Enhancing Data Protection in Digital Communication: A Novel Method of Combining Steganography and Encryption

**Khaled H. Abuhmaidan[1*], Marwan A. Al-Share[2], Abdallah M. Abualkishik[3],
and Ahmad Kayed[4]**

Faculty of Computing and Information Technology, Sohar University, Sohar, Oman
[1] [e-mail: khmaidan@su.edu.om, kabuhumaidan@yahoo.com]
[2] [e-mail: MShare@su.edu.om]
[3] [e-mail: AAbualkishik@su.edu.om]
[4] [e-mail: akayed@su.edu.om, drkayed@ymail.com]
*Corresponding author : Khaled H. Abuhmaidan

Abstract

In today's highly digitized landscape, securing digital communication is paramount due to threats like hacking, unauthorized data access, and network policy violations. The response to these challenges has been the development of cryptography applications, though many existing techniques face issues of complexity, efficiency, and limitations. Notably, sophisticated intruders can easily discern encrypted data during transmission, casting doubt on overall security. In contrast to encryption, steganography offers the unique advantage of concealing data without easy detection, although it, too, grapples with challenges. The primary hurdles in image steganography revolve around the quality and payload capacity of the cover image, which are persistently compromised.

This article introduces a pioneering approach that integrates image steganography and encryption, presenting the BitPatternStego method. This novel technique addresses prevalent issues in image steganography, such as stego-image quality and payload, by concealing secret data within image pixels with identical bit patterns as their characters. Consequently, concerns regarding the quality and payload capacity of steganographic images become obsolete. Moreover, the BitPatternStego method boasts the capability to generate millions of keys for the same secret message, offering a robust and versatile solution to the evolving landscape of digital security challenges.

1620

Khaled H. Abuhmaidan et al.: Enhancing Data Protection in Digital Communication:
A Novel Method of Combining Steganography and Encryption

1. Introduction

Ensuring information security has emerged as a paramount concern in the realm of information technology and communication, especially with the advent of the Internet. The expansion of digital communication networks has heightened the public's expectations for privacy and security in data transmission. Various strategies have been implemented to protect data integrity and facilitate secure data transmission processes, mitigating the risks of unauthorized access or data leakage due to irresponsible actions

For centuries, techniques in cryptography and steganography have been employed to safeguard sensitive data. Cryptography involves transforming confidential information into ambiguous data through mathematical algorithms and keys. While still identifiable, this obscured data can attract the attention of unintended third parties, leading to potentially dire consequences. The primary objective of cryptography is to ensure that even if intercepted, the encrypted message remains confidential and incomprehensible to unauthorized individuals.

In contrast, steganography focuses on the concealment of sensitive data to evade detection [1]. It revolves around hiding the communication itself within another medium, such as an image, audio file, or video, in a manner imperceptible to human senses [2]. The most popular carrier format on the Internet is images because of their high frequency on the Internet. There are many existing steganography techniques for hiding secret messages in images with their respective pros and cons [3]. Obviously, techniques with modest strengths would increase data vulnerability to discovery, extraction, and attack.

Image steganography has many applications. In addition to sending sensitive information embedded in an image through channels such as email and social media, it embeds a digital watermark in the image as well [4]. These watermarks can be used for copyright protection or image authentication. Moreover, image steganography can be employed to store personal or sensitive information, such as passwords or financial data, within an image. Therefore, image steganography is an effective technique for preventing unauthorized parties from accessing sensitive information [3].

The accuracy in identifying the pixels essential for concealing the secret message is a crucial aspect of steganography embedding methods. Furthermore, it is insufficient merely to identify the current embedding location; it is equally vital to establish a sequence of locations. Thus, there is a need for an effective and adaptable technique to define the entire path of embedded locations.

Moreover, an effective steganography technique should exhibit high visual quality with minimal distortion and provide a sufficient payload [5]. While certain steganography approaches prioritize improved stego-image quality, they often come with inherent payload constraints. The trade-off between hiding payload and the quality of the stego-image is obvious, as reducing payload enhances un-detectability. Striking a balance among various steganography requirements poses a challenge for steganographers [6][7].

This study introduces a novel method known as BitPatternStego, aimed at concealing desired text bits within the blue channel of an RGB image. The approach involves utilizing an identical bit pattern shared between the text and the blue channel, ensuring a seamless integration without altering the cover image. By maintaining this consistency, the method successfully avoids any distortion, enhancing security through undetectability. Moreover, it concurrently augments the image's data storage capacity (payload), enabling it to accommodate a substantial volume of data, potentially unlimited.

2. Related work and Background

The development of image steganography has gathered extensive research attention as a result of its superiority to some limitations of cryptographic methods whose computational complexity is enormous as well as their ability to attract the attention of attackers [8]. Consequently, image steganography is used in many practical applications, including secure mobile computing [9], secure online voting systems [10], and secure communication between two parties [11].

Implementing image steganography methods is generally divided into two domains: the frequency domain and the spatial domain [12]. In a frequency-domain method, it manipulates the image's orthogonal transformation, which involves magnitude and phase, which represent frequency and space respectively. This method consists of an algorithm plus the transformed image. Although it is more flexible against image processing attacks, the frequency domain has a limited payload [13] and is not suitable for many real-time applications [14]. Conversely, spatial domain methods [15][16] act directly on the image's pixels, replacing the least significant bit (LSB) with the embedded secret message. In this regard, spatial domain algorithms are simpler, faster, more powerful, and more resistant to attacks [17]. In addition, despite having a good payload capacity, it lacks flexibility against statistical attacks and causes slight changes to the cover media [18][13].

The LSB method [12] is one of the spatial domain techniques based on the RGB (Red Green Blue) color model. The pixel's LSB is replaced with a secret message bit based on a secret key. It begins with one bit per pixel (BPP) and proceeds to two or three LSB or even more bits per pixel. The more substituted LSBs there are, the more obvious the image distortion [19].

Numerous efforts have been made to mitigate image distortion, as evidenced in works like [19][6]. Tsai et al. [20] researched into the concept of pixel relationships, a crucial consideration when concealing secret data within an image's pixels. Failing to account for this can lead to diminished visual quality and payload, impacting both smooth and edgy areas of the input image. To enhance payload and visual quality, [20] proposes a strategy of concentrating data hiding in edgy pixels while reducing it in smooth areas. This idea has inspired several researchers, resulting in various improved versions of Tsai's approach, as seen in [21][22] and [23]. Additionally, efforts have been directed towards enhancing visual quality. For instance, Chang et al. [24] devised a strategy to specify pixel adjustments for embedding data, further refined by Tsai et al. through pixel value differencing. Despite improvements in visual quality and payload capacity in both approaches, suitable for a wide range of applications, the original cover image still struggles with distortion and payload challenges.

