International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 03, March-2017

Image Steganography Using LSB and Software Agents

Diego De Lima Nascimento
Department of Computer Science
ICMC - University of São Paulo
São Carlos - SP, Brazil

Fagner Roger Pereira Couto
Department of Computer Science
Federal University of São Carlos
São Carlos - SP, Brazil

Luciano Zamperetti Wolski
Department of Computer Science
Mato Grosso State University
Barra do Bugres - MT, Brazil

Igor Antonio Kuhnen
Institute of Physics
Mato Grosso Federal University
Cuiabá - MT, Brazil

***Abstract*— Nowadays the society provides several digital means and technologies in order to facilitate the daily tasks, increasing the need in the incentive for studies and researches that help the communication and protection of data. This data can contain simple and personal message without major importance and redundant, or highly valuable and confidential for some corporation. The objective of this work is to present an information protection approach using digital images (lossless), Least Significant Bit insertion technique and software agents. Finally, the algorithm was able to hide a substantial amount of data without increasing the image size and without degrading, since only 1.04% of the image is altered.**

***Keywords*— Steganography; Digital Security; Images; Software Agents.**

I. INTRODUCTION

Information security was always present in society, in ancient times cryptographic methods were used to mask important messages. Since then, the society uses several methods in different contexts to protect more sensitive data, among these methods stands out cryptography and steganography.

According to [1], in the area of digital security there are numerous people who try to "trick" the technology to have access to personal files. On the other hand, there is the other side, which seeks the development of technologies and techniques to protect this information, causing a great impact on research and development. In this context, there is a challenge in the authenticity of communication of small or large work groups in universities or private institutions, mainly because they work with sensitive information. In this scenario, steganography and cryptography help to fill several problems quickly and safely.

In this sense, the objective of this research is to present a digital security approach for small and large workgroups that need to protect precious information with copyright. In this case, the work uses steganography with the LSB insertion technique, AES encryption and software agents that are responsible in performing tasks.

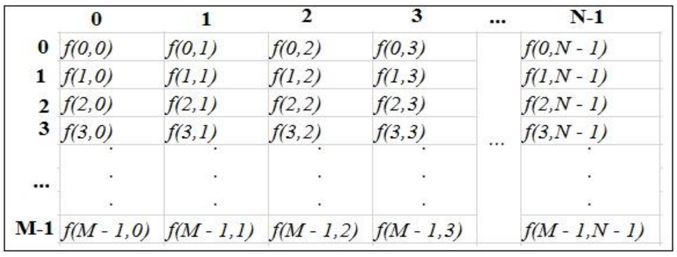
The rest of this paper is organized as follows. Section II describes terminology and aspects of digital image, steganography and software agents. Section III related works. Section IV presents the methodology. Section V presents the results. Finally, in section VI the conclusion and possibilities for future work.

II. BACKGROUND

A. Digital Image

An image is described from a visual representation of something or some form in which it is desired to replicate, be it abstract or concrete [2]. However, this concept could not be easily applied without development in the field of science, precisely in astronomy, which led to the practicality of obtaining digital images [3].

Digital image are arrays, where the image is described by the function *F(x, y)*, the coordinates being integers where each point of the image has a certain location and value, called Picture Element (pixel). The images are called digital, when the intensity values of *F* in the coordinates *(x, y)* have finite and discrete quantities of pixels [3] (Fig 1).



	0	1	2	3	...	N-1
0	f(0,0)	f(0,1)	f(0,2)	f(0,3)		f(0,N - 1)
1	f(1,0)	f(1,1)	f(1,2)	f(1,3)		f(1,N - 1)
2	f(2,0)	f(2,1)	f(2,2)	f(2,3)		f(2,N - 1)
3	f(3,0)	f(3,1)	f(3,2)	f(3,3)	...	f(3,N - 1)
...	. . .	. . .	. . .	. . .		. . .
M-1	f(M - 1,0)	f(M - 1,1)	f(M - 1,2)	f(M - 1,3)		f(M - 1,N - 1)

Fig. 1. Digital image representation in array M x N (M rows e N col).

The size that an image occupies on disk is proportional to the size of bits that each pixel has, however, this value is dependent on the color system that the image uses. Some models available are Grayscale (1bit), Red-Green-Blue (RGB) (3bits), Cyan-Magenta-Yellow-blacK (CMYK) (4bits). The size of an image can be calculated by the following equation (1), where the pixels of the vertical and horizontal are multiplied with the depth and bytes per component [4].

$$Size = H * W * D * BPC \qquad (1)$$

The storage peripherals available on the market are of relatively large capacity, to the extent that the size occupied by a FullHD image becomes irrelevant. However, the amount of data produced by devices is extremely high, a large part of this data is from the Web (news, social networks, among others), resulting in a large amount of duplicate and shared data online. It is important to note that reducing file size allows more data to be stored in a certain amount of memory space. Also resulting in a reduction in the time required to be sent over the Internet or downloaded from Web pages.

