal of Innovative Research in Engineering Sciences, 8-14.

Stoyanova, V., & Zh, T. (2015). Research of the characteristics of a steganography algorithm based on LSB method of embedding information in images. *Machines. Technologies. Materials.*, *9*(7), 65-68.

Sun, W., Lu, Z. M., Wen, Y. C., Yu, F. X., & Shen, R. J. (2013). High performance reversible data hiding for block truncation coding compressed images. *Signal, Image and Video Processing*, *7*(2), 297-306.

University Souther California (2021-31). USC-SIPI Image Database http://sipi.usc.edu/database.

Wang, Z., Simoncelli, E. P., & Bovik, A. C. (2003, November). Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003* (Vol. 2, pp. 1398-1402). IEEE.

Zhang, H., Hu, J., Wang, G., & Zhang, Y. (2011, September). A steganography scheme based on fractal images. In *2011 Second International Conference on Networking and Distributed Computing* (pp. 28-31). IEEE.