In [25], the Sequential Color Cycle (SCC) technique serves as an illustration of a steganography method overwhelmed by distortion and payload issues. This approach involves cyclic modifications to the color channels (RGB) of each pixel, following a specific pattern. This alteration facilitates the concealment of one to four bits within the least significant bits (LSBs). For instance, in the case of the one-LSB pattern, secret message bits are distributed among the LSBs of the red, green, and blue channels. While SCC offers enhanced security and an increased payload compared to the original LSB method, it comes at the cost of declined image quality. This degradation may result in challenges related to uncovering hidden information and detecting the employed cycling pattern. Furthermore, SCC still exhibits a limited payload capacity, particularly when dealing with large hiding text.

In [26], there's another instance of limitations in cover image payload and visual quality. Despite the authors' efforts to enhance security through a simple randomization technique for embedding and extracting data from the stego-image, a significant portion of the cover image payload is wasted. The algorithm outlined in the paper, exemplified by the pattern (10),

involves using one pixel to conceal secret bits and skipping the next pixel, resulting in a 50% payload waste. Similarly, the key pattern (1011) dictates that three out of four pixels will be utilized for concealing hidden text, leading to a 25% payload wastage, along with a reduction in stego-image quality. Despite the authors' attempts to minimize wasted payload capacity, it unintentionally affects the quality of the stego-image.

As a result, numerous steganography techniques strive to enhance security levels, while others focus on improving the quality of the stego-image [27][28]. However, these approaches often come with inherent capacity limitations. A proficient steganography technique should aim to strike a harmonious balance among various steganography requirements [6][7].

3. Preliminaries

3.1 Bits for Digital Text and Colors.

Bits represent both text data and colors (in digital forms) through encoding. As known, encoding is the process of converting information into a format that can be stored or transmitted digitally. In the case of text data, the ASCII code is a common encoding system that uses 8 bits to encode each character or symbol. Therefore, $2^8 = 256$ different combinations of 0s and 1s that can be used to represent 256 different characters. For example, the letter "A" is represented by the binary code 01000001, see ASCII code table.

On the other side, the colors (in digital images) in each pixel are encoded by a combination of bits that indicate the intensity of the red, green, and blue components of the color. This is known as RGB (red, green, blue) encoding. Each component is usually represented by 8 bits, which means that there are 256 possible values for each component, ranging from 0 to 255 (the 0 value represents no color, while 255 represents the maximum amount of color). Therefore, each pixel can be represented by a combination of 24 bits (8 bits for red, 8 bits for green, and 8 bits for blue) (see **Fig. 1**), allowing for a total of 16,777,216 (2^{24}) different colors.

One Pixel (24 bit)		
8-bit Red	8-bit Green	8-bit Blue
0-255	0-255	0-255

Fig. 1. Pixel structure at RGB images

3.2 LSB Steganography: A Basic Overview

In the LSB steganography technique, the last bit of each pixel is replaced (changed) with one bit of the secret text. For example, if a secret text of 100 characters is to be hidden in a cover image, 800 pixels are needed. This is because 100 letters require 800 bits (ASCII Code system) and for each bit, one pixel is needed. In the embedding process, there are two possibilities. The first one happens when changing the bit with the same bit type, 0 by 0, or 1 by 1. While the second possibility is to replace the bit with a different bit type, for example, 0 by 1 or 1 by 0, see **Fig. 2**.
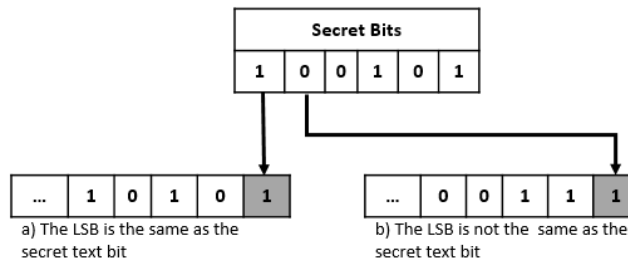
Fig. 2. illustrates image distortion in scenario (b) and the absence of distortion in scenario (a).

As for distortion of the stego-image, the first possibility produces no distortion since the original pixel bit remains unchanged. In the second possibility, distortion occurs in the original image due to a change in bit type. Other LSB techniques would require more LSBs for the embedding process [19], for instance, 2 LSBs, 3 LSBs, or even 4 LSBs, see **Fig. 3**. With more LSBs, the cover image payload would be higher, but distortion would be higher as well. This would result in poor quality and easy detection of the secret message.

Fig. 3. Illustrates LSB techniques necessitating a greater number of LSBs for the embedding process, such as 2 LSBs, 3 LSBs, or even 4 LSBs, leading to increased image distortion.

In summary, with dimensions of 256x256 pixels encompasses a total of 65,536 pixels, allowing for the concealment of around 8,192 characters if only one least significant bit (LSB) is utilized, and twice this size if two LSBs are employed. Earlier methods involved embedding through sequential or random pixel selection, as seen in [29][25] and [26]. However, these approaches ultimately pose limitations on payload capacity and the risk of compromising visual quality. The proposed technique in this research, BitPatternStego, addresses the crucial concerns of both image quality and payload capacity.

4. BitPatternStego Methodology

This study proposes to map the corresponding (identical) bit patterns in both the secret text and the cover image, called BitPatternStego. In the literature, selecting the cover image pixels for embedding the secret text is done sequentially or randomly.

BitPatternStego systematically scans the pixel channels of a cover image to determine the bit pattern corresponding to the alphabet (a-z) and some commonly used characters (!, ?, space, etc). Initially, the focus is solely on the blue channel, while potential utilization of other channels (red and green) to triple the payload capacity is reserved for future work. However, for the purposes of this study, the collected data from the blue channel proves sufficient.

After analyzing numerous commonly used images on the internet, it is observed that each character may have thousands of identical blue channels corresponding to its bit pattern (refer to appendices B, C, D). Consequently, the BitPatternStego technique compiles all indexes corresponding to each character into a separate array.

During the embedding process, a random selection of indexes is made for each character, enhancing security compared to sequential embedding. The result is an unchanged output image quality, as no alterations are made to the bit types (refer to **Fig. 2**). Furthermore, the cover image's payload becomes virtually limitless, accommodating a substantial volume of number of characters, as their corresponding indexes are subject to repetitive selection as needed.