This problem can be mitigated through the matrix values of an image. In this context, algorithms (e.g., JPEG, PNG) are used to compress such data, since with compression the size in bytes of a graphic file is minimized without degrading image quality to an unacceptable level.

B. *Steganography*

Steganography is known as a method of hiding information (independent of information), making it unintelligible. Steganography derives from the Greek, steganos = covered and graphien = writing, thus defined as a covert technique of writing.

The main characteristic of steganography is defined as a technique that allows and ensures the safety of virtually any file, regardless of its nature, in the form of communication. According to [5] the technique is present in two strands, a search or improvement for greater data protection and the other, a way to circumvent this security, removing the integrity of the file to have access to this data.

C. *Least Significant Bit*

Least Significant Bit (LSB) is a data insertion technique. The least significant bit is the one to the right of each binary chain, which makes this technique feasible, since when inverted the value produces minimal changes, being imperceptible to human vision.

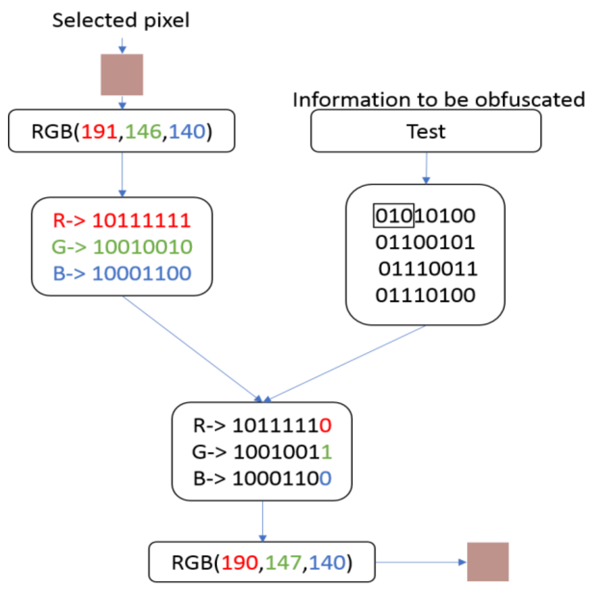


Fig. 2. LSB technique.

Consider a scenario where the user wants to steganography a particular message in a digital image. Fig 2 depicts how the LSB technique occurs. First, a pixel of the image is selected for insertion of the first three bits of the information to be hidden. After the last three bits of the RGB are exchanged for the information, the modified pixel will be inserted back into the image. It is important to note that this process continues until all the information is inserted in the image.

D. *Software Agent*

A software agent has a very broad definition, however, [6] conceptualized as an entity that has such characteristics:

- Acting in an environment.
- Be able to interact with other agents.
- It has a goal.
- Being able to identify changes in the environment.
- Has a part of representing the environment.
- Has skills and can offer services.

According to [7] a system can be considered an agent based on some properties, autonomy, social ability, reactivity and proactivity. But in general, they are programs with instructions that have decision-making power to perform a certain task.

III. RELATED WORKS

There are several methods applied in the area in question, such methods presented below, use some techniques discussed in this work. [8] In this study called Chameleon, a bilingual open source system was developed under the Genaral Public GNU License (GPL) using the Java programming language that allows the hiding of images (JPG or PNG), texts or another binary file within another image (The exit of the object of concealment will be PNG), the technique used was the LSB from a masking based on periodic or non-periodic displacement, in addition to being linear or random.

[9] Proposes to use the LSB method in conjunction with AES encryption, in order to determine if the information entered is completely randomized. The author noted through tests that the coding process has altered the quality of the image, but ensuring that the changes are insignificant with a naked eye. However, this process has altered the final image size, unlike the approach presented in this work, which, although similar, ensures that the final image size remains the same after the process.

[10] Presents the design and implementation of steganography using DWT (Discrete wavelet Transform), Huffman coding and modified LSB technique in RC4 algorithm. In the approach, the information to be hidden is compressed using the Huffman coding, then the bits are incorporated into the LSB and applied to high frequency regions of the coverage image, using an arbitrary order generated by the RC4 algorithm.

