

# Investigation of the Influence of a Template Matrix on the Embedding of Information in an Image

*Dimitar Todorov<sup>1</sup>, Zheyno ZheyNov<sup>1</sup>, Hristo Valchanov<sup>1</sup>, Milena Karova<sup>1</sup>, Ivaylo Penev<sup>1</sup>*

1 – Technical University of Varna, Department of Computer Science and Engineering, 9010, 1 Studentska Street, Varna, Bulgaria

Corresponding author contact: [ivailo.penev@tu-varna.bg](mailto:ivailo.penev@tu-varna.bg)

**Abstract.** Steganography is a modern approach to protect classified data against malicious attacks and misuse. Presented, accordingly, in this paper is a novel method for steganographic embedding of information. A template matrix is used for screening the original message embedded in an image. The efficiency of the steganographic embedding depends on the length of the message. The particular dependency is, therefore, the primary focus of the proposed work. The end results of the experiment were extremely satisfactory with the percentage of successfully retrieved messages being more than 90%, and the size of the processed images with embedded messages being fully acceptable and capable of being used in a communication environment.

**Keywords:** steganography, template, matrix, message length

## 1 Introduction

Steganography deals with composing hidden messages so that only the sender and recipient know that the message even exists. Considering that no one but the sender and recipient are cognizant of the existence of the message, this does not attract unwanted attention (Grebennikov, 2019; Stanev, S. & Szczypiorski, K., 2016). Steganography has been used since ancient times and these ancient methods are called physical steganography. Some examples of these methods are messages hidden in the body of messages, messages written in secret inks, messages written on envelopes in areas covered with stamps, and more. Modern methods of steganography form digital steganography. These methods include hiding messages in the noise pixels of the images, embedding a message in random data, embedding photos with the message in video files, etc. (Gribunin et al. 2009; Stanev et al, 2012; Stoyanov et al, 2016).

There are many different types of methods for using steganography. The different categories are classified below according to the file formats used for steganography techniques (Gribunin et al. 2009; Fridrich, 2010, 2004).

- Text steganography
- Steganography with images
- Audio steganography
- Video steganography

The emphasis of the present study is on steganography with images or hiding useful information by embedding it in the image. In this model, the image is a carrier of information. In the style of steganography, it is important that the changes made to the image carrier, after the incorporation of the useful information, do not lead to its noticeable change, i.e. the original image and the modified steganographic one should have almost no visual differences. Otherwise, the system would arouse external interest as a result of the perceptible difference.

Different steganographic methods and algorithms have been developed and tested (Lokhande, 2014; Zhelezov, 2016; Zhelezov & Paraskevov, 2015). The template used for embedding information has a strong influence on the effectiveness of the steganography. A coherent explanation of the emphasis is provided for example in (Kruus et al, 2003; Kordov & Stoyanov, 2017; Rani & Chaudhary, 2013; Neil & Johnson, 1998; Ross & Petitcolas, 1998; Sarita & Kumar, 2014). Advantage is taken in the current

paper of a template matrix in accordance with the primary purpose of the study to explore the influence of the template matrix on the performance of the steganographic processes.

## 2 Layout

### 2.1 Bit pixel representation

A widely used method of embedding is the introduction of the message bit by bit, i.e. the message is converted from a text to a bit message, and then its bits are imported into those of the image, replacing some of them. Thus, the last bit appears to be the most frequently used item. An image with a depth of 24 bits contains pixels with 3 by 8 bits for each component color red (R - red), green (G - green) and blue (B - blue) (Fridrich, 2010). For the experimental staging of this study, BMP images were selected for use. This is an uncompressed image file format developed by Microsoft. In addition to the three RGB components in its 32-bit version (32 bpp), this format also contains a fourth component A, which determines the brightness of the pixel. The purpose of the experimental staging is to insert the message that needs to be hidden within this particular component (Fig. 1).


<i>one pixel</i>		<i>int format</i>	<i>bit format</i>
	<i>R component</i>	242	11110010
<i>red</i>	<i>G component</i>	12	00001100
	<i>B component</i>	12	00001100
	<i>A component</i>	255	11111111

Fig. 1. Representation of a pixel into BMP file

### 2.2 Process of embedding a message into an image

The process of embedding information (i.e. the text of the message) into the image involves the following steps:

1. Converting the message from text format to binary and determining its size;
2. Image splitting - carrier of a matrix of its constituent pixels;
3. Embedding the converted message in the image.