The embedding and extraction processes of BitPatternStego are inherently reversible and straightforward. During the embedding process, as outlined earlier, the cover image is scanned to identify pixels with bit patterns corresponding to alphabet characters and symbols. Subsequently, secret text characters are assigned to the respective pixels' indexes, resulting in an array of indexes for the secret text characters (see **Fig. 4**).

Fig. 4. displays an array where each number corresponds to a specific pixel index in the stego-image, containing an identical bit pattern to the secret message characters in sequential order.

For the extraction process, it is imperative to determine the indexes array, stego-image name, and size. Utilizing the ASCII code system, the recipient must read the bits of the pixel channel (specifically the blue channel) indicated by the indexes array and convert them into characters, thereby reconstructing the complete secret text, as will be explained in detail in section 5.3.3, "Extraction Stage."

4.1 BitPatternStego Stage 1: The cover image.

1. Choose a JPEG file for the cover image. (which supports 24-bits of RGB color per pixel).
2. Flattening the cover image indexes. which involves copying the 24 bits of each pixel into the cells of one dimensional array (1D array). In this process, each cell within the new array accommodates 24 bits of each pixel, as illustrated in **Fig. 5**. It's noteworthy that this flattening step is crucial for minimizing the size of the resulting indexes array; Instead of having two coordinate values (row and column) for each pixel, it produces only one single coordinate value for each pixel, which will be referred to as the index.

   However, the index can be easily calculated based on the two coordinate values (row, column) of the pixel and the image's width (number of columns). The formula for this computation is:

$$Index = Row \times I_{width} + Column$$

As illustrated in **Fig. 5**, the index (2,3) can be substituted with the single index (11) through the calculation 2 * 4 + 3.
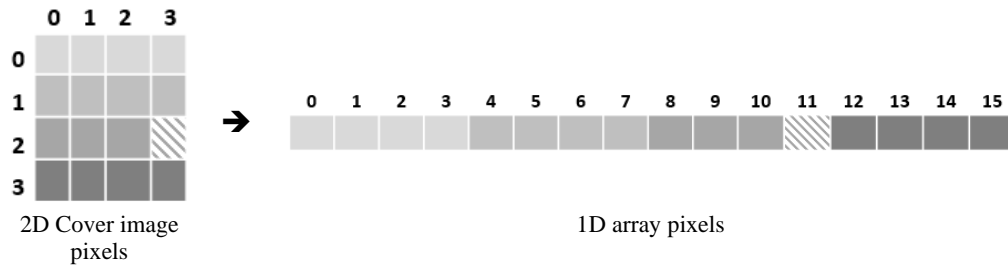
2D Cover image
pixels

1D array pixels

Fig. 5. Flatten the coordinates of the cover image (represented by two coordinates) into a one-dimensional array (indicated by a single coordinate index).

3. Scan (read) the 8-bits of the blue channel at each cell in the 1D array.
4. Create individual arrays to store corresponding indexes for each character, where, the 8-bit pattern of the blue channel corresponds with the alphabet and commonly used characters (refer to Appendices B, C, D). see sample **Table 1**.

Table 1. depicts the single-coordinate indexes where the 8-bit pattern of the blue channel corresponds to each alphabet character

Char	Indexes for each character					
a	125	377	424	427	510	...
b	45	214	347	663	924	...
c	199	254	288	314	556	...
.	.	.	.	.	.	...

Here is the pseudocode for stage 1 of BitPatternStego:

0	**procedure Characters-In-Cover-Image** (*cover-image*, *char-list*)
1.	Open cover-image file.
2.	Read cover-image data.
3.	Close cover-image file.
4.	Flatten cover-image into 1D-array.
5.	Scan blue channel of each cell in 1D-array.
6.	Create char-index-array
7.	For each cell (pixel) in 1D-array
8.	If blue channel value corresponds to a character in char-list
9.	Increment char-count for that character
10.	Add the index of the cell to char-index-array for that character
11.	End-If
12.	End-For
13.	Return char-index-array
14.	End-Procedure

4.2 BitPatternStego Stage 2: The secret message.

1. Select the text that needs to be embedded.
2. Convert text characters to lowercase. (e.g. 'A' to 'a').
3. For every character in the secret text, randomly assign one index from the array of indexes (where the 8-bit pattern of the Blue-channel corresponds to the given character). (See **Fig. 6**).

Text	h	e	l	p		m	e
Indexes	325	547	124	21	210	254	147

Fig. 6. Assign randomly one coordinate index of the cover image to each character of the secret text "help me", where the 8-bits of blue channel corresponds.

4. Establish an array to store the selected indexes corresponding to characters in the secret text. Consequently, a string of 10 characters requires an array of size 10, each holding a unique index (refer to **Fig. 6**).

Here is the pseudocode for stage 2 of BitPatternStego:

0	**procedure Embed-Text(*text*, *char-index-array*)**
1.	Converted_text = convert_to_lowercase(text)
2.	Index_storage = create_array(size(converted_text))
3.	For i = 1 to size(converted_text)
4.	character = converted_text[i]
5.	random_index = randomly_select_index(char-index-array, character)
6.	index_storage[i] = random_index
7.	End-For
8.	Return index_storage
9.	End-Procedure

5. BitPatternStego Implementation

5.1 Underlining Technologies

The Java Programming Language has been utilized to implement BitPatternStego using the NetBeans IDE. Java is a powerful programming language with several classes offering a variety of image manipulation tools and techniques. In terms of image processing using Java, a certain type of object with the name 'BufferedImage' needs to be implemented and created. Through this object, all methods concerning image editing can be accessed. Nevertheless, it is worth mentioning here that Python could be another option for implementation as it is one of the easiest programming languages to learn and utilize since it has a wide array of built-in libraries designed to help with different aspects of programming. Despite that, using Java wasn't difficult as we had mastered it.

5.2 BitPatternStego Dataset

In this study, BitPatternStego is implemented using a dataset consists of 18 standard test images, which vary in sizes such as 512x512, 256x256, or 2560x1150 pixels. Refer to Appendix A for details. The images were in JPEG format which is ideal for RGB files as it is a reasonable middle-ground between file size and quality, and it can be read almost anywhere. Other formats can be used as well, but they are not ideal for RGB like PNG and GIF. Other formats, such as TIFF, EPS, PDF, and BMP, should be avoided for RGB purposes. These formats are not compatible with most software and can be unnecessarily large in terms of data [30].

As mentioned earlier, that RGB images are made up of red, green, and blue color channels. And each pixel in the image is represented by a combination of these three primary colors. These colors (red, green, or blue) are typically represented using 8 bits per color channel (3 X 8 = 24 bits per pixel), allowing for 16.7 million possible color combinations.

