Journal of Information Hiding and Multimedia Signal Processing ©2021 ISSN 2073-4212
Ubiquitous International Volume **12**, Number **2**, June **2021**

A Novel Approach to Information Hiding Technique using ASCII Mapping Based Image Steganography

Huwaida T. Elshoush

Computer Science Department
Faculty of Mathematical Sciences
University of Khartoum, Sudan
htelshoush@uofk.edu

Ibtihal A. Ali

Computer Science Department
Faculty of Mathematical Sciences
University of Khartoum, Sudan
bebe.it@hotmail.com

Mahmoud M. Mahmoud

Computer Science Department
Faculty of Mathematical Sciences
University of Khartoum, Sudan
Mahmoud.mustafa101@gmail.com

Abdelrahman Altigani

Computer Science Department
College of Computer Science and, Information Technology
Imam Abdulrahman Bin Faisal University, Saudi Arabia
a.altigani@gmail.com

Received January 2021; revised March 2021

Abstract. *Steganography is hiding secret data in a cover without suspicion. This paper proposes a new information hiding technique using image steganography based on ASCII code matching. The cover image file is first read as decimal and the American Standard Code of Information Interchange (ASCII) code characters are searched, then the ASCII codes positions are stored in a mapping table. Similarly, the secret message is converted into decimal and hence matched to know the characters' positions. Compression and encryption have also been applied to the characters' positions of the secret message before being sent. The great advantage is that there will be no change in the image size nor pixels' values, only the positions of the secret message are rather noted. Moreover, several experimental tests have been applied to demonstrate the efficiency and performance of the proposed method to evaluate it according to the widely used metrics: Mean Squared Error (MSE), Peak Signal to Noise Ratio (PSNR) and the hiding capacity. From the security aspect, Structural Similarity Index Measure (SSIM) together with the Histogram analysis have been performed. The proposed method is further compared with some related works. The experimental results verified the effectiveness of the proposed method.*

Keywords: Image Steganography, Information Hiding, Mapping Techniques, ASCII Matching Techniques

1. **Introduction.**

Over the decades, the rapid evolution of communication technology allowed the sharing and transferring of sensitive information over the internet, thus necessitating information security. The information security system can be classified into cryptography and information hiding techniques, where information hiding techniques are further classified into steganography and digital watermarking as shown in figure 1. Cryptography is one such way that converts the original information to an encoded form, thus providing confidentiality. But this encrypted form may often draw imposters, which may have harmful aftereffects. Watermarking is also used to protect, verify and authenticate the user/owner identity. This preserves copyrights. It embeds any form of marker in a carrier which may or may not be covertly concealed. Thus, digital watermarks may also be recognizable [22]. Steganography, on the other hand, hide secret messages into innocuous digital media without inducing suspicion [16]. The word steganography comes from the two Greek words, stegos that means cover and grafia which means writing [1].

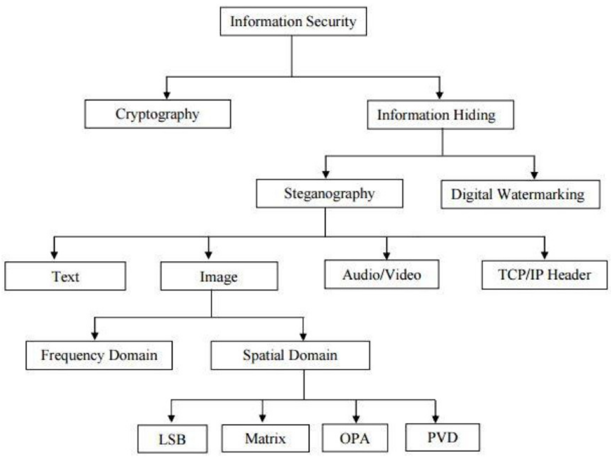

Figure 1. Overview of Information Security System and Steganography Classifications

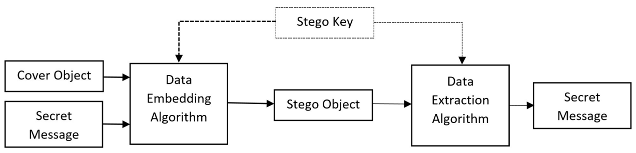

Figure 2. General steganographic model [6]

Steganographic process is generally presented by Sadek et al. [6] as shown below in figure 2. A good steganographic method should have good visual/statistical imperceptibility and a sufficient payload. The former is essential for the security of covert communication and the latter ensures that a large quantity of secret data can be conveyed [17].

It is desired to maximize the amount of hidden information (embedding capacity) while preserving security against detection by unauthorized parties (confidentiality) [18][34][35][36].

