Steganography: A Flexible Embedded Randomization Technique

Khaled H. Abuhmaidan[1*], Ahmad K. Kayed[2], and Maryam Alrisia[3]
Faculty of Commuting and Information Technology, Sohar University, Sohar, Oman
[1] [e-mail: khmaidan@su.edu.om, kabuhumaidan@yahoo.com]
[2] [e-mail: akayed@su.edu.om, drkayed@ymail.com]
[3] [e-mail: maryamal_risia@yahoo.com]
[*]Corresponding author: Khaled H. Abuhmaidan

Received February 10, 2022; revised December 13, 2022; accepted December 26, 2022; published January 31, 2023

Abstract

With the expansion of digital communication networks, a considerable number of randomization techniques have been invented and implemented to enhance the different data transmission algorithms' levels of security. Steganography is among the data transmissions techniques used to hide secret data. Nowadays, several randomization techniques have been used in steganography to elevate the security of transmitted data. Unfortunately, the majority of these techniques lack some simplicity, efficiency, and flexibility, in addition to other limitations. This research presents a new randomization technique called *Rand*-Stego. *Rand*-Stego could be applied/practiced over any steganography technique. It provides simplicity and efficiency and elevates the security level. Examples of implementing the proposed technique on some steganography algorithms will be explored. The proposed and current techniques will be compared. The obtained results show *Rand*-Stego's superiority in terms of efficiency and flexibility when compared to the current techniques.

Keywords: Encryption, Least Significant Bit (LSB), Randomization, Random key, Steganography, Security.

1. Introduction

With the expansion of digital communication networks, there is increased public demand to achieve data transmission privacy and security. One way to maintain this is to perform efficient security data transmission processes that eliminate the chances of data leakage or extraction through irresponsible actions.

Steganography is the process of concealing digital secret data in different file formats (e.g. images) as a cover media. However, randomly concealing secret data would elevate any steganography algorithm's security level [1]. Therefore, several randomization techniques have been embedded in steganography algorithms for this purpose.

A modest steganography algorithm would increase the vulnerability of data to discover, extract, and attack. Randomization techniques play an important role in protecting and securing data against different types of attacks. However, there are many advantages to randomization techniques, such as being rapid and easy to handle different problems, easy implementation, and high probability with rapid and optimum output.

For centuries, encryption and steganographic techniques have been used to secure sensitive data. In cryptography, the secret data is changed into vague data, yet the secret information is still identifiable. Unlike encryption, steganography is about concealing the sensitive data to avoid it being identifiable [2].

Different randomization techniques are heavily implemented in encryption and steganography. For instance, the randomization features employed in encryption gained remarkable attention in web-based applications to prevent SQL injection attacks (SQLIA) (which is the most critical vulnerable as described in the top 10 web security issues by the Open Web Application Security Project [3], Also, randomization has been used in a concept named SQLrand [4] to encrypt SQL keywords. Furthermore, in [5], the randomised encryption is described as a procedure that enciphers a message by randomly choosing a ciphertext from a set of ciphertexts corresponding to the message under the current encryption key. Also, randomization techniques have been employed in the physical layer of the multiuser systems in order to improve a stream cipher's security [6]. However, in order to increase the encryption techniques' security levels, several approaches for embedding randomization techniques have been discussed in published studies that can be easily found on the web.

In a similar way, randomization has been extensively involved in steganography due to its efficiency. Several studies in steganography have been achieved for enhancing data privacy, developing, and finding out which modern methods make data transmission more secure [7]. Moreover, several authors have influenced the advantages of the randomization technique, each having its weaknesses and strengths. Hence, there exist different kinds of randomization techniques based on behaviour, such as NUBASI [8], which adds three-layer securities for concealing secret messages in a digital image; and Arnold [9], which uses the scrambling Arnold technique to scramble the cover image and improve the hidden image's security. There are also LDA [10], Henon map [11], and the Knight Tour [12]. Generally, concealing secret data using the randomization technique will enhance the steganography security [1], even if the attacker recognises the cover media.

The precision of identifying the required pixels/bits for hiding the secret data in steganography is one of the basics of the embedding methods. Moreover, it is not enough to identify the current embedded location, but also to determine a sequence of locations. Therefore, it is necessary to have an efficient and flexible technique that can be used to set/define the complete route of embedded locations.

The large number of steganography algorithms using randomization techniques motivates us to propose a new randomization technique that is more flexible and can subsume these and other techniques. This is the instant paper's main contribution.

This research presents a new technique called *Rand*-Stego. The efficiency and flexibility of *Rand*-Stego over other randomization techniques are shown as well. Here, the *Rand*-Stego will be applied to one of the noteworthy steganography algorithms. The steganography algorithm in [13], which we will call *Muh-algo*, will be used to apply *Rand*-Stego. Embedding *Rand*-Stego in *Muh-algo* will enhance the level of its security without compromising the image quality. The results before and after embedding *Rand*-Stego to *Muh-algo* will be compared. The comparison shows *Rand*-Stego's superiority in terms of efficiency and flexibility against the available techniques. Moreover, the *Rand*-Stego could be embedded into any steganography technique as we will see later.

2. Background and Related Works

Image steganography is one of many branches of information security, where secret data can be hiding in images such that it cannot be detected by the human visual system (HVS) [14]. This technique has extensive research interests because of its superiority over some limitations of cryptographic methods which have enormous computational complexity in addition to fascinating the attackers' attention [15]. Therefore, image steganography is applied in many useful applications, such as secure mobile computing [16], securing online voting systems, captioning and contents protection [17], and secure communication between two communicating parties [18].

Generally, there are two domains for implementing image steganographic methods: frequency and spatial domains [19]. The frequency-domain methods are based on manipulating the image's orthogonal transformation which has two components: the magnitude which consists of the image's frequency content and the phase that is used to restore the image back into the spatial domain. Therefore, these methods consist of an algorithm plus the transformed image. Frequency domain has more flexibility against the attacks on image processing, but is computationally complex with limited payload [20], making it inappropriate for many real-time applications [21]. On the other hand, spatial domain methods [19] [22] work directly with the image's pixels by replacing its least significant bit (LSB) according to the embedded sensitive data bits. Thus, the spatial domain algorithms are simpler, faster, more powerful, and more resistant to attack [23]. Moreover, it has a large payload that causes slight changes to the cover media (image or video); however, it shortages for flexibility against statistical attacks [24] [20].

The LSB substitution method [19] is one of the spatial domain techniques that is based on the RGB (Red Green Blue) colour model. Here, each pixel of the image is replaced with secret message bits that are based on a secret key. It starts from one bit per pixel (BPP) of the cover-image to two or three LSB or even more. The more substituted LSBs there are, the more obvious image distortion that is produced [25].

Accordingly, many attempts to reduce image distortion have been proposed, as in [26] [27]. Tsai et al. [28] explored the concept of the pixel relationship, which should be put into account when hiding the secret data in the image's pixels. Otherwise, the visual quality and payload will be reduced by affecting the smooth and edgy areas of the input image. Therefore, to increase the payload and visual quality, the author of [28] suggests concentrating the data hiding at the pixels in the edgy area while reducing them in the pixels of the smooth area. As a consequence, many researchers were inspired by this idea and proposed several versions for

improving Tsai's approach, such as that seen in [29] [30]. Also, the visual quality took a place of improvement as well. For example, Chang et al. [31] designed a strategy to specify a pixel adjustment for embedding data, which in turn Wu et al. [32] has improved using differencing of the pixel value. This made Tsai's approach more suitable for several applications.

On the other side, embedding encryption or randomization techniques to increase the security level in traditional steganography algorithms were not considered. This increased the challengers' chances of extracting data through their successful attacks of data-hiding algorithms.

An example of easy data extracting steganography algorithms is available in [33]: the Sequential Color Cycle (SCC) technique. The SCC has been developed to improve the secret data's payload. In SCC the colour channels (RGB) of each pixel, where the secret data will be hidden, is cycling frequently for every bit according to a specific pattern, such that one-bit LSB, two-bit LSBs, three-bit LSBs, and even up to four-bit LSBs at each RGB colour pixel depending on the size of the secret data. Therefore, if one-LSB pattern is applied, then the first bit of the secret data is stored in the LSB of the red channel, the second bit in the green channel, the third bit in the blue channel, and so on (see **Fig. 1**, Although, this technique is more secure than the original LSB, but detecting the cycling pattern will reveal the secret data.