<i>Message: "text"</i>		<i>Dec. number in ASCII</i>	<i>Binary representation</i>
<i>by symbols:</i>	<i>t</i>	116	01110100
	<i>e</i>	101	01100101
	<i>x</i>	120	01111000
	<i>t</i>	116	01110100

Fig. 2. Representation of the text into binary format

The process of converting the text message is simply converting the characters in the string into a binary form with a length of 8 bits, with ASCII coding table - Fig.2. All characters are then returned to the string, but in binary form. Thereby, a result is obtained - a string of bits. The length of the message should be taken into account, as it will be embedded first before the message itself, and is determined by the following simplified formula:

$$L_{msg} = \frac{L_{bit}}{8}, \quad (1)$$

where  $L_{msg}$  is the length of the message, and  $L_{bit}$  is the length of the entire string of bits generated during the conversion of the message.

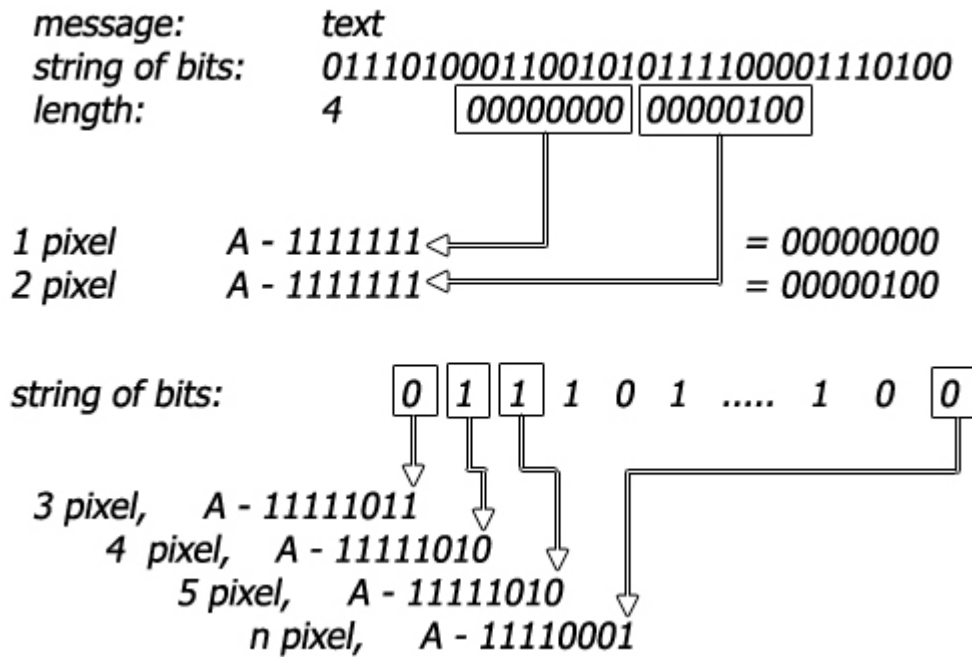


Fig. 3. Process of embedding the message

The image is decomposed into a composite matrix of pixels, each pixel being in the form of Figure 1, and the emphasis is on the bits of component A. The first two pixels are used to embed the length of the message. Their entire length of bits is used, i.e. 2 by 8 bits - a total of 16 bits. This would allow the encoding of 65536 message lengths. The bits of component A of the first two pixels are replaced by one and the other half of the bits by the message length - Fig.3. The last bit in each subsequent pixel for component A is replaced by the next bit of the string of bits of the text message, starting from the first bit in the string of the text message - Fig.3. The maximum length of the message  $L_{tMAX}$  that can fit in the media image depends on its resolution (i.e. its dimensions -  $I_{width}$  and  $I_{height}$ ):

$$L_{tMAX} = \frac{(I_{width} \cdot I_{height}) - 2}{8} \quad (2)$$

For example, for an image measuring 1024x768 pixels, a message with a maximum length of 98303 characters of text can fit. In the staging model, when the last bit of the message string is reached and if there are still free bits in the image for further use, the last bits are replaced with random ones. The goal is that the image always undergoes the same overall change regardless of the length of the message, and that when a bit string is subsequently extracted from the image, it is always the same length when using the same image.

### 2.3 The process of extracting the message from the image

The extracting process consists of the following steps:

1. Breaking the modified pixel matrix image and focusing on the bits of component A;

2. Extracting the bits for component A of the first two pixels and decoding the length of the message in binary form;
3. Extracting all the last bits of component A of all other pixels and loading them into a single string of bits;
4. Extracting the real message from the common string of bits by means of a pattern consistent with the decoded length of the message;

Fig. 4 shows the scheme of extracting the original message from the image.