In an RGB image, the color of each pixel is determined by the intensity of each color channel. RGB images are widely used in digital photography, computer graphics, and other applications where color accuracy is paramount. They are also the standard format for displaying images on computer monitors and other electronic devices [31].
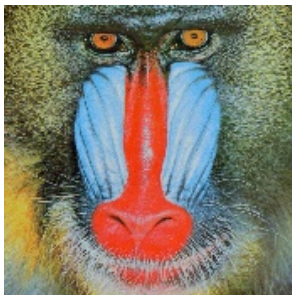
5.3 BitPatternStego Implementation Example.

5.3.1 Embedding stage

Consider the following inputs for embedding secret text into a cover image:

- Secret text: "tic tac toe"
- Cover Image: mandril_color.jpg (see **Fig. 7.a**).

After scanning the cover image, mandril_color.jpg, **Fig. 7.b** displays the indexes corresponding to the most frequently used characters and symbols.

Char	#	Char	#	Char	#	Char	#	Char	#
a	1579	k	1459	u	1258	?	2035	0	1866
b	1551	l	1432	v	1188	!	1360	1	1943
c	1498	m	1355	w	1141	+	1664	2	1913
d	1563	n	1358	x	1180	-	1851	3	1889
e	1486	o	1286	y	1124	*	1769	4	1858
f	1470	p	1302	z	1161	/	1837	5	1969
g	1330	q	1336			=	2086	6	2029
h	1455	r	1224			_	1546	7	1938
i	1365	s	1247			.	1875	8	1949
j	1411	t	1186			Space	1375	9	1974

(a) mandril_color.jpg   (b) amount of numbers corresponding bit patterns for each character

Fig. 7.a. 512 X 512 cover image size in jpg format. **b.** Upon scanning the cover image, the amount of bit patterns corresponds to each character/symbol is calculated.

For instance, the letter 'a' can be represented by 1579 indexes of identical bit pattern in the cover image. The 1579 indexes are stored in a separated array for each character. See **Table 1**.

As depicted in **Fig. 7.b**, numerous pixels (indexes) exhibit identical bit patterns for each character. This implies the potential to generate millions of distinct encryption keys for a single secret message. Presented below are just five keys for the provided secret text (refer to **Fig. 8**).

	t	i	c	t	a	c	t	o	e
Key 1	207365	43636	193893	113068	208313	137235	226731	165923	93721	78705	52582
Key 2	72702	249311	206570	33536	37759	21544	221189	132787	20487	222075	125362
Key 3	192667	96729	250016	2660	17114	259466	231661	225822	249267	240779	230775
Key 4	87877	219588	45555	119971	144747	191628	238053	138810	49289	166739	238485
Key 5	226566	138094	163793	133553	202404	12217	196427	194154	215860	122753	245876

Fig. 8. Illustrate five district keys for the given secret message "tic tac toe"

Significantly, in the provided example, the alphabet character 't' is observed with 15 different indexes among the five generated keys, each sharing an identical bit pattern. These 15 indexes represent only a fraction of the total number of indexes for the letter 't,' which amounts to 1186, as illustrated in **Fig. 7.b.** The BitPatternStego algorithm employs a random selection of indexes for each character, potentially resulting in the repetition of index selection, further elaboration on this aspect will be provided in the subsequent section.

Various randomization techniques, recognized for their efficacy in enhancing encryption security [26][32] and [33], have been widely applied in steganography. Consequently, the random selection of indexes serves to mitigate the vulnerabilities associated with breaking ciphers through methods that rely on letter frequency analysis, such as those employed in mono-alphabetic substitution ciphers, Caesar shift ciphers, and Vatsyayana cipher

The proposed technique, BitPatternStego, incorporates various intermediate encryption methods to enhance security. This includes mapping the ASCII code of the text to the image RGB code and employing a randomization technique for selecting embedding pixels. Additionally, the encrypted text is structured as a set of single coordinate values instead of two coordinates for the indexes. While the resulting encrypted text can undergo established encryption techniques like AES, Homomorphic Encryption, Paillier Cryptosystem, or Lattice-based Encryption for heightened complexity and security, our research intentionally maintains simplicity. This decision leaves room for future enhancements and improvements in the ongoing exploration of these concepts.

5.3.2 Index Repetition and Cover Image Suitability

BitPatternStego algorithm employs a random selection of indexes for each character, which can lead to instances of index repetition. Such repetition may occur coincidentally or when the hidden message is extensive and the cover image is small, resulting in a limited number of indexes for certain letters. Therefore, it is recommended to utilize appropriately sized cover images that match the text size, along with rich cover images containing a sufficient source of data bits.

For illustration, in **Fig. 9**, a secret message containing 100 words (618 letters) demonstrates differing repetition patterns across two cover images of identical dimensions. In the first cover image, labeled 'House', all letters, except for two instances of index repetition, do not repeat. Similarly, in the 'mandril_colore' cover image, the majority of letters do not repeat, with only six exceptions. It's important to highlight that even repeated trials with the same cover image

can yield varying numbers of repetitions, hence concealing secret text in various ways. This variability adds an additional layer of security and complexity to the concealment process, making it more robust against potential decryption attempts.

When managing longer secret messages, such as a 5000-word (27144-character) composition, within images of two different sizes (512 x 512 and 2560 x 1150 pixels), a significant increase in the occurrence of repeated indexes can be observed in the smaller size image. Conversely, the larger size image exhibits far fewer repetitions, as well as many letters still show no repetitions, as demonstrated in **Fig. 10**.
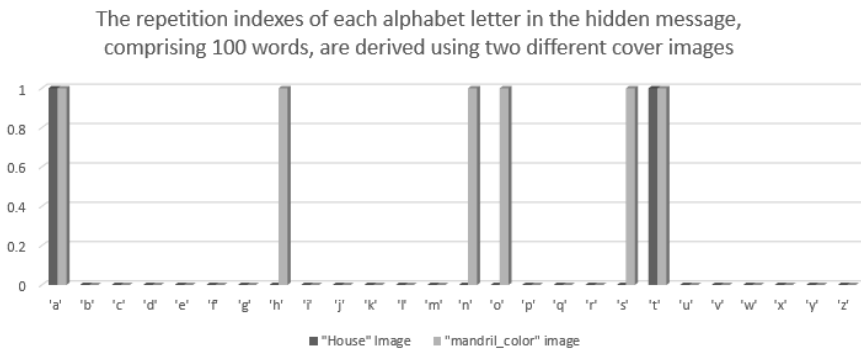
Fig. 9. The number of repeating indexes in two images for a 100-word secret message differs slightly. The House image (black) exhibits fewer repetitions compared to the mandril_color image (grey), indicating that it contains a greater abundance of indexes for each letter.
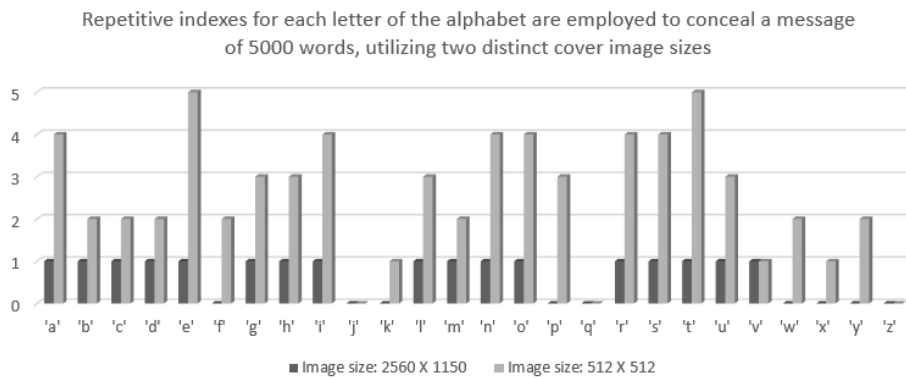