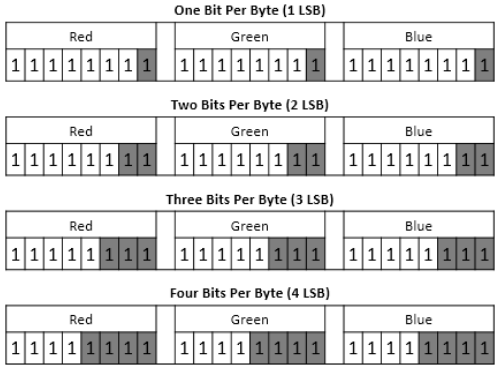


Fig. 1. In the SCC algorithm, the secret data will be hidden according to a specific pattern, such as on-bit LSB, two-bit LSB, or three-bit LSB at each RGB colour pixel.

Therefore, many attempts to improve steganography security using the randomization technique have been proposed. For example, a random selection of pixels to embed the required message (text) using the LSB substitution have been used in [34]. This method generates random numbers using a random function generator, such as $K = ((i*2) + 1) \ mod \ 11$, where i corresponds to the text-bit position, K corresponds to the pixel position, and *mod* 11 might vary depending on the image's size. This method is targeted to improve security since a password is embedded in the first column of the arranged pixels matrix.

Similarly, in [35], a sequence of random pixels is generated uniquely for a particular image using a Pixel Locator Sequence (PLS) that acts as a key during the decoding process. The pixels can be randomly distributed in PLS using the Modern Fisher-Yates Shuffle. Moreover, the location of the data (in the form of pixel numbers) is encrypted/decrypted using Advanced Encryption Standard (AES) which provides double encryption for data and its location is stored over pixels.

Also, in [29], the authors added ±1 randomly to each pixel of the input image in order to reduce the asymmetric effects in their so-called LSB matching (LSBM) method. However, this random addition depends on the data secret bits as well as the cover-image's pixels. A

revisited LSBM (LSBMR) approach suggests an independent interpretation of the image's pixels by hiding two bits in a two-pixel pair to minimise the asymmetric effects of the previous approaches based on LSB [22]. However, the continuous modifications in any of the channels would affect the other channels, which in turn, would produce low-quality stego-images.

The embedded randomization techniques into steganography algorithms are varied and extended. However, an efficient and flexible randomization technique is still required. This demand has motivated us to propose a new randomization technique that is more flexible and efficient. The main contribution of this paper is to propose a new randomization technique (i.e. *our Rand*-Stego technique) that absorbs almost all of the existing techniques into one technique.

3. Proposed Work

The proposed technique (called *Rand*-Stego) is used for a random selection of pixels (where the secret data will be hidden) without affecting the "internal" implementation of the steganography algorithm itself. Based on the literature, the majority of steganography algorithms are based on the sequential selection order of the pixels, but with different concealing techniques inside each pixel. *Rand*-Stego would improve the security level through the random selection of pixels with simple and fewer calculation processes. The *Rand-Stego* could be used with almost every existing steganography algorithm. The problem with hiding data without randomization is that, once the user hands over a stego-image, they may discover the secret data using stego-analysis techniques [1]. Our *Rand*-Stego technique could be used with any existing steganography algorithm to add a random selection of pixels. This will enhance the security level of the original steganography algorithm.

The *Rand-Stego* is based on a random key (called the pattern key) of a limited number of digits. The pattern key consists of (0s or 1s, The ones (1s) indicate the pixels' positions where the embedding (hiding part of a secret message) will occur. Meanwhile, the zeros (0s) indicate where "no hiding" data should occur. The length of the pattern key varies. It is based on your need, could be short with two or three digits or a bit more, or could be long with a large number of the bits of the secret message. The pattern key could be generated manually or by using random functions. Furthermore, as will be shown later, it is easy to either generate the pattern of a random-key or track it when extracting the secret message.

For a small key pattern example, we considered (01) to conceal secret data in an image of 2x13 pixels (see **Fig. 2** below, Where the 1-pixels (gray) indicate to the pixels' positions where the hiding process of secret data will take place (i.e. pixels positions 2, 4, 6, etc), and the 0-pixels (white) indicate to pixels' positions where no hiding data should occur (i.e. pixels positions 1, 3, 5, etc, This means exploiting 50% of the image payload, as will be discussed later.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1	0	1	0	1	0	1	0	1	0	1	0
2	1	0	1	0	1	0	1	0	1	0	1	0	1

Fig. 2. The key pattern (01) conceals secret data in 2x13-pixel image. In gray, 1-pixels indicate concealment, while 0-pixels (white) indicate non-concealment.

For an example with a longer key pattern, consider **Fig. 3** below which represents the 14x14 image pixels, and the key pattern (101011011, Here, the length of the key pattern is nine, therefore, in the first round, the first nine pixels of the image will be subject to this key pattern.

Then the key pattern will repeat itself along till the end of the image pixels. Concealing data will be at the pixels corresponding to the ones of the key pattern. The original steganography technique will be used to conceal data in the selected pixels. Therefore, the role of *Rand*-Stego will be only organizing where to-hide and where not-to-hide, regardless of the underlying steganography technique.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	0	1	0	1	1	0	1	1	1	0	1	0	1
2	1	0	1	1	1	0	1	0	1	1	0	1	1	1
3	0	1	0	1	1	0	1	1	1	0	1	0	1	1
4	0	1	1	1	0	1	0	1	1	0	1	1	1	0
5	1	0	1	1	0	1	1	1	0	1	0	1	1	0
6	1	1	1	0	1	0	1	1	0	1	1	1	0	1
7	0	1	1	0	1	1	1	0	1	0	1	1	0	1
8	1	1	0	1	0	1	1	0	1	1	1	0	1	0
9	1	1	0	1	1	1	0	1	0	1	1	0	1	1
10	1	0	1	0	1	1	0	1	1	1	0	1	0	1
11	1	0	1	1	1	0	1	0	1	1	0	1	1	1
12	0	1	0	1	1	0	1	1	1	0	1	0	1	1
13	0	1	1	1	0	1	0	1	1	0	1	1	1	0
14	1	0	1	1	0	1	1	1	0	1	0	1	1	0

Fig. 3. This represents an image of 14x14 pixels. Pixels with 1 (gray) indicate the hiding of secret data, whereas 0 (white) indicates no hiding data in these pixels. The thick lines indicate the start/end of the random key pattern.

Accordingly, the pattern (1111) would mean to hide the secret message's bits in every pixel of the cover image, having the same effect as the sequential order pixel selection techniques. On the other side, the pattern (0000) means no pixels were selected for the concealing process, thus there would be no changes to the cover-image at all. Obviously, pattern (0000) has the best results of MSE (Mean Square Error) and PSNR (peak signal noise ratio) with zero payload capacity but unfortunately, no data will be concealed. Therefore, calculating the payload of the cover-image and secret message for each pattern is very important for the success of the concealing process in *Rand*-Stego. **Table 1**, provides more *Rand*-Stego pattern examples.

Table 1. Sample of key pattern and the affected and not affected pixel's positions (index)

Key Pattern	No-Hiding data at Pixels Positions	Hiding data at Pixels Positions
01	1, 3, 5, …	2, 4, 6, …
10	2, 4, 6, …	1, 3, 5, …
0011	1, 2, 5, 6, 9, 10, 13, 14, …	3, 4, 7, 8, 11, 12, 15, 16, …
0	All places	NO Places
0001	1, 2, 3, 5, 6, 7, 9, 10, 11, …	4, 8, 12, 16, 20, …

3.1 *Rand*-Stego Random Key Function

As mentioned above, the *Rand*-Stego random key is a pattern that consists of binary digits: 0s and 1s, where 1 represents the situation where embedding (hiding) will occur, and 0 represents the situation where no hiding of the secret message will occur. The random key—or in short, "the key"—should be generated according to the secret message length as well as the payload of the selected cover-image. The key may be stored in a text file during the embedding process and could be sent separately to the recipient after encrypted for use in the extraction process. Thus, the key is a must in order to specify the pixels' position where the secret message is hidden for extracted purposes.

The following pseudocode is only for demonstrating the use of the *Rand*-Stego key function in the LSB technique. This function could be embedded in any stego-algorithm in the data hiding/extracting process. The main idea of the code is to read the key bit-by-bit (i.e., number by number) from position one till the end of the key. Then, repeat the same process till hiding/extracting all of the secret message bits in the cover-image.