On the other hand, [11] describe a technique of steganography in order to generate encrypted messages from free context grammars. The work of the author uses the animation of the frames to hide data in the gestures of the characters.

IV. METHODOLOGY

Implementation of the algorithm was performed in the Java programming language using the LSB steganography technique and block cryptography AES (128 bits). In this context, an architecture based on software agents was developed, through the Jade framework that works as a development platform, receiving the requests and processing their respective requisitions.

A. *Implementation*

The LSB technique applied in the approach uses images lossless. For the insertion and retrieval of information, 2 bits of the Red channel, 1 bit of the Green channel and 1 bit of the Blue channel were used, totaling 4 bits per pixel. In this sense, the possible amount of bytes to be inserted into an image is described in equation (2).

$$Max = \frac{(Width * Height)}{2} - 10 \qquad (2)$$

An important constant in the technique is the criterion of reading and stopping applied in the object of coverage, some projects mentioned use acronyms or initials to indicate the beginning and end of reading. However, the read and write criteria used in this work is based on the size of the files to be inserted. Therefore, the *n* files indicated for cover-up are compressed into a zip file, in which it will be used to check the end of the reading. In this case, a 10-pixel header is made to indicate the size of the zip that will be written to the coverage object, since the reading of the bytes in the image will be equal to the size of the compressed file. Consequently, the compression process in addition to fulfilling its purpose expands the amount of possible data to be hidden in the image. To improve security, the method allows the use of 128-bits AES encryption that helps protect compressed files before they are inserted into the coverage image. In addition to security, the sequential insertion of data, i.e. pixel after pixel, is not used. In this approach we use a pseudorandom number generator algorithm (PRNGs).

According to [12] its output satisfies many conditions that can be expected from random numbers, although the generator operates deterministically (equation 3).

$$X_{n+1} = (a * X_n + b)\ (mod\ m),\ (X_n)\ n \geq 1 \qquad (3)$$

Since the parameter *a*, *b* and *m* are chosen, the pseudo-random sequence will only depend on the initial value of X_0. The characteristic of PRNGs is that they produce a long sequence of random numbers from an initial input, the so-called seed, thus defining a completely deterministic algorithm for various applications. Before the definition, the randomness of the pixels is generated by the algorithm given to the seed described in equation (4).

$$Seed = \sum_{i=1}^{10} (MaxPixels\ \ mod\ i) + i \qquad (4)$$

The Fig 3 demonstrates the insertion using this approach. It is important to note that in the illustration the file is already compressed and also the insertion of the header was performed, since the insertion process of the header is similar to the inner loop of the algorithm. Accordingly, in Fig 4 it is exemplified how the reverse process is performed.

```
Algorithm 1 Insert Values
Input: $image, zip$
$pixelNumber \leftarrow 10$
 $pixelsOrder \leftarrow generateList()$
 while $byte$ = $(zip.read()$ != $-1)$ do
   $i \leftarrow 0$
   repeat
     if $i == 0$ then
       $value = byte >> 4$
     else
       $value = byte$
     end
     $pixelNumber \leftarrow pixelsOrder.get(++pixelNumber)$
     $pixel \leftarrow image.getRGB(pixelNumber \% image.getWidth(), pixelNumber / image.getWidth())$
     $pixel \leftarrow pixel \& 0xFFFCFEFE$
     $value \leftarrow value \& 0x0F$
     $pixel \leftarrow pixel | (value \& 0x01)$
     $pixel \leftarrow pixel | ((value >> 1) \& 0x01) << 8$
     $pixel \leftarrow pixel | ((value >> 2) \& 0x03) << 16$
     $image.setRGB(pixel)$
     $i++$
   until $i = 2$;
end
```

Fig. 3. Insertion method.

```
Algorithm 2 Get Values
Input: $image, zipSize$
$pixelNumber \leftarrow 10$
 $pixelsOrder \leftarrow generateList()$
 while $pixelNumber < (zipSize * 2 + 10)$ do
   $byte[] \leftarrow 0$
   $i \leftarrow 0$
   repeat
     $pixelNumber \leftarrow pixelsOrder.get(++pixelNumber)$
     $pixel \leftarrow image.getRGB(pixelNumber \% image.getWidth(), pixelNumber / image.getWidth())$
     $pixel \leftarrow pixel \& 0x00FFFFFF$
     $value \leftarrow (pixel \& 0x01)$
     $value \leftarrow value | ((pixel >> 8) \& 0x01) << 1$
     $value \leftarrow value | ((pixel >> 16) \& 0x03) << 2$
     if $i == 0$ then
       $byte \leftarrow value << 4$
     else
       $byte \leftarrow byte | value$
     end
     $i++$
   until $i = 2$;
   $out.write(byte)$
end
```

Fig. 4. Get method.

