Modification of the RGB Color Space in MS Word Documents for Steganographic Information Protection

Vladimir Barannik[1,†], Mykhailo Babenko[2,†], Yurii Babenko[3,*,†], Kateryna Yalova[2,†] and Kseniia Yashyna[4,†]

[1]*V. N. Karazin Kharkiv National University, Svobody Square 4, Kharkiv, 61000, Ukraine*
[2]*Dniprovsky State Technical University, Dniprobydivska Street 2, Kamianske, 51918, Ukraine*
[3]*Taras Shevchenko National University of Kyiv, Volodymyrska Street 64/13, Kyiv, 01601, Ukraine*
[4]*SIRIS Academic, Av. Francesc Cambo 17, 08003, Barcelona, Spain*

Abstract

Images, audio, and video are commonly used for steganography. However, text is an ideal medium for steganography due to its ubiquity and smaller size. Text steganography involves hiding information within text. Traditional steganographic methods using multimedia files often rely on replacing the least significant bit of the original data with a bit mask of the hidden information. In this paper, we adapt this method for text steganography. We present an algorithm that uses a modified least significant bit approach to subtly alter the color parameters of text characters in a Microsoft Office Word file by modifying the low-order bits of the RGB color channels of the text characters. These changes are completely imperceptible to the human eye. We have developed software that implements this algorithm, allowing for the embedding and extraction of hidden information.

Keywords

text steganography, MS Word document, least significant bit method, RGB color space, confidentiality, hiding information, information security, coding

1. Introduction

The rapid growth of technology in recent years has led to the generation of a vast amount of information, which is typically transmitted over unsecured network channels. The Internet, in particular, has gained widespread popularity for the exchange of digital data, utilized by both private individuals and governmental organizations. Despite the numerous advantages of such information exchange, a significant drawback remains the confidentiality and security of the

ICyberPhyS-2024: 1st International Workshop on Intelligent &CyberPhysical Systems, June 28, 2024, Khmelnytskyi, Ukraine
*Corresponding author.
†These authors contributed equally.
v.v.barannik@karazin.ua (V. Barannik); mvbab130973@gmail.com (M. Babenko); babenkomahalych@gmail.com (Y. Babenko); yalovakateryna@gmail.com (K. Yalova); yashinaksenia85@gmail.com (K. Yashyna)
0000-0002-2848-4524 (V. Barannik); 0000-0003-1013-9383 (M. Babenko); 0000-0002-8115-3329 (Y. Babenko); 0000-0002-2687-5863 (K. Yalova); 0000-0002-8817-8609 (K. Yashyna)

data. The availability of various easily accessible tools capable of compromising the confidentiality, integrity, and security of transmitted data has enabled the emergence of diverse malicious threats. The most common solution to this issue is cryptography, where data is encrypted into ciphertext using an encryption key. On the receiving end, the ciphertext is decrypted back into plaintext using a decryption key. While encryption ensures that the original data is not visible, the presence of ciphertext in its encrypted form is noticeable, raising suspicions and prompting further investigation. Consequently, a new area of research, steganography, has gained attention, allowing for the concealment of data in a manner that is invisible to the human eye [1, 2].

Methods of hiding information have existed for a long time [3, 4], but their significance has recently increased [5, 6]. The primary reason for this is the growing volume of data traffic over the Internet and social media. Although the goals of cryptography and steganography are similar, there is a fundamental difference between them. Cryptography makes data unbreakable and unreadable, but the ciphertext is visible to the human eye. In contrast, the distinctive feature of the steganographic approach is that it does not explicitly reveal the existence of protected information. This characteristic allows for solving important information security tasks within the traditionally existing information flows or environments. The term "steganography" refers to hidden messages that completely eliminate the possibility of third-party detection. Thus, an uninformed person fundamentally cannot decipher the message because they are unaware of its existence. In other words, while cryptography conceals the content of a message, steganography conceals the very existence of the message. Cryptography remains the most popular method of information hiding, but steganography has been gaining increasing popularity recently. The most promising method appears to be the combination of cryptography and steganography, where already encrypted information is hidden.

Computer steganography is based on two main principles:

1. Files that do not require absolute precision (such as image, video, or audio files) can be modified to some extent without losing functionality.
2. The lack of specialized tools or the human sensory system's inability to reliably detect minor changes in such files.