Pseudocode of using *Rand*-Stego in embedded process:

```
1_ keyT = fileread ('NewKey.txt');          % reading key from a file
2_ key = charbits2uint8bits(keyT);          % converting key
3_ firstlsb = reshape(firstlsb,1,m*n);      % conversion from 2D to 1D.
4_ KeyPos  = 1;
5_ KeyPosL = length(key);                   % calculating the key length
6_ NoHide = 0;
7_ Hide = 0;
8_ for k = 1 to length(secretMsg)
9_    if (KeyPos > length(key))
10_     KeyPos = 1;
11_   end if
12_   if (key(KeyPos) == 0 )
13_     NoHide = NoHide + 1;                % not to hide data
14_   else
15_     Hide = Hide + 1;                    % hiding data
16_     firstlsb(k) = secretMsg(k);         % replacing LSB with data
17_   end if
18_   KeyPos = KeyPos + 1;                  % moving to next position
19_ End for
```

The following will explain the above code:

Lines 1 to 7 are used to read the key, convert the key to a one-dimensional matrix, and initialise the Hide and NoHide counters, respectfully.

Lines 8, 9, and 10 start the loop over the length of the secret message. The current position of the key will be re-set to position one and incremented by one (line 18), then re-set again to one when the current position of the key goes beyond its length.

Lines 12, 13, and 14 will check if the value in the key is zero, where no hiding data should occur, and therefore, it will do nothing except increment the NoHide counter by one. Lines 15 to 19 of the pseudocode will call/invoke the original steganography algorithm.

Furthermore, the above code can be used in the Embedding and Extracting process. The only difference in extracting the hidden data is to call the extract procedure of the original steganography algorithm instead of the hiding procedure according to the key.

4. *Rand*-Stego Implementation.

Randomization techniques including our proposed technique have many usages besides steganography, for example, in science, art, statistics, cryptography, gaming, gambling, and other engineering fields. For instance, it helps video games such as video poker [36]. In engineering, randomization techniques are also commonly used in computer simulations (such as real-life phenomena [37]).

However, in cybersecurity, the success of many cyberattacks relies on the hacker's ability to know or guess the position of the encrypted/concealed data. With *Rand*-Stego, we are able to give unpredictable (i.e., increasing unpredictability) positions of the concealed data, increasing the difficulty of attack.

In order to show the applicability and advantages of implementing *Rand*-Stego over steganography techniques, the current techniques will be categorized into two types, Type-1 and Type-2. Type-1 will represent the currently existing steganography techniques which already have an embedded randomization technique, whereas Type-2 are the ones that do not use any randomization technique. Next, some examples will be provided to show how *Rand*-Stego can be implemented for both types.

4.1 *Rand*-Stego Implementation on Type-1

This section will briefly show how implementing *Rand*-Stego would enhance the security level of some existing steganography techniques in a simple way. Below are two examples of using *Rand*-Stego for Type-1, the steganography which includes another randomization method. However, *Rand*-Stego could cover nearly all existing steganographic methods.

For the first example of implementing *Rand*-Stego let us consider the technique of [34], which is mentioned above in section 2. As described before, this technique is based on the LSB substitution method with one difference: the selection of the pixels' positions for concealing secret data that is randomly generated. The generating of these random pixels' numbers are based on a Random Function Generator (RFG) (i.e. K= ((i*2) +1) *mod* 11, Here, two options are available to embed *Rand*-Stego into this technique. The first option is to completely replace the RFG with *Rand*-Stego thereby providing more flexibility and simplicity as previously described. The second option is to map the RFG-generated random pixels into the *Rand*-Stego key pattern. For example, if RFG generates the following random numbers (5, 9, 4, 10, and 7) which indicate the position of the pixels where hiding data will occur. Here, the corresponding key pattern of the *Rand*-Stego will be (0001101011), which matches the RFG random key (5, 9, 4, 10, and 7), see **Fig. 4**. This mapping would ease the embedding/extracting of the secret messages at [34].

1	2	3	4	5	6	7	8	9	10	⋯
0	0	0	1	1	0	1	0	1	1	⋯

Fig. 4. An example of mapping the RFG-generated random numbers (e.g. 5, 9, 4, 10, and 7) to the *Rand*-Stego key pattern (0001101011).

For the second example of implementing *Rand*-Stego, consider a recent algorithm described in [38]. In their article, Kordova and Zhelezov presented a brand-new steganography approach for hiding confidential data in colored images by combining a steganography technique with cryptography in two stages. In the first stage, the secret message has to be encrypted using the XOR operation. In the second stage, the encrypted message's bits are concealed by the pixels' bits of the cover image. Here, the selection of pixels is made randomly, based on a pseudo-random generator (PRG, The proposed *Rand*-Stego does not change the internal design of any steganography techniques, such as how the secret bits will be hidden or which channels/bits of the pixels to use in the cover image. Instead, our technique would provide a simple method of picking pixels randomly to conceal sensitive data or could serve as a multi-purpose randomization technique.

The *Rand*-Stego contribution to the [38] technique does not take place at the encryption stage, but at the second stage, where it might enhance or replace PRG's method of selecting pixels. According to [39], the PRG technique suffers from a range of issues, such as the correlation of successive values and improper implementation. Therefore, replacing their PRG method with our *Rand*-Stego would be the most appropriate choice, since *Rand*-Stego is simple to implement and tracks the selected pixels when the extraction process proceeds. Or, simply, the generated random numbers from the PRG method can be mapped onto the *Rand*-Stego key.

4.2 *Rand*-Stego Implementation on Type-2

This section describes how the *Rand*-Stego technique could be applied to existing steganography techniques of Type-2 (which do not incorporate randomization techniques) to demonstrate its efficacy in improving security.

Now, let us consider the findings in [33], which is mentioned in the related work section above. The security level of the Sequential Color Cycle (SCC) technique, is considered a shortcoming because it uses a sequential order of pixels when hiding the secret data. This obviously increases the attacker's chances to extract these secret data.

If we embedded *Rand*-Stego into SCC, it would elevate the SCC security level in a simple way. Regardless of the SCC used pattern (one LSB, two LSBs, or three LSBs), let us assume that the *Rand*-Stego key pattern (101) has been applied to SCC. Here, the selected pixels for concealing data using SCC are not in sequential order any more. In another word, the SCC technique will not be applied on each pixel, sequentially, as it is done before, rather it will be applied based on the *Rand*-Stego key pattern only.

As '1' comes at the first position of the key pattern, then the first bit of the secret message will hide at the first pixel of the cover-image while '0', as the second position of the pattern, means no data should be hidden in the second pixel. The third pixel will hide data since '1' is in the third position of the key, and so on as the pattern 101 continued to apply along the secret message's length (see **Fig. 5**, This will make detecting/revealing the secret data harder for the attacker.

In **Fig. 5**, a description of the *Rand*-Stego key pattern (101) applied to SCC. Regardless of the SCC-used pattern, pixel 1 (gray) will contain some bits of the hidden data, pixel 2 (white) will not have any hidden data, and pixel 3 will hide some secret bits. Then, pattern (101) will be applied again and again starting from pixels 4, 7, 10, and so on as long as the secret message is not concealed completely.

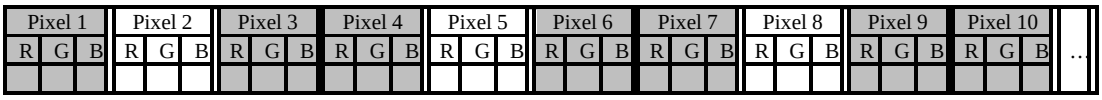


Fig. 5. Applying *Rand*-Stego using pattern (101) in SCC

In the second example of applying *Rand*-Stego, we have applied our technique on [40]: Highly randomized image steganography using secret keys. In [40] two secret keys to randomise the hiding process of data bits. This approach uses the red, green, and blue channels of each pixel. It performs some calculations to hide the secret data bits in random positions at each channel. In their approach, although the mechanism of hiding the data bits seems to provide a good level of security, their selection of pixels, where to conceal data, is still in sequential order as well. Thus, practicing the *Rand*-Stego would elevate their security level. In this practice, the selected pixels will be random based on the key pattern of the *Rand*-Stego rather than in sequential order. Technically, to implement *Rand*-Stego in this approach, all you need is to change the invoke statement of *Rand*-Stego Random Key Function in Section 3.1 line 16 to the name of this approach.

The final example of implementing the *Rand*-Stego technique will be within the algorithm of Muhammed et al.'s [13]. This algorithm has been selected since it is one of the most notable studies in image steganography [41].