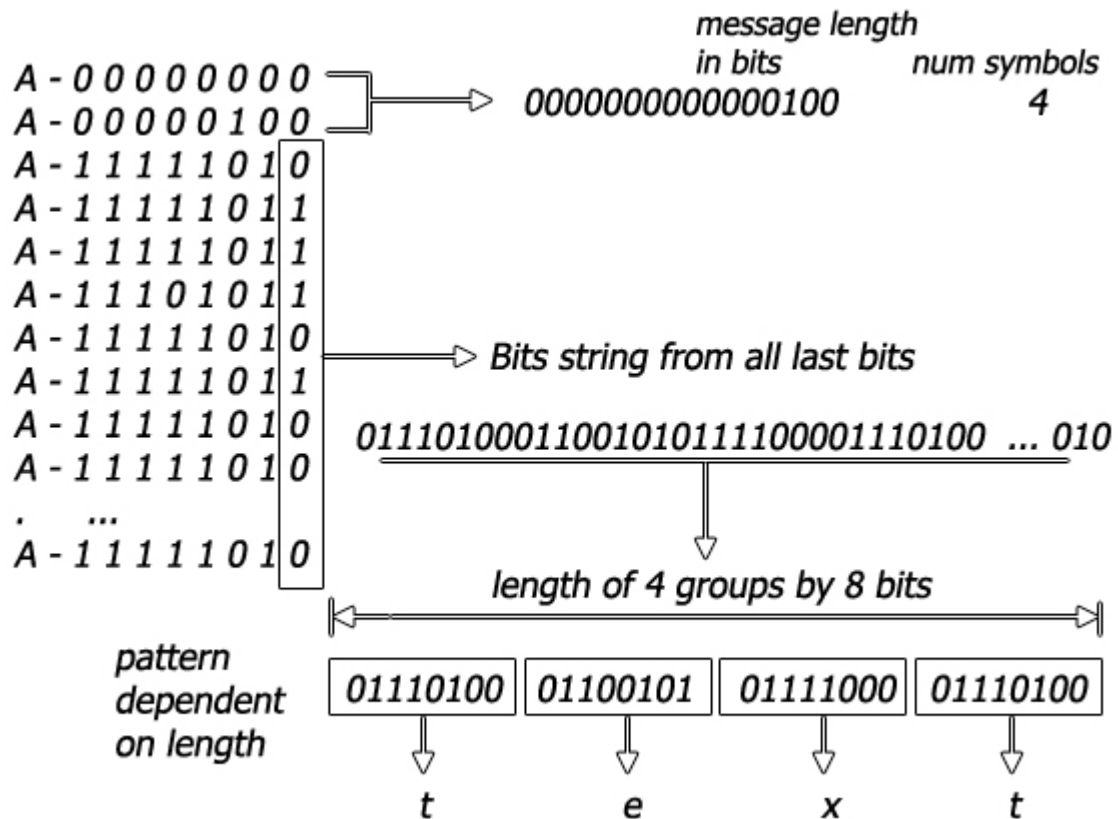


Fig. 4. Message retrieval process

A key role in the model is the attached template, without which it is impossible to retrieve the correct message. It depends on the length of the message and has a dimension of 8 bits per cell.

### 3 Experimental results

The model is implemented in the C# programming environment. The implemented application for the experimental setup is installed and tested on the following computer configuration: Processor Intel Xeon E5450 3.0 GHz, 8 GB RAM, Motherboard Gigabyte EP43-DS3LR (Chipset: Intel P43; South bridge: Intel 82801JR / ICH10R /; LPCIO: ITE IT8718), Windows 7 Ultimate Service Pack 1 64-bit (x64), .NET Framework 4.7.2. The application runs normally, taking up about 15 MB in standby mode and up to about 149 MB of total RAM space. In a single case, the application did not complete its work: when working with an image with a size of 24.7 MB and a resolution of 6048 x 4032, due to lack of system memory of the system on which it is running, and before it stops working, its consumption the system memory peaked at 3 GB.

An experiment was performed with 50 images differing in both file size and different resolution. Each of them has the same message with a length of 1024 characters. Table 1 summarizes the data of this study, showing the representatives of different groups of images.