The basic approaches to implementing computer steganography methods involve identifying insignificant fragments within a medium and replacing the existing information in those fragments with the information intended to be protected. Since computer steganography deals with environments supported by computational means, the entire information environment can ultimately be represented in digital form. Thus, fragments that are insignificant to the frame of the information environment are replaced (or mixed) with fragments of the hidden information according to a specific algorithm [6]. This means that a family photo can contain commercial information, and a file with a favorite melody can hide a secret message.

In the context of implementing software and hardware protection in information systems, two main approaches are distinguished:

1. Embedded Protection: Protection mechanisms are distributed across other components of the system or implemented as separate elements of the information system.

2. Additional Protection: Protection mechanisms serve as supplements to the primary hardware and software components of the information system.

The second approach is more flexible, allowing protection mechanisms to be added or removed as needed. However, its implementation may encounter issues with ensuring compatibility between different protection methods and the software-hardware complex of the information system. Embedded protection is considered more reliable and optimal but is also more rigid, making it difficult to modify. Due to these characteristics, real-world systems often combine both approaches to leverage their respective advantages.

2. Related Works

Existing algorithms for embedding secret information can be categorized into several groups:

1. Algorithms that operate directly on the digital signal. This group includes the LSB (Least Significant Bit) method [6, 7], which will be used further in this study.
2. Embedding secret information. In this case, the hidden image, sound, or text is superimposed on the original content. This method is often used for embedding digital watermarks (DWM).
3. Utilizing file format capabilities. This involves embedding information into metadata or other reserved fields of a file that are not used.

Steganographic methods that use multimedia files as containers often rely on replacing the least significant bit of the original data with a bit mask of the hidden information. This approach alters the bit distribution pattern. To avoid significantly disrupting the structure of the original data, either pseudorandom filling is used, or the original data's density distribution function is analyzed, and the container is filled according to the identified pattern.

In the study [8], a method for embedding data into image fragments with a high level of decorrelation is proposed, which corresponds to a high level of fractality. Embedding hidden data only in those fragments where the data is least structured allows for more reliable embedding.

Recently, increasing attention has been given to deep learning technology, which has become a powerful tool in various applications, including image steganography. In [9], the authors explore different deep learning methods available in the field of image steganography. They categorize deep learning methods for image steganography into three groups: traditional methods, convolutional neural network-based methods, and generative adversarial network-based methods.

Images, audio, and video are among the most popular carriers for steganography. Hiding information in video containers is particularly relevant because these containers have an immense amount of redundant data, allowing for the effective concealment of large volumes of information. Numerous studies have been dedicated to this topic by both local researchers [10–12] and international researchers [13–15].

Image, audio, and video steganography have received significant attention due to their visual and auditory nature, making them attractive for hiding information. However, text steganography, despite being less popular, is gaining momentum. In the era of ubiquitous electronic document circulation, text becomes a key element of data exchange. This growth is

driven by the need to ensure confidentiality and protect information in the digital environment, making text steganography an important and promising field. Therefore, hiding information in text documents is becoming increasingly relevant, although there are not as many publications on this topic as it deserves.

Images, audio, and video are some of the most popular carriers for steganography. However, text is ideal for steganography due to its ubiquity and smaller size compared to these media. Text steganography involves hiding information within text. Text has been one of the oldest means of hiding data, with letters, books, and telegrams being used to conceal secret messages before the advent of digital steganography. Additionally, text documents are among the most common digital media today, found in the form of newspapers, books, web pages, source codes, contracts, advertisements, and more. Therefore, the development of text steganography and steganalysis is very important. On one hand, methods for hiding data in text documents pose a significant threat to cybersecurity and can be a new communication tool for terrorists and other criminals. On the other hand, these methods can have legitimate applications in document tracking, copyright protection, authentication, and the investigation of counterfeits and forgeries.