This algorithm will be called *Muh-algo* now and onward. Muhammed et al. [13] proposed a secure framework to tackle the problem of securing sensitive content transmission. Their algorithm uses an imperceptible adaptive LSB substitution framework based on uncorrelated colour space (UCS) which proved its efficiency in processing, de-correlation, the better quality of Stego-Image, and suitability for steganography through many experiments. Furthermore, *Muh-algo* gives a good balance between image quality and security.

As we will show later, the security and quality of *Muh-algo* will be enhanced by implementing *Rand*-Stego. Therefore, in the following subsections, *Muh-algo* will be described first followed by the positive effect of applying (implementing) the *Rand*-Stego technique.

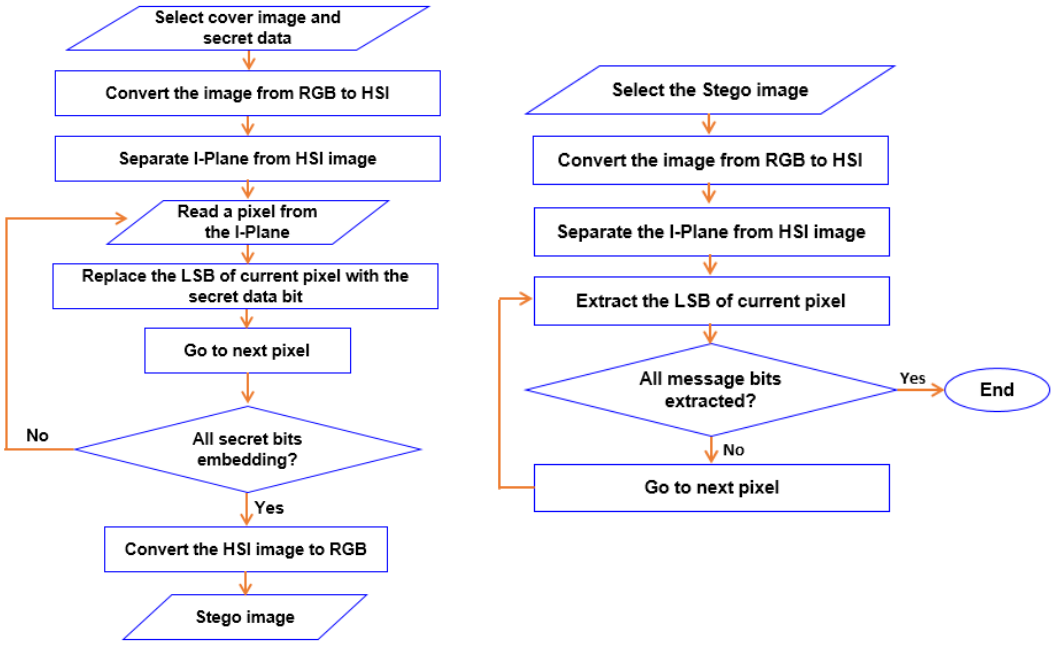
Muh-algo (Muhammad Techniques)

Muh-algo is based on converting the RGB colour images into an HSI (Hue, Saturation, and Intensity) colour model, where the concealing process of the secret message will take place in the LSB of its I-Plane. Then the HSI image is converted back, again, into the RGB colour model as a final stego-image.

It is worth mentioning here that the HSI colour model represents colour as three components: hue (H), saturation (S), intensity (I, It is very important and attractive for image processing applications because it represents the colours in the way that the human eye senses them. Whereas, the RGB colour model represents the primary colours (red, green, and blue) that are added together in various ways to reproduce a wide array of colours.

The authors of *Muh-algo* have selected the HSI colour space, rather than the RGB one, for hiding secret messages for several reasons, including the fact that the image processing of the RGB colour scheme is relatively difficult, time-consuming, and ineffective when handling real photos. Whereas, HSI can be used simply for processing, programming, and manipulating [13].

The flowcharts of the embedding (hiding) and extraction processes of *Muh-algo* can be seen in **Fig. 6**, a and b, respectively, below.

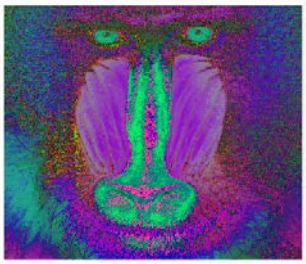


(a) (b)

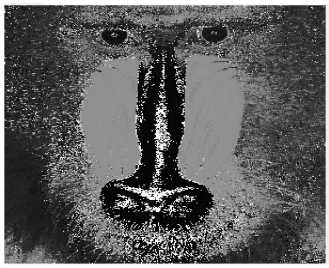
Fig. 6. (a) embedding and (b) extraction processes flowcharts of *Muh-algo*.

Accordingly, in **Fig. 7**, a sequence of images *a* – *h* are shown below to demonstrate concealing (embedding) secret data using the *Muh-algo* technique.

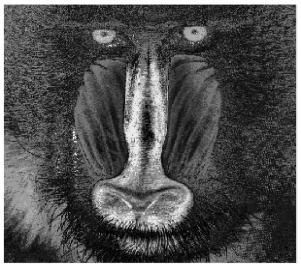
(a) RGB cover-image



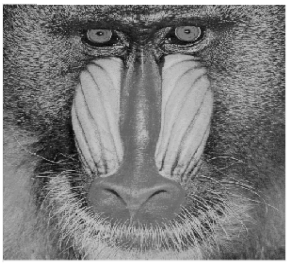
(b) Convert RGB into HSI Image



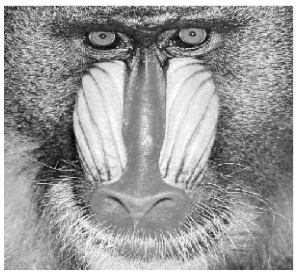
(c) The H Plane

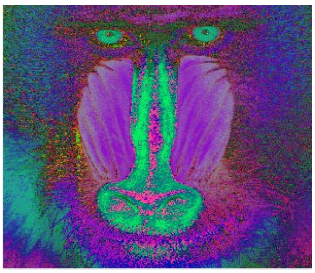


(d) The S Plane



(e) The I Plane


(f) The I Plane after embedding


(g) Stego image in HSI format

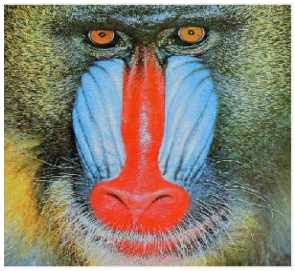

(h) RGB stego image

Fig. 7. a sequence of images *a* – *h* that demonstrates the concealing secret data using the *Muh-algo*

Implementing *Rand*-Stego in *Muh-algo*

This section will apply *Rand*-Stego to *Muh-algo*. Moreover, the *Rand*-Stego algorithm as a Pseudocode of producing a random key of selected pixels (where concealing the secret message will take place) will be provided. It is worth mentioning here that this is the key point that distinguishes *Rand*-Stego from *Muh-algo* (i.e., the *Rand*-Stego technique hides the secret data at randomly selected pixels whereas *Muh-algo* hides data in sequential order of pixels, The performance of *Rand*-Stego will be evaluated through payload capacity, PSNR, and MSE on several cover-images.

The main steps of implementing *Rand*-Stego into *Muh-algo* are:

1. *Selecting RGB cover-image and the secret message.*
2. *Converting the RGB cover-image into a HSI colour model.*
3. *Separating the I-Plane from the HSI cover-image.*
4. *Generating a random key to hide the secret message.*
5. *Transforming the secret message into the 1-D array of bits.*
6. *Replacing the LSB of the I-Plane of the selected (based on the random key) pixel by the bits of the secret message. Repeat this process until all bits of the secret message are replaced.*
7. *Converting the HSI stego-image into RGB stego-image using different formulas. The RGB stego-image can be evaluated by MSE and PSNR.*

The main difference is in step 6 which will be detailed in the following subsection.

As mentioned above, the *Rand*-Stego technique can be used in any steganography technique. Therefore, the main steps of Embedding and Extracting *Rand*-Stego procedures on *Muh-algo* will be described below. The main changes of both procedures are underlined in steps 6 5, respectively.

- **Embedding *Rand*-Stego into *Muh-algo* concealing steps**

1. Select the RGB cover-image.
2. Convert the RGB image into the HSI colour model.
3. Generate the Random Key position.
4. Convert the secret message into 1-D array of bits.

5. Select a pixel from the I-Plane.
6. Replace random bit with secret bit according to the key.
7. Repeat step 6 until all of the secret bits are encoded in the I-Plane pixels.
8. Convert the HSI image model into the RGB colour space.
9. Write the stego-image.