Hamid et al. [3] mentioned three properties to measure the performance of a steganographic system, which are:

- *Imperceptibility:* is the statistical undetectability of the data which means the difficulty of discovering the existence of a hidden message.
- *Capacity:* Steganographic capacity is the maximum information that can be hidden in a carrier without detection.

- *Robustness:* measures the ability of the steganographic technique to survive the attempts of removing the hidden information. Such attempts include, image manipulation like cropping or rotating.

The challenge is to achieve a good balance between the above three goals of steganography techniques [29].

Steganography can be categorized according to the type of carrier into four categories: digital media, linguistic/text, file system and network steganography [2]. Furthermore, digital media can be:

- *Image Steganography:* The images are considered the most carriers used to hide data [3]. Digital Image is most widely used on internet and provides high redundant data [4][21]. Image steganography is hard to detect by the Human Visual System (HVS).
- *Audio Steganography:* inserts the hidden bits in the place of redundant bits, which can be modified without decreasing the quality of the carrier [5].
- *Video Steganography:* The secret data can be hid into videos either in the images or audio components [6].

Based on the hiding mechanism, image steganography can be classified into two types [19] as illustrated in figure 1:

- *Frequency (or transform) domain:* here the cover image is transformed into the frequency domain and secret information is hidden into the coefficients. There exist many techniques like Discrete Wavelet Transform (DWT), Fast Fourier Transform (FFT), and Discrete Cosine Transform (DCT), etc.
- *Spatial domain:* the secret message is hid in the bitmap image pixels. Most commonly one is the Least Significant Bits (LSBs) where the LSBs are modified to embed the secret message [11][12][27]. Other techniques include Pixel Value Differencing (PVD)[30] and Mapping base techniques [14][23][24][28][31]. These latter techniques generate the cover-secret mapping that will *realize* the hidden secret message without embedding the actual message.

  Thus, the mapping table does not reveal any correspondence to the secret message, and even if disclosed to an attacker will not convey any useful information without the cover file. Mapping-based techniques can be classified into pixel mapping [32], bit mapping [33], ASCII mapping [14]. This proposed method is a mapping based technique using ASCII codes.

ASCII code is the computer's numeric representation for the keyboard characters. It is ranged from 0-255. The character is encoded in 1 byte, actually in 7 bits and the 8^{th} bit is used to store symbols of other languages such as $\grave{a}$ *or* $\widetilde{n}$ [9][10].

The initiation of this research work is precisely that the steganography process is carried out in an unrevealed manner and without any difference in the cover file. Moreover, there is no ceiling for the secret message size to be hidden, as the proposed method is based on a comparison between the ASCII codes of the secret character and the image cover file. There is no objection to using a specific position, being already used, more than once if that character appears again somewhere. Hence, there will be no limit to the size of the secret text message to be hidden. Above that, the Huffman compression algorithm is used first to reduce the size of the confidential message to a certain extent, as the smaller the size of the hidden message, the greater the quality of steganography and the better the performance time. Furthermore, the message is encrypted using Advanced Encryption Standard (AES) before hid using the new proposed model.

The paper is organized as follows: section 2 presents the related work. The proposed method is explained in section 3, together with elucidating the hiding and extraction processes. Section 4 discusses the implementation of the proposed method where the experimental results are discussed in section 5. Finally, section 6 concludes the paper.

2. Related Work.

The steganography ASCII Mapping Technique (AMT) is used to create an encoded table by mapping the text message and matching some bits with that of the cover image. Recently, researchers have designed several steganography ASCII mapping techniques.

In December 2010, Sukhpreet Kaur and Sumeet Kaur [28] presented a proposed model for steganography by comparing and matching. Their methodology for concealment was that written letter and marks are represented in 7 digits of bits and saved in a table called T1. A letter is read from the secret message and the corresponding ASCII code is found in T1, and then the search process is carried out inside the pixel in the image so that the comparison is first with the red inside the pixel. If there is a match, the row and column for this pixel are saved in another matrix and that is where the conceal result is stored in. In the event that there is no match in the red, the conversion to blue will take place, and the comparison process will take place. If there is no matching, the green conversion will take place and so on for next pixel. After this process, the matching result is compressed with a lossless algorithm, and then encrypted using AES algorithm with a length of 128.

It is noted from the nature and composition of this model that the retrieval process is somewhat complicated, as there are no clear features that show where the concealment took place. That is, the matching process takes place in a way that is unknown to the recipient, because the recipient of the message does not have a parameter that indicates to him where the specific byte that must be read inside the pixel is (red, green or blue).

In 2011, Tech et al. [26] converted a secret message into ASCII by mapping letters to pixels and indicator channels. As ASCII code is a 7 bit code, so for matching bits every byte only 7 MSBs of each channel will be used and the LSB of all three channels are free to work as an indicator. Their method works as follows: the indicator channels will be selected for every pixel. Every time one moves to next pixel, the indicator channel pair will be selected depending upon the value of pseudo random number. Basically for every pixel, a new pseudo random number is generated. Convert that into binary bit sequence. Count number of 1's and of zero's present in the bit sequence. Also calculate the parity of the pseudo random number.