Since the plan is to embed hidden data into a text file, let's consider the most common methods of text steganography for this purpose:

1. Case Modification: Assume that uppercase letters represent bits with a value of one, and lowercase letters represent bits with a value of zero. Using this approach, a text of length N can hide a message of N/5 characters, which is quite inconvenient.
2. Whitespace Modification: Assume that a single space represents a bit "0", and two spaces represent a bit "1". The program takes any text as a container and embeds the message by replacing its bits with the corresponding number of spaces. The encoding method plays an important role here. The character code should be of optimal length, and double spaces should occur as rarely as possible.
3. Line-Shift Coding: This technique changes the distance between lines in electronic text to encode information.
4. Word-Shift Coding: This method changes the distance between words in a text. It involves identifying the maximum and minimum distances between words and assigning them values of 1 and 0, respectively, while other distances are adjusted accordingly. A specific case of this method is the previously mentioned whitespace modification method.
5. Feature Coding: This involves making specific changes to the fonts of individual letters, such as varying the length of the descender in the letter "p".
6. Using Characters from Different Alphabets with Identical Appearance: For example, the letter "i" has the same appearance in both the Ukrainian and English alphabets. Assume that the Ukrainian "i" represents bit "0" and the English "i" represents bit "1". Given that many letters have this property ("a", "e", "o", "p", etc.), a significant amount of information can be hidden.
7. Using the Order of End-of-Line Markers (CR/LF): Typically, text display tools are indifferent to the order of the carriage return (CR) and line feed (LF) symbols that end a line of text. Assume that the traditional combination CR/LF represents "0", while the inverted LF/CR represents "1".

In [16-18], a new method of text steganography called AITSteg is proposed, which provides end-to-end protection during the transmission of text messages via SMS or social networks between end users. The authors claim that the proposed method can hide a large volume of secret information within a short message in such a way that the embedding trace is completely invisible to observers. Additionally, the method implements a combination of mathematical encoding and symmetric key algorithms, which generates different secret bits even for the same confidential information at different times, making it completely secure against attacks, including "man-in-the-middle" attacks, message disclosure, and manipulation by readers.

Microsoft Word is one of the most popular word processors, and several methods exist for embedding data into documents created with it. In [19], a new type of data hiding method in Microsoft Word documents, called property coding, is presented. This method uses the properties of various document objects (such as characters, paragraphs, and sentences) to embed data. The authors present four different property coding techniques that are resilient to saving actions, impose very low overhead on document size (about 1%), can embed up to 8 bits per character, and, of course, remain unnoticed by readers. Property coding belongs to the formatting methods of text steganography.

In [20], a new method of text steganography in Microsoft Word documents is proposed. The main idea is that setting any foreground color for invisible characters, such as spaces or carriage returns, does not display during document viewing.

The method presented in [21] considers font types and works by replacing the font with a similar one. The secret message is encoded and embedded using similar fonts in the capital letters of the container text. The proposed text steganography method can work in various accompanying documents with different font types. The container size increases by approximately 0.766% from the original size. The performance of this method is very high, and the secret message remains invisible to an adversary.

3. Proposed methodology

3.1. LSB methodology

The methods discussed above are easily embedded into any text, regardless of its content, purpose, or language. However, these methods are unfortunately susceptible to being broken, allowing the secret information to become accessible to third parties. Another major drawback is that these methods cannot transmit a large amount of hidden information. Therefore, in this work, we apply the LSB method to text steganography, which is typically used with graphical images and digital signals [5–9], but in a modified form known as LSB matching. This modified method is considered more resistant to many steganalysis techniques. The key features of this method include, first, the use of a sequence of random binary numbers during encoding and, second, the use of addition and subtraction during the replacement of the last bit, conducted in "regular" binary arithmetic (with carry-overs) [22, 23].

The LSB (Least Significant Bit) replacement method is a fundamental technique in the field of steganography. It involves replacing the least significant bits in the main container, such as images, audio, or video, with the hidden bits of the message. The goal is for the visual or

auditory difference between the original and modified containers to be imperceptible to the human observer [1, 2, 6, 7].

Let's describe the classic LSB replacement method. For example, consider an 8-bit grayscale image, where each pixel's color is stored in eight bits. In this case, '00h' (00000000b) represents black, and 'FFh' (11111111b) represents white, giving a total of 256 possible shades of gray. Suppose the message to be hidden consists of one byte, such as '01101011b'. If two least significant bits in the pixel descriptions are used, then four pixels are needed to hide the message.