- **Embedding *Rand*-Stego into *Muh-algo* extraction steps**

1. Select the RGB stego-image.
2. Convert the RGB image into the HSI colour model.
3. Separate the I-plane from the HSI.
4. Extract the LSB of the current pixel from the I-Plane of the HSI stego-image.
5. Repeat step 4 until all of the secret bits are extracted according to the random key positions.
6. Converting the secret bits into the secret message.

- **Preliminaries**

A set of 25 cover-images, each in jpg format and 481x321 pixels, were collected from different sources (e.g. Berkeley segmentation dataset) as shown in the Appendix, **Table A**. The MATLAB R2015b was used for concealing and extracting the secret message. In the concealing process, the jpg image format will be used, which will generate a stego-image in tiff format for the extraction process. The mean square error (MSE) and the peak signal noise ratio (PSNR) tests are used to find the differences (changes) of the cover-image before and after implementing the *Rand*-Stego. Essentially, the MSE represents the cumulative squared error between the generated and original images, and is considered a controlling and quality accepted measure, whereas PSNR represents a measure of the peak error which is a reliable quality metric [42]. The PSNR in the logarithmic function over the MSE using some of metrics of the image.

The *Muh-algo* before and after implementing *Rand*-Stego will be subject to the MSE and PSNR tests. In other words, *Muh-algo*—which conceals secret messages in sequential order of LSB—will be compared with the enhanced *Muh-algo* that embeds the *Rand*-Stego and uses random positions of LSB (pixels, Moreover, in order to make the comparison more scientific, different lengths of the secret messages and different patterns will be involved in these tests. All results will be compared and illustrated.

Several key patterns have been used to study the effect of *Rand*-Stego on *Muh-algo*. For example, the key patterns below specify the positions (pixels) where to hide and not to hide (NoHide) secret data should take place.

Pattern#1 below has a key of 12-bits length (i.e., key (1-12)), and the positions of NoHide are 3, 6, and 12, respectively. The internal representation of this key is "110110111110" where the zeros indicate the NoHide positions (pixels, This way of pattern expression was used for the sake of variety and ease of explanations. All of the used key patterns are provided below:

Group 1: Long secret message.

- Pattern#1: key (1-12) with three "NoHide" positions: 3 – 6 – 12
  - Key pattern : 110110111110
- Pattern#2: key (1-50) with six "NoHide" positions: 1 – 10 – 17 – 25 – 33 – 50
  - Key pattern : 01111111101111110111111101111111011111111111111110
- Pattern#3: key (1-200) with nine "NoHide" positions: 5 – 13 – 21 – 50 – 54 – 76 – 77 – 132 – 200

- Pattern#4: key (1-500) with fifty "NoHide" positions: 1 – 10 – 20 – 30 – 40 – 50 – 60 – 70 – 80 – 90 – 100 – 110 – 120 – 130 – 140 – 150 – 160 – 170 – 180 – 190 – 200 – 210 – 220 – 230 – 240 – 250 – 260 – 270 – 280 – 290 – 300 – 310 – 320 – 340 – 350 – 360 – 370 – 380 – 390 – 400 – 410 – 420 – 430 – 440 – 450 – 460 – 470 – 480 – 490 – 500

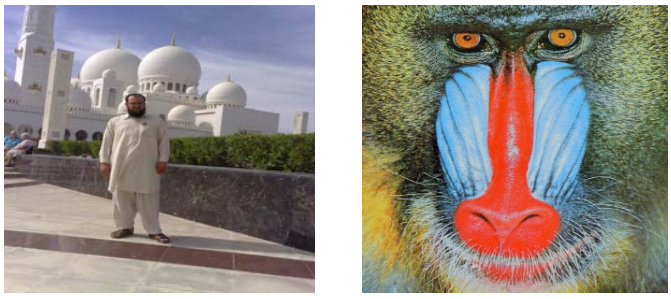
Group 2: Short secret message
- Pattern#5: key (1-12) with three "NoHide" positions as Pattern#1 in group 1
- Pattern#6: key (1-500) with fifty "NoHide" positions as Pattern#4 in group 1

The key patterns above have been used for concealing secret data in 25 images and thus the MSE and PSNR have been calculated. Then, the average of the MSE and PSNR of 25 images were computed.

• Results and Discussion

In this section, the results of implementing *Rand*-Stego in *Muh-algo* using a set of cover-images (standard colour images) are presented and discussed. Moreover, the efficiency and performance of *Rand*-Stego will be measured in terms of MSE and PSNR. The results, as we will see, prove the enhancement of *Muh-algo* without compromising the image quality and with elevating the level of security via better MSE and PSNR results.

The *Rand*-Stego key pattern has wide-ranging flexibility when embedding into the LSB technique such that it could be embedded without having an effect on the used algorithm. For example, in **Fig. 8** below, two images were used to hide secret data using *Muh-algo* with the *Rand*-Stego key patterns (1, 11, and 111, Obviously, these keys are supposed to hide data in sequential order of pixels since no zeros are included in the keys, and therefore, the results of MSE and PSNR for *Muh-algo* before and after implementing *Rand*-Stego are the same as shown in **Table 2** below.



Mandril colour Scene

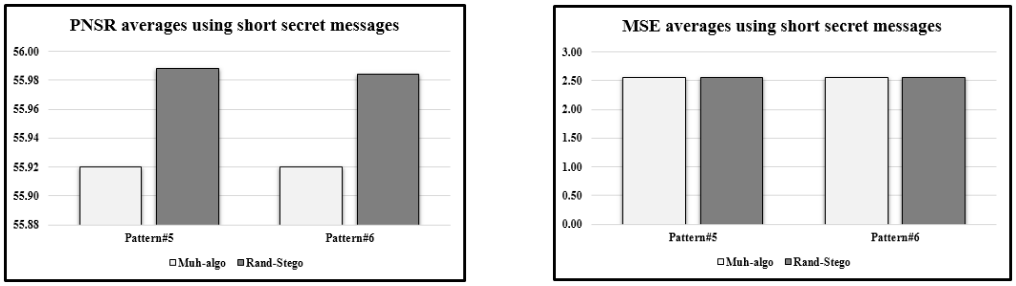
Fig. 8. Images were used to hide secret data by applying the key patterns (1, 11, and 111) of *Rand*-Stego with *Much-algo*.

Table 2. The MSE and PSNR test results before and after applying *Rand*-Stego to *Muh*-algo.

Image	**Pattern (1, 11, and 111)**			
	Muh-algo		***Muh-algo* With *Rand*-Stego**	
	MSE	**PSNR**	**MSE**	**PSNR**
Scene	0.001	77.1	0.001	77.1
Mandril colour	0.000	89.9	0.000	89.9

The results of MSE and PSNR for *Muh-algo* with and without *Rand*-Stego implantation will be discussed and compared below. These results are based on different key patterns and lengths of secret messages.

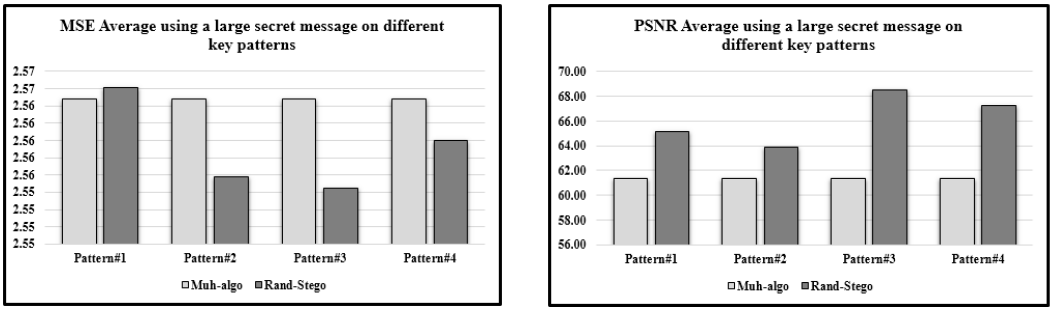
The averages of the results of both MSE and PSNR for the 25 cover-images using short secret messages with patterns #5 and #6 in group 2 above indicate a minimal level of differences in *Muh-algo* with and without applying *Rand*-Sego, as can be seen in **Fig. 9** (see **Table B** in the Appendix for more detail).



(a) (b)

Fig. 9. (a) PSNR and (b) MSE test average charts using two key patterns of *Rand*-Stego #5 & #6 with short secret message compared to the original *Muh*-algo results.