In the November 2011, Al-Husainy [23] proposed a method that uses the ASCII code, but it is different from our method. According to his experimental results, his method can embed a limited length of the secret message depending on the size of unused bytes at the end of the stego-image file. Our method has a flexible size and one can hide unlimited size messages (as long as all its ASCII codes are found in the image) without any suspicion. It also changes the characters' positions for every new message. On the other hand, his method embeds the large message in different stego-images or using the same image many times and that might increase the suspicion. But clearly, Al-Husainy method applies some types of encoding which increases the security of the secret message. Both methods use Huffman coding method for compression.

In 2013, V. Sharma and S. Kumar [13] tried to replace the binary values of RGB components in each pixel by the corresponding binaries of ASCII code for the characters of the secret message. They replace the red component of the first pixel with the first character and replace the green component of the second pixel with the second character. They store the information about bits embedded in a binary address file. Then, the image is covered by another image. Their method shows that using .bmp images works more efficiently than .jpg while the size of the stego image was the same as the original image.

In January 2017, Umamaheswari and Sumathi [14] introduced a technique that matches the secret message and the cover image binary value. They use the position of LSB values that match the bits of the secret message. They just take the green bits in each pixel until the message bits are complete; otherwise, the algorithm starts searching the red and then the blue. They note the positions in a file which is opened by a secret key at the receiver, then the receiver combines the bits to form the ASCII characters of the message. Their method is comparable to ours in the matching and sending of the positions separately, but theirs will produce large output. Since they use a byte for every bit and that means 7 bytes for every character, so the hiding process will take 7 times more than the proposed method. Moreover, once the position is used, it cannot be used again and this limits the size of the hidden message.

A year later in February 2018, Hussein et al [24] presented a new steganography technique called "Hiding text in gray image using mapping technique". Their method matches the bits of the text message with that of the cover image and save the locations of matching parts. The system saves the character parts matching and the location of which part of the pixels. Then it changes the related flag from zero to one for the matched locations so that they cannot be used again to strength the technique and make it more secure. Reading image in bits may take long time to search all pixels, while our method reads the image and the text in decimal representation, so it takes less time. Concerning the mapping table size, Hussein et al method has more size of the pixel location (4 $\times$(row, column)) than our method which has three parts (row, column, R/G/B) only. Even their position array is represented in 8 columns whiles the proposed method in 3 columns only. Furthermore, the colorful image (RGB) is more useful to find all message characters in the ASCII representation. Moreover, the proposed method has unlimited hiding capacity.

Three months later in May 2018, Keshav Joshi [25] proposed an approach that uses ASCII values to embed a secret message into an image. The method divides each character ASCII value to three segments and combines with the RGB values. So each last digit of the RGB decimal values is replaced with data values. Joshi's method is a bit similar to the Al-Husainy [23] approach as they both change some of pixels' values by embedding the secret message. In our method, there is no any change at all on the image. Joshi's approach is not applicable on loosy formats like JPG, but our method depends on matching ASCII codes of secret messages and images so it is applicable on any format readable in MATLAB.

3. The Proposed ASCII Based Matching Image Steganography Method.

In the digital color images each 8 bits represents a color in the RGB color model, Red, Green and Blue, to produce a pixel of 24 bits [7][8]. The LSB algorithm distributes the embedded characters to the pixels in the 8^{th} bit or it can be done in the last two bits, namely 7^{th} and 8^{th} bits. Considering that the mode of the image is RGB, this makes the LSB needs two or three pixels to hide one character. Since the RGB color contains 24 bits (8 bits for each color), the ASCII codes of the secret characters and those of the image colors can be matched, save the characters' positions in the image and then send the positions to the receiver. The positions play the role of the secret key in the symmetric algorithms. The comparison and matching process on a byte by byte basis is better than on a bit by bit basis. The reason is that in order to compare one character using bit by bit, the comparison is performed 8 times to complete on character, but when one matches using byte by byte, the comparison is done only once per character. Figure 3 demonstrates the proposed method. Note that the characters positions are sent in a separate channel from the cover image for more security. Furthermore, the characters positions are first compressed by Huffman algorithm and then encrypted by AES-128.