For simplicity, let's assume the pixels are black. Then, the pixels containing the hidden message will have the following bit sequences: '00000001', '00000010', '00000010', and '00000011'. This manipulation results in very small color changes: the first pixel changes by 1/255, the second and third by 2/255, and the fourth by 3/255. These changes are practically imperceptible to the human eye, especially when viewed on low-quality display devices.

The situation is even better when using a color image, where each pixel's color spectrum is stored in twenty-four bits (three bytes—one byte per color channel), resulting in a total of 16,777,216 (2^24) possible color shades (Figure 1).

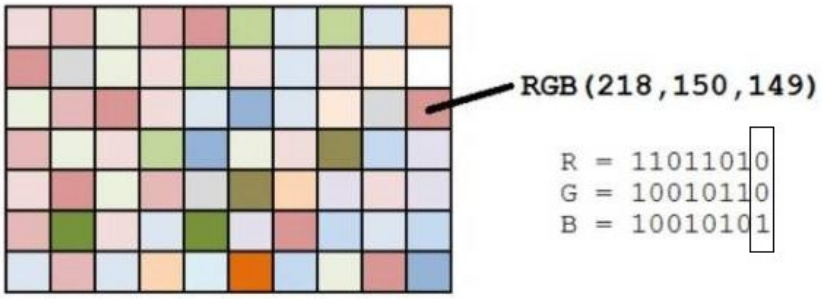

Figure 1: The least significant bits of a color image's pixel

To hide one bit of information, the least significant bit of one of the color channels of a pixel needs to be replaced. Therefore, one pixel can contain three bits of hidden information. To hide one byte, three pixels are needed, which can store nine bits – the original byte plus one "extra" bit that remains unmodified, which is not very efficient. For example, the process of hiding the character "S" (ASCII code – 83, in binary 01010011) is illustrated in Figure 2.

When using the two least significant bits from each color channel of a pixel, hiding one byte of information requires using all three color channels of one pixel and one color channel of another pixel, which is also not very convenient (Figure 3).

To hide one byte of information, the optimal scheme would be as follows: replace three least significant bits in the red channel, three in the green channel, and two in the blue channel (Figure 4). This way, one byte of information can be hidden in one pixel. However, such significant changes in the color channels can noticeably distort the container image, making it detectable.