Accordingly, the MSE results based on pattern#3 and pattern #4 in group 1 using long secret messages show a near similarity in both techniques. But, the PSNR results, on the same conditions, show a significant improvement for the *Muh-algo* technique when embedding *Rand*-Stego. See **Fig. 10** and **Table 3** (see also **Table C** in the Appendix).



(a) (b)

Fig. 10. MSE (a) and PSNR (b) test averages using long secret messages with different key patterns.

Table 3. MSE Averages for using different patterns with long secret messages

Key Patterns Group 1	**MSE - Average**	
	Muh-algo	***Muh-algo* with *Rand*-Stego**
Pattern#1 (1-12)	2.564	2.566
Pattern#2 (1-50)	2.564	2.555
Pattern#3 (1-200)	2.564	2.554
Pattern#4 (1-500)	2.564	2.559
Average	2.564	2.558

Detailed results can be seen in the Appendix tables D and C.

In contrast to pattern#1, the results of MSE for pattern#2 indicate a slight improvement when implementing *Rand*-Stego on *Muh-algo*, specifically going from 2.564 to 2.555. However, the PSNR results again showed a significant improvement for *Rand*-Stego implementation, specifically 63.89 to 61.39 in pattern#1 and 65.16 to 61.39 in pattern#2. See **Fig. 10** and **Table 3** (also **Table D** in the Appendix).

Obviously, as can be seen in **Fig. 10**, the *Rand*-Stego technique with a longer key pattern can achieve better MSE and PSNR results. Pattern#3 achieves the best test results (MSE =2.554 and PSNR = 68.4) while, in contrast, pattern#1 achieves the worst in MSE test (MSE =2.566) and almost the worst in PSNR test (PSNR = 65.16).

Fig. 11 shows an overall MSE average for using long secret messages. The lower average means a lower error rate. Therefore, applying *Rand*-Stego on *Muh-algo* achieves better results than *Muh-algo* itself.

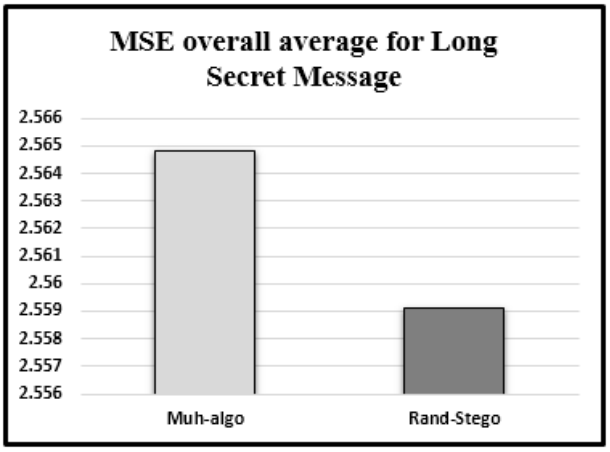


Fig. 11. MSE overall average of different patterns of long secret message with *Muh-algo* average.

On the other side, **Table 4** and **Fig. 12** demonstrate the overall PSNR results' average for long-secret messages. However, as is well-known, a higher PSNR means better results. Therefore, applying *Rand*-Stego into *Muh-algo* achieves better results than the original *Muh-algo* alone.

Table 4. PSNR overall average for different patterns with long secret message

Key Patterns Group 1	PSNR - Average	
	Muh-algo	***Muh-algo* with *Rand*-Stego**
Pattern#1 (1-12)	61.39	65.16
Pattern#2 (1-50)	61.39	63.89
Pattern#3 (1-200)	61.39	68.48
Pattern#4 (1-500)	61.39	67.29
Average	61.39	66.2

Detailed results can be seen in Appendix tables D and C.

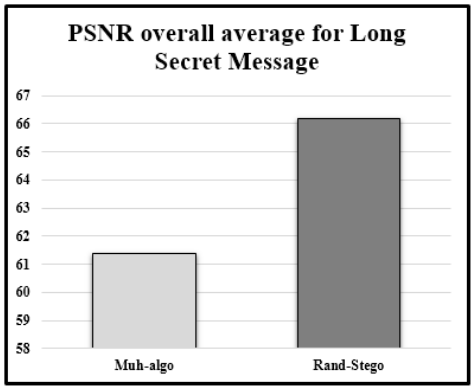


Fig. 12. PSNR overall average of different patterns of long secret message with *Muh-algo* average.

5. Key Pattern and Cover-Image Payload

Calculating the length of the secret message with the payload of the cover-image is essential in every steganography technique. Hiding a long secret message in a limited cover-image payload would certainly fail the steganography process since there is not enough space to hide the complete secret message.

The *Rand*-Stego key patterns have a wide range of flexibility such that they can be adjusted to fit the secret messages' length as well as the cover-images' payload. But looking closely into the *Rand*-Stego key pattern will lead directly to a clear inference that the number of zeros in the key pattern would affect the cover-image payload. For example, the key pattern (01) would waste 50% of the cover-image payload while the key (1011) would waste (25%) of the payload. This occurs even though the key (1111) is 0%, meaning it would not cost any extra waste of the cover-image payload, and this pattern implements no randomization at all. However, minimal zeros in the key pattern are better for saving the cover-image payload.

In **Table 5**, the key patterns with wasted payload are shown. Increasing or decreasing the number of zeros in the key pattern would greatly affect the quality of the cover-image as well as its payload capacity. However, based on the previous experiment with *Muh-algo*, the key pattern (1-200) yield the best results. Therefore, the fewer zeros at the key pattern, the lower MSE, and higher PSNR results.

Table 5. Number of zeros in the key patterns and the percentage of wasted payload

Key patterns with number of zeros	Percentage of wasted payload
Pattern#1 (1-12) with 3 NoHide positions	25%
Pattern#2 (1-50) with 6 NoHide positions	12%
Pattern#3 (1-200) with 9 NoHide positions	4.5%
Pattern#4 (1-500) with 50 NoHide positions	10%

As a consequence, a key pattern with no zeros (0%) would give exactly the same result as the original technique (without applying *Rand*-Stego, Whereas, the key patterns with 50% zeros would consume double the payload of the cover-image than what the original technique would consume. Therefore, calculating the number of zeros at the key pattern and the cover-image payload, in addition to the secret message's length, are essential for the success of the *Rand*-Stego and all other steganography methods.

6. Flexibility *of Rand*-Stego

As demonstrated in Section 4, *Rand*-Stego can easily be applied to other steganography techniques, especially to those using sequential order of pixels or those with mathematical complications.

However, based on the secret message length as well as the cover-image payload, the *Rand*-Stego gives a flexible selection of key pattern that fits with the available requirements. For instance, the key pattern with no zeros (0%) means disabling the effect of *Rand*-Stego and keeping the original steganography techniques as it is. By increasing the number of zeros in the key pattern (while taking into account the secret message length and cover-image payload), you increase the randomization level thus the security level. This flexibility can easily be gained by applying the *Rand*-Stego Random Key Function in Section 3.1.

The majority of steganography techniques are applied sequentially to pixels. Therefore, using *Rand*-Stego will definitely reduce the ability of hackers to know or guess the position of the encrypted/concealed data as it is done on a random basis though. The simplicity of the proposed technique comes from its ease of generation and application. However, its efficiency comes from the ability to control the number of selected pixels as well as enhance the security level of other techniques.

Using *Rand*-Stego with *Muh-algo* yields effective results regarding MSE and PSNR. However, our *Rand*-Stego is more generalized than our previous work [43] and can be applied to most steganography techniques. In the [43] article, the integrity and security of Bhallamudi's work [44] have been enhanced by adding a randomization technique to the initial work. The Bhallamudi [44] technique has been improved by embedding the text bits randomly, depending on the value of different bytes in the original image. It hides data randomly in the image by checking the first byte of the original image and hiding data based on the values of these bytes. The article [43] proved that effectiveness was enhanced. The PSNR has increased on average by 9%. Finally, the method used in [43] is an implementation of a special case of our newly developed *Rand*-Stego technique.