Fig. 10. illustrates the frequencies of index repetition observed in two cover images of different sizes (512x512 in grey and 2560x1150 in black), each concealing a 5000-word message.

The concept is that as the data volume increases, so does the frequency of indexes, regardless of the quantity of data. Yet, the repetition of indexes does not yield evident outcomes, complicating the task of letter prediction, especially with the inclusion of additional characters and symbols in the secret message (e.g., spaces, dots, numbers, etc.).

Accordingly, it is fundamental to acknowledge that factors such as image size, pixel density, and available bit patterns play a significant role in dispersing index frequencies. Careful consideration of these factors is essential to mitigate the risk of statistical analysis or detection.

When considering the suitability of a cover image, it's crucial to recognize that certain images may lack the necessary bit patterns in their blue channels to accommodate specific

letters, rendering them incompatible with the proposed method. For example, in Appendix D, images 5 ("woman_darkhair") and 16 ("lena_color_512") demonstrate a deficiency of indexes for the "space" character, indicated by the absence of any indexes. Consequently, it is prudent to confirm the availability of corresponding bit patterns for all characters within the cover image prior to employing the BitPatternStego method.

5.3.3 Extraction stage

The decryption process for the BitPatternStego method relies on several key elements: the indexes array, which indicates the pixel locations in the image corresponding to characters of the secret message, the agreed-upon image name, and its size.

Using these components, the recipient retrieves the bits from the blue channel of the specified indexes (pixels) and converts them into text data using the ASCII code system to reconstruct the original message.

For example, if key-1 in **Fig. 8**, along with the image name and size (mandril_color.jpg, 512x512), is provided to the recipient, they can extract the corresponding bit patterns to the characters of the secret message. This can be achieved by either flattening the stego-image, as described in point 2 of section 4.1, or converting each value of key-1 (indexes array) into two coordinate values (row, column) for the image.

To clarify, according to key-1, the first character of the secret message is located at index 207365 of the stego-image. If the image has been flattened, the recipient can directly extract the bits from the specified index. Otherwise, the recipient needs to convert each index into two coordinate values (row, column) of the image. This conversion involves a straightforward calculation:

$$\text{Row} = \text{index } Div \text{ I}_{width},$$
$$\text{Column} = \text{index } mod \text{ I}_{width}$$

Where, *div* is the integer division, *mod* is the modulus operation, and I_{width} is the Image width (number of image's columns).

Therefore,

$$\text{Row} = 207365 \; Div \; 512 = 405,$$
$$\text{Column} = 207365 \; mod \; 512 = 5$$

This suggests that index 207365 in the indexes array corresponds to row 405 and column 5 of the stego-image. Consequently, the recipient must extract the blue channel bits at this specific pixel location, which will represent the character 't' according to the ASCII code (refer to **Fig. 11**).
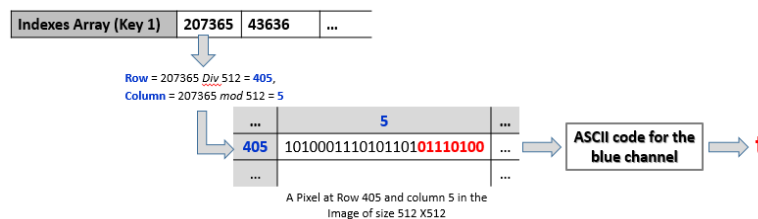

Fig. 11. Illustrates the procedure for extracting a single character of the secret message from the image.

5.4 Evaluating BitPatternStego

In the assessment of image steganography and other image processing methods, the effectiveness of steganographic algorithms is commonly evaluated using metrics such as Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR). This evaluation involves comparing the original image with the distorted images (stego-image). MSE calculates the average squared difference between each pixel in the original and distorted images, as shown in equation (1).

$$MSE = \frac{\sum(I\_original - I\_distorted)^2}{I_{width} * I_{height}} \tag{1}$$

PSNR is determined by employing the MSE and the maximum pixel value, as illustrated in equation (2)

$$PSNR = 10 * log_{10} \frac{(\max pixel\ value)^2}{MSE} \tag{2}$$

The result is usually expressed in decibels (dB). A lower MSE value indicates greater similarity between the stego-image and the original image, which is a desirable outcome in steganography. However, relying solely on MSE may not be adequate for assessing stego-image quality, as it overlooks visual considerations. PSNR, in contrast, offers a more comprehensive evaluation by considering both visual quality and the differences between the original and stego-images. A higher PSNR value suggests that the stego-image closely resembles the original image and exhibits superior visual quality.

Accordingly, The BitPatternStego method effectively conceals information without altering any bits within the cover image. As it scans the cover image to locate bit patterns aligning with those of the secret message. Consequently, there is no meaningful deviation on the calculation of MSE or PSNR, as the cover image remains unchanged.

However, when calculating the MSE and PSNR for two images, the "mandril_color" and the "lena_gray_512", before and after implementing the BitPatternStego process, the resulting MSE is 0, and the PSNR is Infinity decibels (dB) for both images. (refer to **Table 2**). This scenario arises when the stego-image precisely matches the original, resulting in no difference between them.