**Table 1.** Results of the system operation with different images

image		converting of message, ms	inserting time, ms	stego image		extracted message	extracting time, ms
size	resolution			size	resolution		
858 KB	1024x768	22,1838	5,0949	116 KB	1024x768	yes	16,999
826 KB	1024x768	25,1496	4,9747	120 KB	1024x768	yes	15,4362
548 KB	1024x768	23,3738	5,385	90 KB	1024x768	yes	16,7695
112 KB	800x600	38,5494	4,8829	59 KB	800x600	yes	21,3598
81 KB	800x600	36,2246	4,8436	67 KB	800x600	yes	20,7895
1,67 MB	1920x1200	44,2086	5,0345	321 KB	1920x1200	yes	26,8954
1,47 MB	3840x2160	40,5388	5,5032	423 KB	3840x2160	yes	26,2369
6,04 MB	4503x3227	1844,7407	6,419	788 KB	4503x3227	yes	1235,895
24,7 MB	6048x4032	unsuccessful	unsuccessful		6048x4032	no	unsuccessful

## 4 Conclusion

From the results of the experiments performed with different images, it can be concluded that the system has almost 100% retrieval of embedded messages. Stego images or processed images are relatively small in size and are suitable for use in transporting messages in a communication environment. The processing times are within normal limits, but they are also highly dependent on the characteristics of the hardware host system. It can be seen that the retrieval time is less than the embedding time of the message, although the two operations are rather reversible and it is assumed that they are likely to have identical execution time in view of the fact that the use of the template is dependent on the length of the message, which is used in the retrieval, and practically everything outside it is ignored. On the whole, definitely large files are not suitable for use in such a system, but those up to 1 MB in size and a resolution of 1024 x 768, 800 x 600 and similar, are quite appropriate both as a time required to perform the necessary actions and as a capacity to accommodate a plain text message.

## References

- Grebennikov, V. V. (2019). *Steganography. History of cryptography*, LitRes, (pp. 66-69). (in Russian)
- Gribunin, V. G., Kokov, I. N. & Turintcev I.V. (2009). *Digital stenography*, Moscow: SOLON-PRESS. (in Russian)
- Fridrich, J. (2010). *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press, ISBN 978- 0521190190
- Fridrich, J. (2004). Steganography II. *In International Multimedia Conference: Proceedings of the 2004 workshop on Multimedia and security*, 2004.
- Kruus, P., Scace, C., Heyman, M., & Mundy, M. (2003). A survey of steganography techniques for image files . *Advanced Security Research Journal*. 5(1), 41-52.

- Kordov, K. & Stoyanov, B. (2017). Least Significant Bit Steganography using Hitzl-Zele Chaotic Map. *International Journal of Electronics and Telecommunications*, 63(4), 417-422.
- Rani, N. & Chaudhary, J. (2013). Text Steganography Techniques: A Review. *International Journal of Engineering Trends and Technology (IJETT)*, 4(7).
- Neil, F. & Johnson, S. J. (1998). George Mason University, "Exploring Steganography: Seeing the Unseen", *IEEE Computers*, 26-34.
- Ross, J., & Petitcolas, F.A.P. (1998). On the limits of steganography. *IEEE Journal of Selected Areas in Communications*, 16(4), 474-481. May 1998. Special Issue on Copyright & Privacy Protection. ISSN 0733-8716.
- Nain, S. & Kumar, S. (2014) Steganography and Its Various Techniques. *International Journal of Enhanced Research in Science Technology & Engineering*, 3(6), 241-245. ISSN: 2319-7463
- Stanev, S. & Szczypiorski, K. (2016). Steganography Training: A Case Study from University of Shumen in Bulgaria. *International Journal of Electronics and Telecommunications*, 62(3), 315-318.
- Stanev, S., Zhelezov, S. & Yakimov, I. (2012). An approach for parallel steganalysis based on data compression. *In Proceedings of Jubilee International Congress*, 40, 360-367.
- Stoyanov, B. P., Zhelezov, S. K. & Kordov, K. M. (2016). Least significant bit image steganography algorithm based on chaotic rotation equations. *Comptes rendus de l'Academie bulgare des Sciences*, 69(7), 845-850.
- Lokhande, U. (2014). An Effective Way of using LSB Steganography in images along with Cryptography. *International Journal of Computer Applications*, 88(12)
- Zhelezov, S. (2016). Modified Algorithm for Steganalysis. *Mathematical and Software Engineering*, 1(2), 31-36.
- Zhelezov, S. & Paraskevov, H. (2015). Possibilities for steganographic parallel processing with a cluster system. *Contemporary Engineering Sciences*, 8(20).