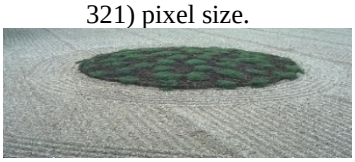
7. Conclusion

Enhancing security for digital data transmitting is of great concern nowadays. Many techniques have been implemented to protect digital data against unauthorized access. Steganography technology proves that it is an effective and powerful technology in securing transmitted digital data compared to others. Several randomization methods were used in steganography with many limitations, such as simplicity, flexibility, and efficiency. In this research, a new randomization technique (called *Rand*-Stego) is presented and proven to have characteristics of simplicity, flexibility, and efficiency. A demonstration implementing *Rand*-Stego to some existent algorithms, is provided. The flexibility of *Rand*-Stego can be seen when selecting the key pattern that is used for the randomly selecting pixels (where the secret message should be hidden, While its simplicity was observed when implemented (embed) it on other steganography techniques and also, through the easiness when calculating its payload, and finally, the results of MSE and PSNR showed its efficiency. As a consequence, we conclude that *Rand*-Stego can be implemented over any steganography technique to improve the security level with minimal effort.

Appendix

Table A. 25 Sample images from Berkeley Segmentation Dataset, each of type jpg and (481 X 321) pixel size.


1


10


18


2


11


19


3


12


20


4


13


21


5


14


22


6


15


23


7


16


24


8


9


17


25

Table B. MSE and PSNR for Random Key position patterns (1-12) and (1-500) with short secret message.

Image	(1-12) short secret message				(1-500) Short secret message			
	MSE		PSNR		MSE		PSNR	
	M*	R*	M	R	M	R	M	R
1	42.4	42.4	31.7	31.7	42.4	42.4	31.7	31.7
2	1.1	1.1	47.7	47.7	1.1	1.1	47.7	47.7
3	0	0	91.2	91.2	0	0	91.2	91.2
4	0.115	0.115	57.4	57.4	0.115	0.115	57.4	57.4
5	0.002	0.001	74.6	76.3	0.002	0.0015	74.6	76.3
6	2.8	2.8	43.5	43.5	2.8	2.8	43.5	43.5
7	0	0	83.8	83.8	0	0	83.8	83.8
8	0	0	98.2	98.2	0	0	98.2	98.2
9	0	0	0	0	0	0	0	0
10	0.029	0.029	63.3	63.3	0.029	0.029	63.3	63.3
11	0.002077	0.002077	74.9	74.9	0.002	0.002	74.9	74.9
12	0.130356	0.132025	56.9	56.9	0.13	0.128	56.9	56.9
13	0	0	0	0	0	0	0	0
14	0	0	84.1	84.1	0	0	84.1	84.1
15	0	0	87	87	0	0	87	87
16	0.033	0.033	62.9	62.9	0.033	0.033	62.9	62.8
17	0.32	0.32	53	53	0.32	0.32	53	53
18	0	0	0	0	0	0	0	0
19	0.001	0.001	74.9	74.9	0.001	0.001	74.9	74.9
20	0	0	80.4	80.4	0	0	80.4	80.4
21	0.044	0.044	61.4	61.4	0.044	0.044	61.4	61.4
22	0.757	0.757	49.3	49.3	0.757	0.757	49.3	49.3
23	0	0	0	0	0	0	0	0
24	0	0	85.8	85.8	0	0	85.8	85.8
25	16.3	16.3	36	36	16.3	16.3	36	36
Average	2.561	2.561	55.92	55,98	2.561	2.561	55.92	55.98

*M: *Muh*-algo
*R: *Rand-Stego*

Table C. MSE and PSNR for the patterns Key (1-500) and (1-200) with Long secret message

Image	(1-500) long secret message				(1-200) long secret message			
	MSE		PSNR		MSE		PSNR	
	M*	R*	M	R	M	R	M	R
1	42.3	42.4	31.7	31.7	42.3	42.3	31.7	31.7
2	1.1	1.1	47.6	47.6	1.1	1.1	47.6	47.6
3	0.023	0	64.4	91.2	0.023	0	64.4	91.2
4	0.119	0.117	57.2	57.3	0.119	0.111	57.2	57.5
5	0.052	0.017	60.8	65.6	0.052	0.011	60.8	67.4
6	2.7	2.8	43.6	43.5	2.7	2.8	43.6	43.5
7	0	0	82.4	84.2	0	0	82.4	84.8
8	0.041	0.003	61.9	72.1	0.041	0.002	61.9	75
9	0.008	0	68.9	89.7	0.008	0	68.9	95.1
10	0.036	0.03	62.4	63.2	0.036	0.03	62.4	63.2
11	0.015	0.003	66.2	72.1	0.015	0.003	66.2	72.7
12	0.209	0.155	54.9	56.2	0.209	0.141	54.9	56.6
13	0.046	0.015	61.4	66.2	0.046	0.007	61.4	69.4
14	0.004	0	71.5	80.6	0.004	0	71.5	84.1
15	0	0	79.3	87.4	0	0	79.3	87
16	0.044	0.039	61.6	62.2	0.044	0.033	61.6	62.9
17	0.323	0.316	52.9	53	0.323	0.322	52.9	52.9
18	0.006	0	69.6	87.7	0.006	0	69.6	93.4
19	0.002	0.002	74.6	74.5	0.002	0.001	74.6	75.2
20	0.009	0	67.8	80.4	0.009	0	67.8	80.2
21	0.066	0.044	59.7	61.4	0.066	0.044	59.7	61.4
22	0.814	0.758	48.9	49.2	0.814	0.756	48.9	49.3
23	0.001	0	76.5	91	0.001	0	76.5	95
24	0.003	0	73	78.4	0.003	0	73	79.1
25	16.2	16.2	36	36	16.2	16.2	36	36
Average	2.564	2,559	61.39	67.29	2.564	2,554	61.39	68,48

*M: *Muh*-algo
*R: *Rand-Stego*

Table D. MSE and PSNR for Random Key position patterns (1-50) and (1-12) with long secret message.

Image	(1-50) with long secret message				(1-12) with long secret message			
	MSE		PSNR		MSE		PSNR	
	M*	R*	M	R	M	R	M	R
1	42.3	42.3	31.7	31.7	42.3	42.5	31.7	31.7
2	1.1	1.1	47.6	47.6	1.1	1	47.6	47.7
3	0.023	0	64.4	91.2	0.023	0	64.4	80
4	0.119	0.115	57.2	57.4	0.119	0.114	57.2	57.4
5	0.052	0.015	60.8	66.1	0.052	0.023	60.8	64.4
6	2.7	2.8	43.6	43.5	2.7	2.8	43.6	43.5
7	0	0	82.4	83.2	0	0	82.4	81.9
8	0.041	0.004	61.9	71.4	0.041	0.009	61.9	68.5
9	0.008	0	68.9	91.1	0.008	0	68.9	80.6
10	0.036	0.03	62.4	63.2	0.036	0.03	62.4	63.1
11	0.015	0.003	66.2	72.8	0.015	0.007	66.2	69.2
12	0.209	0.151	54.9	56.3	0.209	0.171	54.9	55.8
13	0.046	0.014	61.4	66.5	0.046	0.023	61.4	64.4
14	0.004	0	71.5	82.8	0.004	0	71.5	79.2
15	0	0	79.3	87	0	0	79.3	87
16	0.044	0.033	61.6	62.8	0.044	0.042	61.6	61.8
17	0.323	0.32	52.9	53	0.323	0.321	52.9	52.9
18	0.006	0	69.6	88.6	0.006	0	69.6	82
19	0.002	0.001	74.6	75.3	0.002	0.001	74.6	75.2
20	0.009	0	67.8	80.4	0.009	0	67.8	78.1
21	0.066	0.044	59.7	61.4	0.066	0.047	59.7	61.1
22	0.814	0.766	48.9	49.2	0.814	0.764	48.9	49.2
23	0.001	0	76.5	0	0.001	0	76.5	83.3
24	0.003	0	73	78.8	0.003	0.001	73	75.1
25	16.2	16.2	36	36	16.2	16.3	36	35.9
Average	2.564	2.555	61.39	63,89	2.564	2.566	61.39	65.16

*M: *Muh*-algo
*R: *Rand-Stego*

References

[1] J. V. Anand, & G. D. Dharaneetharan, "New approach in steganography by integrating different LSB algorithms and applying randomization concept to enhance security," in *Proc. of the 2011 International Conference on Communication, Computing & Security*, pp. 474-476, Feb. 2011. Article (CrossRef Link)

[2] M. Kharrazi, H. T. Sencar, & N. Memon, "Improving steganalysis by fusion techniques: A case study with image steganography," *Proc. SPIE 6072, Security, Steganography, and Watermarking of Multimedia Contents VIII*, pp. 123-137, 2006. Article (CrossRef Link)

[3] http://www.owasp.org/index.php/Top_10_2010-A1-Injection, retrieve on 13/1/2010

[4] S. W. Boyd, & A. D. Keromytis, "SQLrand: Preventing SQL injection attacks," in *Proc. of International conference on applied cryptography and network security*, pp. 292-302, Springer, Berlin, Heidelberg, June. 2004.