Table 2. Calculating MSE and PSNR for two images before and after extracting the corresponding bits pattern

Cover Image Name	MSE	PSNR
mandril_color.jpg	0.0	Infinity decibels (dB)
lena_gray_512.jpg	0.0	Infinity decibels (dB)

Consequently, when comparing the MSE and PSNR outcomes of the proposed method to those of most, if not all, LSB methods, our results exhibit superiority. Furthermore, in terms of cover image payload, which constitutes one of the primary concerns for the majority of LSB techniques, BitPatternStego has demonstrated its capability to handle a substantial volume of data, potentially unlimited, as discussed in section 5.3.2. This once again underscores its superiority in this aspect as well.

The BitPatternStego method's robustness stems from its capability to conceal data without altering any bits of the cover image, thus preserving its visual fidelity and evading detection. However, the susceptibility to attacks may fluctuate depending on factors like the randomness of index selection and the diversity of embedding symbols, of the secret message, employed in the process. Furthermore, possessing knowledge of the image name and size, along with the indexing array, significantly bolsters the algorithm's resistance to attacks.

The algorithm's complexity is moderate, involving tasks such as scanning the cover image, mapping bit patterns, selecting random indexes, and embedding text. Nevertheless, it can become computationally intensive when dealing with large images or extensive text.

6. Conclusion

Securing digital data transmission is paramount in contemporary times, prompting the implementation of diverse techniques to frustrate unauthorized access. Among these, steganography stands out as a potent method for safeguarding transmitted digital data. Despite its efficacy, certain steganography techniques face limitations, such as constraints on cover-image payload and potential compromises to visual quality, influencing overall security. Striking a delicate balance between concealing data and preserving carrier integrity underscores the need for ongoing refinement in the realm of digital security measures.

In this paper, scholars introduce a steganography method suggesting the alignment of identical 8-bit patterns between the characters of a secret message and the blue channel of the pixels in a cover image. This approach ensures that the bit-type of the stego-image remains unchanged, maintaining the quality of the stego-image identical to the original. Furthermore, the payload capacity of the cover image is significantly enhanced, becoming virtually limitless, as character indexes can be repetitively utilized on random bases. This technique ensures covert data transmission without raising suspicion

References
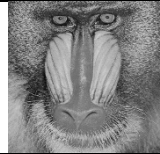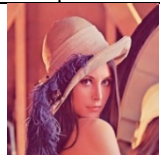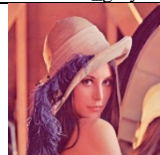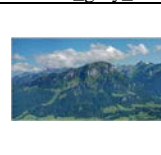
[1] M. Kharrazi, H. T. Sencar, and N. Memon, "Improving steganalysis by fusion techniques: A case study with image steganography," in *Proc. of Transactions on Data Hiding and Multimedia Security I*, pp.123-137, Springer, 2006. Article (CrossRef Link)

[2] K. Muhammad, M. Sajjad, I. Mehmood, S. Rho, and S. W. Baik, "Image steganography using uncorrelated color space and its application for security of visual contents in online social networks," *Future Generation Computer Systems*, vol.86, pp.951-960, 2018. Article (CrossRef Link)

[3] P. C. Mandal, I. Mukherjee, G. Paul, and B. N. Chatterji, "Digital image steganography: A literature survey," *Information Sciences,* vol.609, pp.1451-1488, 2022. Article (CrossRef Link)

[4] P. I. Sharikov, A. V. Krasov, A. M. Gelfand, and N. A. Kosov, "Research of the Possibility of Hidden Embedding of a Digital Watermark Using Practical Methods of Channel Steganography," in *Proc. of Intelligent Distributed Computing XIII*, vol.868, pp.203-209, Springer, 2020. Article (CrossRef Link)

[5] Xin Liao, Zheng Qin, Liping Ding, "Data embedding in digital images using critical functions," *Signal Processing: Image Communication*, vol.58, pp.146-156, 2017. Article (CrossRef Link)

[6] Alan A. Abdulla, Harin Sellahewa, and Sabah A. Jassim, "Stego Quality Enhancement by Message Size Reduction and Fibonacci Bit-plane Mapping," in Proc. of *Security Standardization Research - Springer*, vol.8893, pp.151-166, 2020. Article (CrossRef Link)

[7] K. A. Kingsley and A. M. Barmawi, "Improving Data Hiding Capacity in Code Based Steganography using Multiple Embedding," *Journal of Information Hiding and Multimedia Signal Processing*, vol.11, no.1, pp.14-43, 2020. Article (CrossRef Link)

[8] B. Li, S. Tan, M. Wang, and J. Huang, "Investigation on Cost Assignment in Spatial Image Steganography," *IEEE Transactions on Information Forensics and Security*, vol.9, no.8, pp.1264-1277, 2014. Article (CrossRef Link)

[9] W. Mazurczyk and L. Caviglione, "Steganography in Modern Smartphones and Mitigation Techniques," *IEEE Communications Surveys & Tutorials*, vol.17, no.1, pp.334-357, 2015. Article (CrossRef Link)

[10] L. Rura, B. Issac, and M. K. Haldar, "Online Voting System Based on Image Steganography and Visual Cryptography," *Journal of Computing and Information Technology*, vol.25, no.1, pp.47-61, 2017. Article (CrossRef Link)

[11] L. Guo, J. Ni, and Y. Q. Shi, "Uniform Embedding for Efficient JPEG Steganography," *IEEE Transactions on Information Forensics and Security*, vol.9, no.5, pp.814-825, 2014. Article (CrossRef Link)

[12] M. M. Emam, A. A. Aly, and F. A. Omara, "An Improved Image Steganography Method Based on LSB Technique with Random Pixel Selection," *International Journal of Advanced Computer Science and Applications*, vol.7, no.3, pp.361-366, 2016. Article (CrossRef Link)

[13] K. Muhammad, J. Ahmad, N. U. Rehman, Z. Jan, M. Sajjad, "CISSKA-LSB: color image steganography using stego key-directed adaptive LSB substitution method," *Multimedia Tools and Applications*, vol.76, pp.8597-8626, 2017. Article (CrossRef Link).

[14] M. Sajjad, K. Muhammad, S. W. Baik, S. Rho, Z. Jan, S.-S. Yeo, I. Mehmood, "Mobile-cloud assisted framework for selective encryption of medical images with steganography for resource-constrained devices," *Multimedia Tools and Applications*, vol.76, pp.3519-3536, 2017. Article (CrossRef Link).

[15] J. Mielikainen, "LSB matching revisited," *IEEE Signal Processing Letters*, vol.13, no.5, pp.285-287, 2006. Article (CrossRef Link)

[16] A. Anees, A. M. Siddiqui, J. Ahmed, I. Hussain, "A technique for digital steganography using chaotic maps," *Nonlinear Dynamics*, vol.75, pp.807-816, 2014. Article (CrossRef Link).