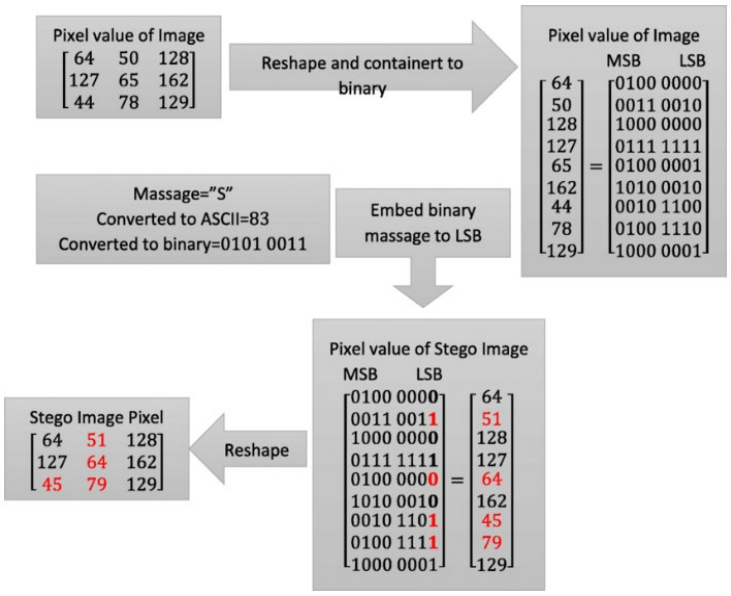

Figure 2: Hiding character "S" using the LBS method

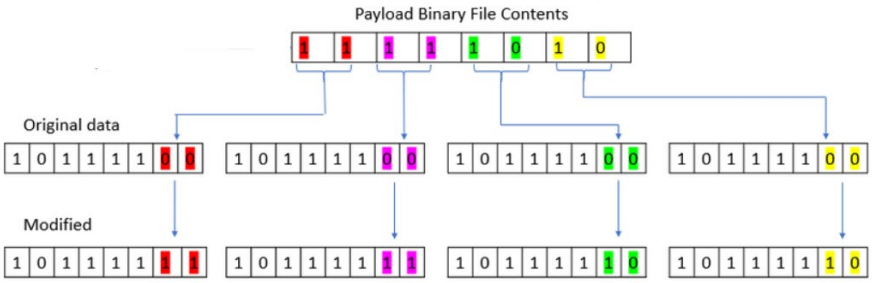

Figure 3: Using two of the least significant bits

The LSB replacement method is a widely-used and straightforward technique in spatial steganography. Since the least significant bit of an image carries the least amount of information, it is an ideal medium for hiding secret data within image data. Essentially, the least significant bit acts as noise within the environment, and humans typically cannot detect changes in this bit. In grayscale images (where each pixel is encoded with one byte), this method can potentially conceal up to 1/8 of the total container volume. For example, an image with a resolution of 512×512 can hold approximately 32 KB of hidden information. By altering two least significant bits, which is also barely noticeable, this capacity can be doubled.

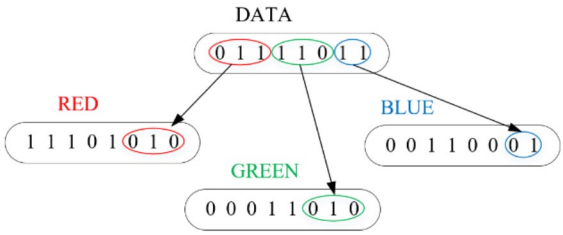

Figure 4: Scheme for using the LSB method to hide one byte of information in one pixel

The popularity of this method stems from its simplicity and its ability to hide significant amounts of data in relatively small files, creating a covert communication channel with a bandwidth ranging from 12.5% to 30%. However, it is important to recognize that the LSB method is vulnerable to various passive and active intrusion techniques. Its primary disadvantage is its sensitivity to even minor distortions in the container. To mitigate this issue, error-resilient coding is often employed. Additionally, LSB methods are unstable in the presence of noise in the transmission channel.

The algorithm proposed in this study is based on modifying the color space of the container, specifically the color parameters of text characters in the RGB model. The fundamental element of the container, whose properties are altered during information embedding, is the text character of an electronic document. This algorithm utilizes LSB matching, which enhances the security of the steganographic embedding. The container for the secret data will be an Office Open XML format file, exemplified by a Microsoft Office Word document with the docx extension. Why choose the docx format instead of doc or txt? There are several compelling reasons. First, a docx file, unlike a doc file, is a zip archive containing XML documents that can be unpacked to retrieve all necessary information: text, images, tables, etc. This makes it relatively easy to embed and extract hidden data. Second, docx is the most popular and widely-used format, and its frequent use will not arouse suspicion regarding embedded data, which is a significant advantage. Third, docx files are much smaller in size compared to their doc counterparts, particularly in files containing numerous images or charts. Using the C# programming language, software has been developed to hide secret information in such a way that no one else can detect its presence. Additionally, the software ensures the capability to extract the hidden message from the container where it has been embedded.

3.2. Proposed LSB methodology modification

As mentioned earlier, the human eye cannot distinguish slight shades of the same color. This is also applicable to text steganography, as demonstrated in [24, 25]. This can be effectively utilized in developing an algorithm for embedding secret data into a text container in docx format. The essence of this algorithm is as follows. We have a message that needs to be hidden in a document with the docx extension. The document itself must already contain textual information. The volume of this information will determine the amount of data that can be embedded. The more text the document contains, the more data can be hidden within it. The data is embedded into the RGB color channels of each text character in the file. To do this, we first need to parse the Word document to extract all necessary data: the text and the color information of each character in RGB format. The extracted color components are then converted to binary, and the least significant bits of the color components are replaced with the bits of our message.

Let's explain this in more detail with an example. Suppose 1 byte of our message looks like this: 10101010. We split this message into three parts (corresponding to the RGB channels – 101 010 10) and replace the last bits. Using black as the base color (binary representation 00000000), the result will be as follows:

- R: 00000101 = 5
- G: 00000010 = 2
- B: 00000010 = 2

The final result, after converting back to RGB format, will be the color corresponding to rgb(5,2,2). This is a shade of black. Since the human eye's ability to differentiate between slight shades of the same color is limited, such a transformation will remain unnoticed (Figure 5).