[5] R. L. Rivest, & A. T. Sherman, "Randomized encryption techniques," *Advances in Cryptology*, Springer, Boston, MA, pp. 145-163, 1983. Article (CrossRef Link)

[6] Choi, J., & Hwang, E., "Secure multiple access based on multicarrier CDMA with induced random flipping," *IEEE Transactions on Vehicular Technology*, 66(6), 5099-5108, 2017. Article (CrossRef Link)

[7] m. o. h. a. m. m. e. d. Mahdi Hashim, & m. s. Mohd Rahim, "Image steganography based on odd/even pixels distribution scheme and two parameters random function," *Journal of theoretical & applied information technology*, 95(22), 2017.

[8] B. Srinivasan, S. Arunkumar, & K. Rajesh, "A novel approach for color image, steganography using nubasi and randomized, secret sharing algorithm," *Indian Journal of Science and Technology*, 8, 1-8, 2015. Article (CrossRef Link)

[9] A. Jain, "A secured steganography technique for hiding multiple images in an image using least significant bit algorithm and arnold transformation," in *Proc. of International Conference on Intelligent Data Communication Technologies and Internet of Things*, pp. 373-380, Sept. 2019. Article (CrossRef Link)

[10] X. Zhang, F. Peng, & M. Long, Robust coverless image steganography based on DCT and LDA topic classification," *IEEE Transactions on Multimedia*, 20(12), 3223-3238, 2018. Article (CrossRef Link)

[11] L. O. Tresor, & M. Sumbwanyambe, "A selective image encryption scheme based on 2d DWT, Henon map and 4d Qi hyper-chaos," *IEEE Access*, 7, 103463-103472, 2019. Article (CrossRef Link)

[12] S. Sun, "A novel edge based image steganography with 2k correction and Huffman encoding," *Information Processing Letters*, 116(2), 93-99, 2016. Article (CrossRef Link)

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

[14] B. Li, M. Wang, X. Li, S. Tan, & J. Huang, "A strategy of clustering modification directions in spatial image steganography," *IEEE Transactions on Information Forensics and Security*, 10(9), 1905-1917, 2015. Article (CrossRef Link)

[15] B. Li, S. Tan, M. Wang, & J. Huang, "Investigation on cost assignment in spatial image steganography," *IEEE Transactions on Information Forensics and Security*, 9(8), 1264-1277, 2014. Article (CrossRef Link)

[16] W. Mazurczyk, & L. Caviglione, "Steganography in modern smartphones and mitigation techniques," *IEEE Communications Surveys & Tutorials*, 17(1), 334-357, 2015. Article (CrossRef Link)

[17] J. Li, X. Li, B. Yang, & X. Sun, "Segmentation-based image copy-move forgery detection scheme," *IEEE transactions on information forensics and security*, 10(3), 507-518, 2015. Article (CrossRef Link)

[18] L. Guo, J. Ni, & Y. Q. Shi, "Uniform embedding for efficient JPEG steganography," *IEEE transactions on Information Forensics and Security*, 9(5), 814-825, 2014. Article (CrossRef Link)

[19] M. M. Emam, A. A. Aly, & F. A. Omara, "An improved image steganography method based on LSB technique with random pixel selection" *International Journal of Advanced Computer Science and Applications*, 7(3), 361-366, 2016. Article (CrossRef Link)
[20] 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*, 76(6), 8597-8626, 2017. Article (CrossRef Link)
[21] 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*, 76(3), 3519-3536, 2017. Article (CrossRef Link)
[22] J. Mielikainen, "LSB matching revisited," *IEEE signal processing letters*, 13(5), 285-287, 2006. Article (CrossRef Link)
[23] P. S. Narule, N. D. Patil, P. S. Kurade, P. D. Patil, & 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, April 2015. Article (CrossRef Link)
[24] Z. Xia, X. Wang, X. Sun, Q. Liu, & N. Xiong, "Steganalysis of LSB matching using differences between nonadjacent pixels," *Multimedia Tools and Applications*, 75(4), 1947–1962, 2016. Article (CrossRef Link)
[25] K. Muhammad, J. Ahmad, H. Farman, Z. Jan, M. Sajjad, & S.W. Baik, "A secure method for color image steganography using gray-level modification and multi-level encryption," *KSII Transactions on Internet and Information Systems (TIIS)*, 9(5), 1938-1962, 2015. Article (CrossRef Link)
[26] R. Z. Wang, C. F. Lin, & J. C. Lin, "Image hiding by optimal LSB substitution and genetic algorithm," *Pattern recognition*, 34(3), 671-683, 2001. Article (CrossRef Link)
[27] C. C. Thien, & J. C. Lin, "A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function," *Pattern recognition*, 36(12), 2875-2881, 2003. Article (CrossRef Link)
[28] S. Dumitrescu, X. Wu, & Z. Wang, "Detection of LSB steganography via sample pair analysis," in *Proc. of International workshop on information hiding*, pp. 355-372, Oct. 2002. Article (CrossRef Link)
[29] W. Luo, F. Huang, & J. Huang, "Edge adaptive image steganography based on LSB matching revisited," *IEEE Transactions on information forensics and security*, 5(2), 201-214, 2010. Article (CrossRef Link)
[30] H. R. Kanan, & 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*, 41(14), 6123-6130, 2014. Article (CrossRef Link)
[31] C. K. Chan, & L. M. Cheng, "Hiding data in images by simple LSB substitution," *Pattern recognition*, 37(3), 469-474, 2004. Article (CrossRef Link)
[32] H. C. Wu, N. I. Wu, C. S. Tsai, & M. S. Hwang, "Image steganographic scheme based on pixel-value differencing and LSB replacement methods," *IEEE Proceedings-Vision, Image and Signal Processing*, 152(5), 611-615, 2005. Article (CrossRef Link)
[33] L.Y. Por, D.B. Yin, T.F. Ang, and S.Y. Ong, "An enhanced mechanism for image steganography using sequential colour cycle algorithm," *Int. Arab J. Inf. Technol.*, 10(1), 51-60, 2013.
[34] V.M. Viswanatham, and J. Manikonda, "A novel technique for embedding data in spatial domain," *International Journal on Computer Science and Engineering, IJCSE*, Feb. 2010.
[35] R. K. L., Nielsen, & P. Grabarczyk, "Are Loot Boxes Gambling? Random reward mechanisms in video games," *Transactions of the Digital Games Research Association*, 4(3), 2019. Article (CrossRef Link)
[36] R. Haryadi, & H. Pujiastuti, "PhET simulation software-based learning to improve science process skills," *Journal of Physics: Conference Series*, Vol. 1521, No. 2, p. 022017, April, 2020. Article (CrossRef Link)

[37] K. Kordov, & S. Zhelezov, "Steganography in color images with random order of pixel selection and encrypted text message embedding," *PeerJ Computer Science*, 7, e380, 2021. Article (CrossRef Link)
[38] A. Li, "Potential Weaknesses In Pseudorandom Number Generators!," 1-12, 2017.
[39] S. Gangurde, and K. Tiwari, "LSB Steganography Using Pixel Locator Sequence with AES," in *Proc. of 2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)*, 2021. Article (CrossRef Link)
[40] M. Hussain, A. W. A. Wahab, Y. I. B. Idris, A. T. Ho, & K. H. Jung, "Image steganography in spatial domain: A survey," *Signal Processing: Image Communication*, 65, 46-66, 2018. Article (CrossRef Link)
[41] M. Hussain, A. W. Wahab, Y. I. B. Idris, A. T. Ho, & K. H. Jung, "Image steganography in spatial domain: A survey," *Signal Processing: Image Communication*, 65, 46-66, 2018. Article (CrossRef Link)
[42] M. J. AlSinani, A. Kayed, & G. Al Farsi, "Improving data security using modified LSB-based image steganography technique," *Journal of Hunan University Natural Sciences*, 47(8), 2020.
[43] S. Bhallamudi, "Image Steganography Final project–Report," In Tech. Rep.. Wright State University, 2015.

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.

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.

Maryam Al-Risia earned a B.S. in Information Science and Technology from Mazoon College, Muscat, Oman, and a master's degree in computer science from Sohar University, Sohar, Oman. Her research interests include computer-based learning.