[17] P. S. Narule, N. D. Patil, P. S. Kurade, P. D. Patil, and J. M. Waykule, "ARM Controller Based Image Steganography Using LSB Algorithm," *International Journal of Advanced Research in Computer and Communication Engineering*, vol.4, no.4, pp.132-136, 2015. Article (CrossRef Link)

[18] Z. Xia, X. Wang, X. Sun, Q. Liu, and N. Xiong, "Steganalysis of LSB matching using differences between nonadjacent pixels," *Multimedia Tools and Applications*, vol.75, np.4, pp.1947-1962, 2016. Article (CrossRef Link)

[19] D. Laishram and T. Tuithung, "A novel minimal distortion-based edge adaptive image steganography scheme using local complexity," *Multimedia Tools and Applications*, vol.80, no.1, pp.831-854, 2021. Article (CrossRef Link)

[20] H. C. Wu, N. I. Wu, C. S. Tsai, and M. S. Hwang, "Image steganographic scheme based on pixel-value differencing and LSB replacement methods," *IEE Proceedings-Vision, Image and Signal Processing*, vol.152, no.5, pp.611-615, 2005. Article (CrossRef Link)

[21] A. Ioannidou, S. T. Halkidis, G. Stephanides, "A novel technique for image steganography based on a high payload method and edge detection," *Expert Systems with Applications*, vol.39, no.14, pp.11517-11524, 2012. Article (CrossRef Link).

[22] W. Luo, F. Huang, and J. Huang, "Edge Adaptive Image Steganography Based on LSB Matching Revisited," *IEEE Transactions on Information Forensics and Security*, vol.5, no.2, pp.201-214, 2010. Article (CrossRef Link)

[23] H. R. Kanan and B. Nazeri, "A novel image steganography scheme with high embedding capacity and tunable visual image quality based on a genetic algorithm," *Expert Systems with Applications*, vol.41, no.14, pp.6123-6130, 2014. Article (CrossRef Link)

[24] C. K. Chan and L. M. Cheng, "Hiding data in images by simple LSB substitution," *Pattern Recognition*, vol.37, no.3, pp.469-474, 2004. Article (CrossRef Link)

[25] L. Y. Por, D. Beh, T. F. Ang, and S. Y. Ong, "An enhanced mechanism for image steganography using sequential colour cycle algorithm," *The International Arab Journal of Information Technology (IAJIT)*, vol.10, no.1, pp.51-60, 2013. Article (CrossRef Link)

[26] K. H. Abuhmaidan, A. K. Kayed, and M. Alrisia, "Steganography: A Flexible Embedded Randomization Technique," *KSII Transactions on Internet and Information Systems*, vol.17, no.1, pp.120-144, 2023. [Article (CrossRef Link)](#)

[27] S. Singh and G. Agarwal, "Use of image to secure text message with the help of LSB replacement," *International Journal of Applied Engineering Research*, vol.1, no.1, pp.200-205, 2010. [Article (CrossRef Link)](#)

[28] S. A. Laskar and K. Hemachandran, "High Capacity data hiding using LSB Steganography and Encryption," *International Journal of Database Management Systems (IJDMS)*, vol.4, no.6, pp.57-68, 2012. [Article (CrossRef Link)](#)

[29] K. Tiwari, S. Gangurde "LSB Steganography Using Pixel Locator Sequence with AES," *in Proc. of International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pp.302-307, 2021. [Article (CrossRef Link)](#)

[30] A. J. Qasim and F. Q. A. Alyousuf, "History of Image Digital Formats Using in Information Technology," *QALAAI ZANIST JOURNAL*, vol.6, no.2, pp.1098-1112, 2021. [Article (CrossRef Link)](#)

[31] M. Owens, "A Discussion of Covert Channels and Steganography," *SANS institute*, vol.1, pp.1-18, 2002. [Article (CrossRef Link)](#)

[32] A. Jain, "A Secured Steganography Technique for Hiding Multiple Images in an Image Using Least Significant Bit Algorithm and Arnold Transformation," in *Proc. of Intelligent Data Communication Technologies and Internet of Things*, pp.373-380, 2019. [Article (CrossRef Link)](#)

[33] Hussein Albazar, Ahmed Abdel-Wahab, Marwan Alshar'e, and Abdallah Abualkishik, "An Adaptive Two-Factor Authentication Scheme Based on the Usage of Schnorr Signcryption Algorithm," *Informatica*, vol.47, no.5, pp.159-172, 2023. [Article (CrossRef Link)](#)

Appendix

Appendix A: The dataset of analyzed JPEG images in this study

(1) house	(2) jetplane	(3) livingroom	(4) cameraman	(5) woman_darkhair	(6) woman_blonde
(7) lake	(8) peppers_gray	(9) pirate	(10) mandril_gray	(11) lena_gray_512	(12) lena_gray_256
(13) walkbridge	(14) peppers_color	(15) lena_color_256	(16) lena_color_512	(17) mandril_color	(18) Fronalpstock big

Appendix B: Illustrates the distribution of bit patterns (pixels) corresponding to each alphabet character in every image presented in Appendix A.