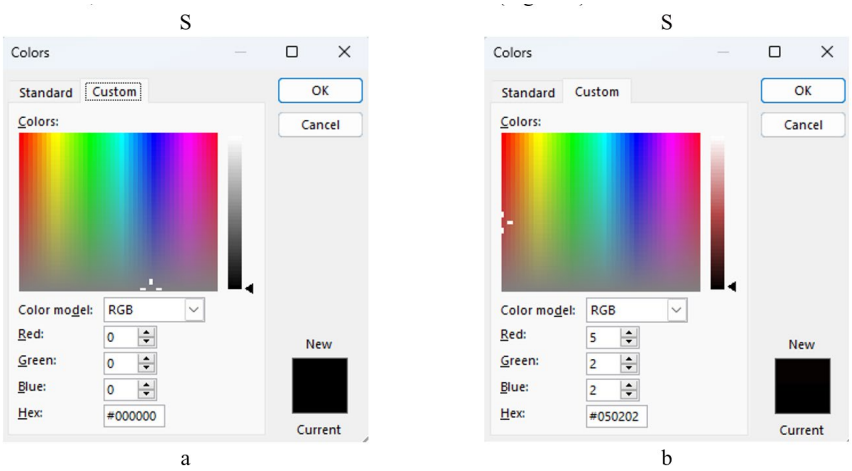

Figure 5: The letter «S» in black (a) and with a shade of black rgb(5,2,2) (b)

This operation will not introduce distortions in the color noticeable to the human eye. Instead, it will allow us to embed exactly 1 byte of our message into the color of each character in the input file. Thus, the maximum number of bytes (or characters) that can be hidden will be equal to the number of characters in the docx document, including spaces, tabs, carriage return symbols, and line feed symbols.

Next, we need to determine the maximum number of bits that can be embedded in one RGB channel without causing significant changes to the character's color. How many bits can be embedded across all three channels so that the resulting color remains unchanged to the point of being imperceptible upon simple viewing? To address this, we will conduct an experiment, gradually increasing the number of modified least significant bits. The results are presented in Table 1.

Table 1
Results of embedding varying numbers of bits into one RGB channel

Symbols in different font colors	Binary color description
S	R – 00000000 G – 00000000 B – 00000000
S	R – 00000001 G – 00000001 B – 00000001
S	R – 00000011 G – 00000011

	B – 00000011
S	R – 00000111 G – 00000111 B – 00000111
S	R – 00001111 G – 00001111 B – 00001111
S	R – 00011111 G – 00011111 B – 00011111
S	R – 00111111 G – 00111111 B – 00111111
S	R – 01111111 G – 01111111 B – 01111111
	R – 11111111 G – 11111111 B – 11111111

In the last row, the letter "S" became invisible because it was white on a white background. We can conclude that replacing five or even six least significant bits does not result in visible changes to the black color of the character. However, for reliable concealment, it is better to limit the replacement to four or even three bits. Thus, similar to the case with raster images, the optimal scheme for hiding one byte of information would be to use three least significant bits in the red channel, three in the green channel, and two in the blue channel. Embedding more bits into the color channels makes the color change noticeable, which is undesirable. The conclusions are summarized in Table 2.

Table 2
Conclusions on embedding different number of bits in one RGB channel for hiding one byte of information

Number of bits embedded in one RGB channel	Data volume that can be embedded in the container	Appearance in the container
1 bit per color channel	N/8, where N is the number of container characters	No visible changes, but the volume of embedded information is significantly small

2 bits in the R channel, 2 bits in the G channel, 2 bits in the B channel, 2 bits in the R channel of the next character	N/4, where N is the number of container characters	No visible changes, the volume of embedded information is twice as large compared to the previous case
3 bits in the R channel, 3 bits in the G channel, 2 bits in the B channel (1 byte per color)	N, where N is the number of container characters	No visible changes, the volume of embedded information is satisfactory
4 bits per color channel	3/2*N, where N is the number of container characters	The volume of embedded information is larger than in the previous case, but the color changes are more noticeable

4. Results

Thus, information of any type can be embedded as long as it is represented in binary form. However, it is crucial to monitor the length of the message that is to be embedded. If the message length exceeds the capacity of the container, the embedding process will not be feasible. In such situations, a decision must be made regarding whether to increase the size of the container or to reduce the length of the message to fit within the available space.