In Fig 5 the operation of the approach is exemplified. The process starts when the user indicates the object of hiding, that is a picture (lossless). The next step is to indicate the *n* files to be hidden, emphasizing that the use of AES encryption is optional and the total size of these files must satisfy equation (2). Finally, the zip file is inserted into the image and returned to the user.

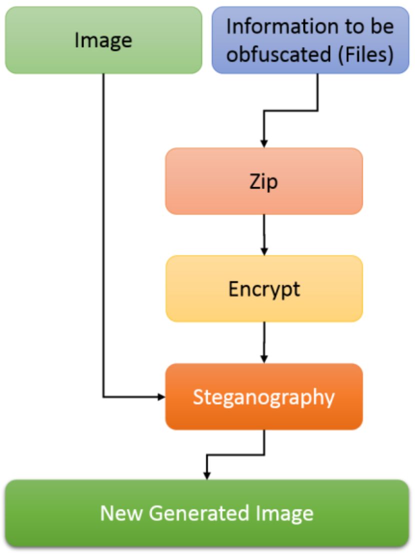


Fig. 5. Implementation Flow.

The decision-making of the main functions of the algorithm is performed by the software agent, emphasizing that the implemented architecture is characterized as a simple reflection agent because it does not require great interaction with the user and is relatively autonomous.

According to [13] this agent is based on a set of condition-action rules set by means of the perception of the environment, since, the agent performs a pre-established action, thus being purely reactive agents.

When the user needs to perform some action (insert image, insert files, steganography, among others) the system sends a request to the agent to execute the task and later, the agent responds to the user whether the task was executed or not (Fig 6).

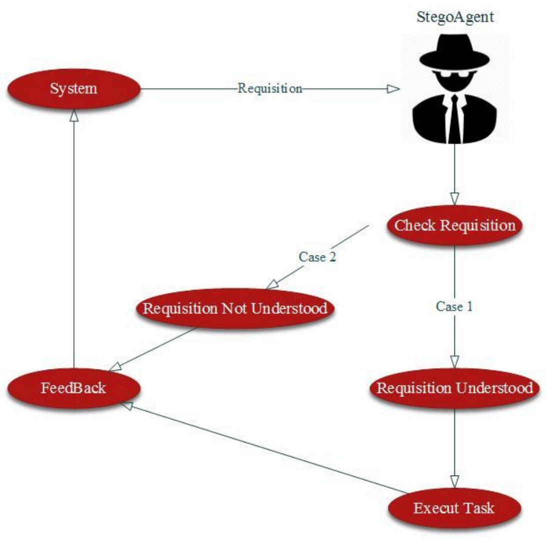


Fig. 6. Agent schema.

V. RESULTS

To analyze the algorithm was carried out a quantitative study by means of a tool developed by [14]. This tool uses the non-parametric test x^2 that has the objective of analyzing two frequencies and adding their discrepancies to determine the amount of times that this sequence occurs, in this case, can be used to verify the occurrence of 0 or 1 in the least significant bit of the Image [9]. The statistic is given by equation 5.

$$x^2 = \sum \frac{(O_i - E_i)^2}{E_i} \quad (5)$$

Equation (5) generates a higher or lower value depending on frequency, but always positive, so O_i is the observed frequency of event *i*, and E_i is the expected frequency of event *i*.

To illustrate the algorithm's performance, the steganography was performed in a BMP format image with a size of 1,265,678 bytes and a resolution of 750x562 (Fig 7). Thus, a text document in the format docx with the size of 32,768 bytes was inserted to be obfuscated in the image. After the steganography process applied by the algorithm, a new image was generated containing the hidden information.

Fig. 7. Image analyzed.

The new image generated was analyzed by the statistic of x^2 and the results are presented in Fig 8, where Scenario (A) refers to the original image and (B) the steganography image.

The red line indicates the test of x^2, where the closest to 1 indicates a great chance of a hidden file, the green line analyzes a block of 128 bytes with the average insertion of the LSB and if close to 0.5 indicates possible hiding. Finally, each vertical of the blue line represents 1 kilobyte of embedded data [14]. In front of the obtained results, it is noticed that in both images, the difference of the scenes is minimal and has weak evidence of something hidden.