char	Number of the Image																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	1994	859	1323	514	1138	978	801	2155	1398	1537	1809	473	1926	1377	781	3024	1579
b	2891	900	1375	566	1138	1000	762	1959	1336	1565	1881	482	1783	1298	731	3030	1551
c	4363	865	1407	591	1154	1100	795	1744	1473	1599	1896	454	1898	1234	738	2946	1498
d	5715	824	1438	634	1116	1113	738	1419	1305	1579	2012	493	1932	1192	661	2778	1563
e	6333	858	1504	694	1070	998	757	1123	1226	1599	1966	464	1903	1077	644	2626	1486
f	6584	897	1674	740	1032	921	720	923	1247	1758	1989	468	1936	1005	631	2419	1470
g	6112	890	1797	772	979	948	736	770	1307	1713	1843	457	1862	995	616	2505	1330
h	5799	822	1856	791	991	930	709	739	1224	1698	1805	463	1843	930	621	2466	1455
i	5992	789	1887	879	930	937	706	748	1298	1765	1684	454	1841	872	661	2564	1365
j	6198	840	1967	851	979	924	681	846	1264	1758	1635	416	1734	837	635	2656	1411
k	6199	855	2044	990	963	941	608	940	1295	1846	1437	392	1749	827	742	2705	1459
l	5796	818	2068	985	912	954	676	1094	1275	1880	1395	365	1776	813	775	2758	1432
m	5148	843	2221	1023	858	929	587	1247	1355	1946	1382	363	1677	787	734	2881	1355
n	4337	812	2376	1089	763	924	601	1444	1245	2067	1386	363	1727	803	700	2754	1358
o	3370	839	2544	1069	803	999	612	1459	1315	2099	1373	324	1556	751	678	2767	1286
p	2885	892	2528	1147	778	1036	656	1355	1391	2268	1399	332	1635	665	673	2631	1302
q	2521	856	2638	1226	765	1041	608	1225	1434	2390	1371	326	1625	628	610	2619	1336
r	2272	889	2681	1186	728	1054	581	1116	1420	2615	1429	355	1666	634	617	2409	1224
s	2062	807	2754	1205	749	1031	616	1063	1496	2520	1345	344	1612	626	651	2460	1247
t	1772	762	2788	1292	749	1154	590	966	1454	2643	1433	321	1655	593	643	2602	1186
u	1601	786	2833	1308	730	1164	548	932	1540	2779	1451	342	1704	616	711	2620	1258
v	1296	812	2714	1422	785	1236	591	941	1562	2806	1557	387	1689	592	648	2722	1188
w	1140	758	2640	1453	806	1229	528	943	1569	2968	1538	394	1606	535	751	2768	1141
x	842	811	2614	1425	815	1272	578	901	1582	2826	1567	410	1592	497	774	2992	1180
y	718	748	2644	1534	788	1314	492	872	1595	2912	1661	443	1625	493	828	3224	1124
z	688	733	2705	1499	886	1320	524	776	1702	2860	1739	485	1591	447	849	3437	1161

Appendix C: Illustrates the distribution of bit patterns (pixels) corresponding to each numeric character in every image presented in Appendix A.

char	Number of the Image																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1336	147	772	260	6028	130	2857	857	1755	506	2035	478	1132	1315	74	210	1866
1	1357	153	810	260	5312	147	2861	818	1588	537	2072	554	1217	1276	95	293	1943
2	1183	120	866	251	5092	137	2761	1021	1444	564	2035	488	1244	1276	113	407	1913
3	789	115	867	245	4728	136	2754	1141	1320	562	2102	517	1268	1169	157	509	1889
4	647	111	881	258	4489	160	2719	972	1293	615	1950	532	1335	1177	197	593	1858
5	578	124	840	261	4017	140	2744	751	1192	627	1840	439	1309	1198	198	753	1969
6	580	120	870	279	3800	182	2460	775	1174	702	1677	484	1351	1186	212	996	2029
7	528	130	825	266	3682	186	2451	770	1195	643	1647	424	1341	1254	295	1173	1938
8	580	156	841	297	3395	191	2309	941	1152	657	1589	376	1494	1275	339	1352	1949
9	571	118	928	288	3126	199	2306	1161	1163	704	1429	351	1403	1367	382	1584	1974

Appendix D: Illustrates the distribution of bit patterns (pixels) corresponding to each symbol in every image presented in Appendix A.

char	Number of the Image																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
?	1104	250	904	349	2007	412	1745	523	1193	774	1003	246	1683	1799	664	2720	2035
!	668	189	414	412	0	99	1159	364	30	225	265	82	785	4242	3	2	1360
+	272	124	564	318	4876	132	2492	458	3257	370	1592	396	1018	2071	36	48	1664
-	371	120	632	281	9562	129	2785	629	2653	429	1887	443	1051	1610	54	76	1851
*	234	132	622	346	2344	108	2421	481	4318	369	1457	392	892	2470	23	39	1769
/	1046	134	747	275	6853	125	2867	843	1976	432	1967	497	1101	1376	65	171	1837
=	763	149	808	339	2352	319	1940	788	1138	776	1029	260	1529	1667	633	2505	2086
_	1285	874	1218	421	1091	913	790	2405	1475	1481	1546	364	1817	1597	901	3294	1546
.	589	139	692	265	7786	123	2727	737	2271	440	2072	449	1054	1497	60	112	1875
Space	740	186	398	370	0	91	1023	397	9	228	172	52	714	4073	4	0	1375

Khaled Abuhmaidan received his B.S. degree in computer information systems from Applied Science University, Amman, Jordan in 1998, his M.S. degree in applied computer science from Free University of Brussels, Belgium in 2001, and Ph.D. degrees in applied mathematics and computer science from Eastern Mediterranean University (EMU), North Cyprus, in 2019. Since 2001, Abuhmaidan has served as a faculty member at several universities, including those in Jordan, Saudi Arabia, Northern Cyprus, and Oman. Abuhmaidan has published a host of papers on Digital Geometry, Pattern Recognition, Image Processing, multimedia, and Steganography. His current research interest is in semantic textual similarity.

Dr. Marwan I. Alshar'e, an Assistant Professor of Computer Science at Sohar University, holds a Ph.D. in Computer Science with expertise in cybersecurity. His extensive career spans teaching, research, and impactful administrative roles worldwide. Passionate about cutting-edge solutions, he actively contributes to Cyber Security, AI, Blockchain, e-learning, and IoT fields, shaping the future of computer science and influencing students globally.

Dr. Abdallah Mohd Abualkishik, an Associated Professor at Sohar University, holds a Ph.D. in AI & Software Engineering. His rich academic journey includes post-doctoral service, Senior Lecturer roles, and mentorship of Ph.D. and Master's students. Acknowledged for outstanding teaching, he received the Vice Chancellors Outstanding Teaching Award. As program coordinator, he shapes the Network & Database program at Sohar University, ensuring industry relevance. An accomplished researcher, his work spans AI, IT, and smart systems, earning grants from the Oman Research Council. Beyond academia, Dr. Abualkishik, with Huawei certifications, contributes as a reviewer for the MoHE in Oman, showcasing his commitment to advancing computer science.

Prof. Ahmad Kayed, Dean of FCIT Faculty –Sohar University, was awarded his Ph.D. from one of the leading Australian universities (Queensland Uni. -2003) and worked with Monash University –Australia. Prof. Kayed has been involved in both the academic and IT industries for almost 30 years. In the IT industry, he held positions as a Solutions Architect, project manager, general manager, and IT consultant. In academics, Kayed worked with Monash Australia and supervised more than 8 Ph.D. and 40 M.Sc. students, created and evaluated several IT-related curricula, several IT programs at B.Sc., M.Sc., and Ph.D. levels, and several IT projects. He worked with several quality assurance committees for internal and international accreditations. Kayed has published more than 50 papers in high-impact journals. He is working in the area of cloud computing, DB security, and semantic web in particular semantic and conceptual engineering for securing cloud DB.