Similarly, data extraction from the container follows the same principles. To retrieve the embedded message, one must parse the Word document to obtain the RGB color values of the text characters. The last bits of each RGB channel must be read to reconstruct the hidden data. Each combination of these bits will constitute one byte (or character) of the concealed message. By repeating this process for all modified character's colors in the text, the entirety of the hidden message will be revealed.

Future research will be directed towards enhancing the protection of the embedded information. This will involve encrypting the data using cryptographic methods to ensure it is secure from unauthorized access. After encryption, the data will be compressed using archiving tools such as ZIP or RAR to reduce its size and add an additional layer of protection. Additionally, a hash of the hidden message will be embedded into the container to ensure the integrity of the information. The hash will be embedded using the same algorithm but will be stored separately from the main message. Only after these steps will the data be embedded into the text container using the aforementioned method. This comprehensive approach aims to significantly enhance the security level of the embedded information, making it more resistant to detection and unauthorized access.

5. Conclusions

Thus, steganographic methods for hiding information in Microsoft Office Word documents have been considered. Based on one of these methods (LSB matching, which is typically not used in text steganography) and using the RGB color model, an algorithm for embedding hidden data in a Microsoft Word document with a docx extension has been developed. This algorithm is based on the replacement of the least significant bit in the color space of characters in electronic documents. Additionally, software that fully implements this algorithm has been created using the C# programming language.

References

[1] A. Podder, P. Roy, S. Roy. Steganography techniques – an overview. International Journal of Scientific Research in Computer Science, engineering and information technology, 8 (2022) 300–304. doi: 10.1007/s00521-022-07366-3.

[2] Y. Yong. Development and future of information hiding in image transformation domain: A literature review, in: Proceedings of the 4th International Conference on Image Processing and Machine Vision, IPMV '22, ACM Press, New York, NY, 2022, pp. 72–77. doi: 10.1145/3529446.3529458.

[3] J. T. Brassil, S. L., N. F. Maxemchuk, L. O'Gorman. Electronic Marking and Identification Techniques to Discourage Document Copying. IEEE Journal on Selected Areas in Communications, 13(1995) 1495–1504. doi:10.1109/49.464718.

[4] W. Bender, D. Gruhl, N. Morimoto, A. Lu. Techniques for Data Hiding. IBM Systems Journal, 35 (1996) 313–336. doi:10.1147/sj.353.0313.

[5] S. Dhawan, R. Gupta. Analysis of various data security techniques of steganography: A survey. Information Security Journal: A global Perspective, 30 (2021) 63–87.

[6] S. Ghoul, R. Sulaiman, Z. Shukur. A review on security techniques in image steganography. International Journal of Advanced Computer Science and Applications, 14 (2023) 361–385. doi: 10.14569/IJACSA.2023.0140640.

[7] O. Awe, O.A. Olukiran, O. Aienko, F. A. Taofeek-Ibrahim. A survey of image steganography using Least Significant Bit (LSB). Electronics and computers science, 20 (2020) 1–8.

[8] M. A. Aslam, M. Rashid, F. Azam, M. Abbas. Image steganography using Least Significant Bit (LSB) – A Systematic Literature Review, in: Proceedings of 2nd International Conference on Computing and Information Technology, ICCIT'22, UT, Tabuk, Saudi Arabia, 2022, pp. 32–38.

[9] N. Subramanian, O. Elharrouss, S. Al-Maadeed, A. Bouridane. Image Steganography: A Review of the Recent Advances. IEEE Access, 9 (2021) 23409–23423. doi: 10.1109/ACCESS.2021.3053998.

[10] V. Barannik, M. Babenko. A Method of Scrambling for the System of Cryptocompression of Codograms Service Components, in: M. Klymash, A. Luntovskyy, M. Beshley, I. Melnyk, A. Schill, A. (Eds.), Lecture Notes in Electrical Engineering, volume 965. Springer, Switzerland, Cham, 2023, pp. 444–459. doi: 10.1007/978-3-031-24963-1_26.