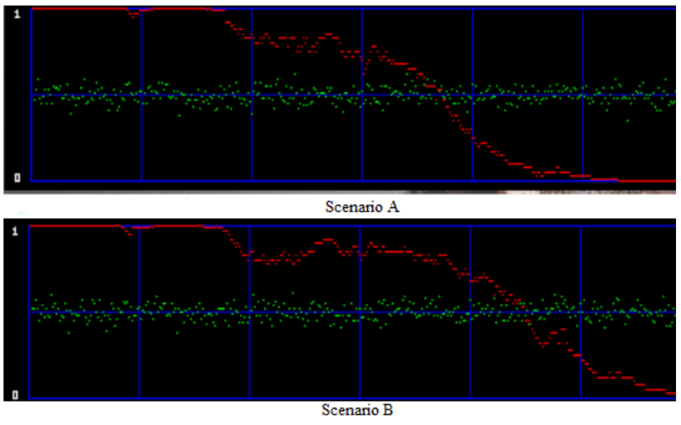


Fig. 8. Results.

Through the presented method, the algorithm manages to mask without modifying the size and without degrading the image. In this case, when there is a total modification in the image, i.e all pixels are modified, the described method will change only 1.04% of the total color of the image. According to [15] the human being can only identify changes in images when their modification size is greater than 3%.

VI. CONCLUSION AND FUTURE WORK

In the method implemented, small values are used that make up the image that after its modification still remain insignificant to those who observe. Therefore, the algorithm is able to use the best practices for which the steganography was created, keeping the information imperceptible in objects, emphasizing that another important goal reached by the method is hiding a substantial amount of data without increasing the size of the image. In addition, software agents allow the use of the approach in a network of computers, that is, it is possible to use the functionalities implemented through the communication of agents in the network. In view of the above, the algorithm can aid in the security of messages or copyrighted files produced in universities or private institutions, thus justifying the importance of the approach developed, so the work besides providing protection for digital data can also encourage Research related to digital security.

A proposal to improve the approach is related to the architecture of software agents. So, it is possible to implement rules for an environment that needs to exchange information simultaneously. In this case, they can be used as mediators so that a network is functional and exclusive.

REFERENCES

[1] Julio, E. P., Brazil, W., & Neves, C. V. (2007). Esteganografia e suas Aplicações. Livro de Minicursos do SBSEG. Rio de Janeiro: Sociedade Brasileira de Computação, 7, 54-102.

[2] Santaella, Lucia and NOTH, Winfried. Imagem: cognição, semiótica, mídia. São Paulo: Imuminuras,1997.

[3] Gonzalez, R. C., Woods, R. E. (2009). *Digital image processing*. 3rd Edition. New Jersey: Prentice Hall, 2008. ISBN 978-0131687288.

[4] Scuri, A. E. Fundamentos da imagem digital. Tecgraf/Puc-Rio, 2002.

[5] Rocha, A., Goldenstein, S., Costa, H., & Chaves, L. (2004). Camaleão: privacidade e segurança na internet por esteganografia em imagens. In: *WebMedia & LA-Web–Software and Tools 2004 Joint Conference*.

[6] Ferber, J. Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[7] Wooldridge, M., Jennings, N. Intelligent Agents: Theory and Practice. Cambridge University Press: Knowledge Engineering Review, vol. 10-2, p. 115, 1995.

[8] Rocha, A. R. Camaleão: um software para segurança digital utilizando esteganografia. Monografia de final de curso, Universidade Federal de Lavras - UFLA, Minas Geras. 2003. Disponível em: <http://www.ic.unicamp.br/~rocha/sci/stego/src/monografia.pdf.>

[9] Nurhayati and S. S. Ahmad, "Steganography for inserting message on digital image using least significant bit and AES cryptographic algorithm," 2016 4th International Conference on Cyber and IT Service Management, Bandung, 2016, pp. 1-6.

[10] H. Gupta and P. Mahajan, "Improvisation of security in image steganography using DWT, Huffman encoding & RC4 based LSB embedding," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2016, pp. 523-529.

[11] Sung, A. H.; Tadiparthi, G. R.; Mukkamala, S.. Defeating the current steganalysis techniques (robust steganography). In: null. IEEE, p. 440, 2004.

[12] Röck, A. (2005). Pseudorandom number generators for cryptographic application.

[13] Russell, S., & Norvig, P. (1995). Artificial intelligence: a modern approach. Prentice-Hall, Egnlewood Cliffs. p. 46.

[14] Guillermito. (2007). A few tools to discover hidden data. Disponivel em: <http://guillermito2.net/stegano/tools/>.

[15] Wayner, P. (2002). Disappearing Cryptography: Information Hiding: Steganography & Watermarking. 2nd. ed., Morgan Kaufmann, San Francisco, Califórnia.