[11] V. Barannik, M. Babenko, A. Berchanov, V. Barannik, R. Onyshchenko, L. Kolodiichuk, Method of Mini Segments Encoding in Difference Space Using Haar Wavelet, in: Proceedings of the 5th International Conference on Advanced Information and Communication Technologies, AICT'23, Lviv, Ukraine, 2023, pp. 1–4. doi: 10.1109/AICT61584.2023.10452674.

[12] T. Belikova, M. Babenko, A. Vlasov, P. Hurzhii, N. Korolyova, O. Voitsekhivska, Methodology for Assessing the Security of Encoded Video Information Resources of Video Conferencing in Public Administration, in: Proceedings of the 4th International Conference on Advanced Trends in Information Theory, ATIT'22, Kyiv, Ukraine, 2022, pp. 57–60. doi: 10.1109/ATIT58178.2022.10024230.

[13] J. M. Jenifer, S. R. Ratna, J. B. S Loret, D. M. Gethsy. A Survey on Different Video Steganography Techniques, in: Proceedings of the 2nd International Conference on Trends in Electronics and Informatics, ICOEI'18, Tirunelveli, India, 2018, pp. 627–632. doi: 10.1109/ICOEI.2018.8553847.

[14] G. R. Manjula, R. B. Sushma. Video Steganography: A Survey of Techniques and Methodologies, 2021. URL: http://dx.doi.org/10.2139/ssrn.3851241.

[15] S. Kamil, M. Ayob, S.N.H.S. Abdullah, Z. Ahmad. Challenges in multi-layer data security for video steganography revisited, Asia-Pacific Journal of Information Technology and Multimedia, 7 (2018) 53–62. doi: 10.17576/apjitm-2018-0702(02)-05.

[16] M. T. Ahvanooey, Q. Li, J. Hou, H.D. Mazraeh, J. Zhang. AITSteg: An Innovative Text Steganography Technique for Hidden Transmission of Text Message via Social Media. IEEE Access, 6 (2018) 65981–65995. doi: 10.1109/ACCESS.2018.2866063.

[17] M. T. Ahvanooey, Q. Li, J. Hou, A.R. Rajput, Y. Chen. Modern Text Hiding, Text Steganalysis, and Applications: A Comparative Analysis. Entropy, 21 (2019) 350 – 381. doi: 10.3390/e21040355.

[18] M. T. Ahvanooey, Q. Li, H. J. Shim, Y. Huang, A Comparative analysis of information hiding techniques for copyright protection of text documents. Security and Communication Networks, 2018 (2018) 1–22. doi: 10.1155/2018/5325040.

[19] M. A. Majeed, R. Sulaiman, Z. Shukur, K. Hasan. A review on text steganography techniques. Mathematics, 9 (2021) 1–28. doi: 10.3390/math9212829.

[20] S. Mahato, D. A. Khan, D. K. Yadav. A modified approach to data hiding in Microsoft Word documents by change-tracking technique. Computer and Information Sciences, 32 (2020) 216-224. doi: 10.1016/j.jksuci.2017.08.004.

[21] K. F. Rafat, M. J. Hussain. Secure text steganography for Microsoft Word Document. International Journal of electrical and information engineering, 11 (2017) 736–741.

[22] Y. Hu, X. Li, J. Ma. A novel LSB Matching algorithm based on information pre-processing. Mathematics, 10 (2022) 1–16. doi: 10.3390/math10010008.

[23] L. Nechvoloda, K. Krykunenko, K. Paramonova. Application of a mathematical model for the generation of separate elements of a steganographic system in a higher education institution. Computer Systems and Information Technologies, 2 (2023) 48–54. doi: 10.31891/csit-2023-2-6.

[24] R. Thabit, N. I. Udzir, S. M. Yasin, A. Asmawi, A. A.-A. Gutub. CSNTSteg: color spacing normalization text steganography model to improve capacity and invisibility of hidden data. IEEE Access, 10 (2022) 65439–65458. doi: 10.1109/ACCESS.2022.3182712.

[25] B. Osman, N. I. Yahya, K. M. Zaini, A. Abdullah. Text steganography using the second quotient remainder theorem and dark colour schemes. Journal of Computational Innovation and Analytics, 2(1) (2023) 21–40. doi: 10.32890/jcia2023.2.1.2.