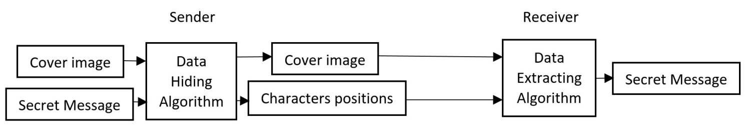

FIGURE 3. The proposed method

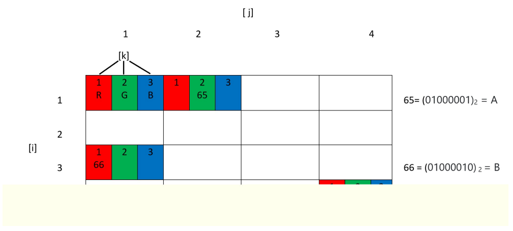

FIGURE 4. Proposed method matching process

The position number contains 3 values [i: row, j: column, k: location]; k is fixed, 1 for red, 2 for green and 3 for blue. If the location is complete, the column number should be increased by one, and after the last column is reached, it goes to the next row and so on. For example, consider that one wants to send a password "ABC", the positions of A, B & C are [1,2,2], [3,1,1], [4,4,3] respectively as illustrated in figure 4.

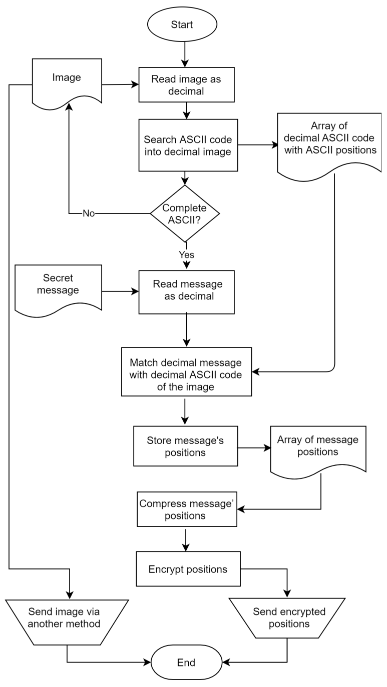
Start
Image
Read image as decimal
Search ASCII code into decimal image
Array of decimal ASCII code with ASCII positions
No
Complete ASCII?
Yes
Secret message
Read message as decimal
Match decimal message with decimal ASCII code of the image
Store message's positions
Array of message positions
Compress message' positions
Encrypt positions
Send image via another method
Send encrypted positions
End

FIGURE 5. Hiding using ASCII Based Matching Image Steganography

3.1. **Hiding using ASCII Based Matching Image Steganography.** The steps of the hiding algorithm are shown in figure 5. It starts with selecting the cover image and reading it as decimal because it is faster than binary (reading 1 number rather than 8 bits). The rows and columns depend on the number of pixels in the width and height of the image. Each pixel has 3 bytes which represent character position. Then it searches the ASCII code characters in the image bytes and store it in a mapping/matching table. The ASCII values should be checked if it is available to represent any kind of message, otherwise another image is chosen. The proposed method converts the secret message to decimal numbers to match the message decimal values with the ASCII decimal values in the ASCII mapping table to find the message positions. Instead of searching the whole image for every character each time, the algorithm first goes back to the stored positions array to check availability, so it does not need to search in the image for every character again. Then, it stores message's positions in a *positions array*. Moreover, these *positions array* are first encrypted by AES-128 after being compressed by Huffman algorithm.

Algorithm 1: The Hiding Algorithm

```
Input: CF: image cover file
       M : the secret message to be hidden
Output: P : the encrypted compressed secret message positions array

1  read CF as decimal
2  for $i = 1$ to $n$ do
       /* n is the length of the ASCII code list                                   */
3      if decimal values of ASCII == decimal values of image then
4          add decimal value to ASCII-positions
               /* ASCII-positions[row][column][location]                          */
5  if all required ASCII values are found then
6      goto 9
7  else
8      goto 1
9  read M characters as decimal values
10 if decimal values of M == ASCII values in ASCII-positions [] then
11     add ASCII locations to M-positions[][][]
12 compress(M-positions[ ][ ][ ]) /* Compress the secret message positions using
       Huffman encoding                                                           */
13 P = encrpyt(compressed(M-positions[ ][ ][ ])) /* Encrypt the compressed the
       secret message positions using AES algorithm                               */
14 send P to the receiver
15 send CF to the receiver
```

3.2. **Extraction using ASCII Based Matching Image Steganography.** Figure 6 depicts the steps of the extraction algorithm. In the extraction phase, the receiver decrypts the encrypted positions of the secret message to decipher it, and then finally decompresses them. Then the ASCII codes of the secret message characters are procured from the *positions array* in the image cover file. For example, if the previous position [1,3,2] is one of the message positions, the algorithm will search in the ASCII positions array to find out its value, 65 as shown in figure 4. Finally, the values will be converted to its real characters and written into a file to be displayed to the receiver.

4. **Implementing the Proposed Method.** The proposed method is implemented using MATLAB software version R2016b. All the experimental results were tested on a laptop with Windows 10, Core i5 processor with 2.5 GHz speed and 4 GB RAM.

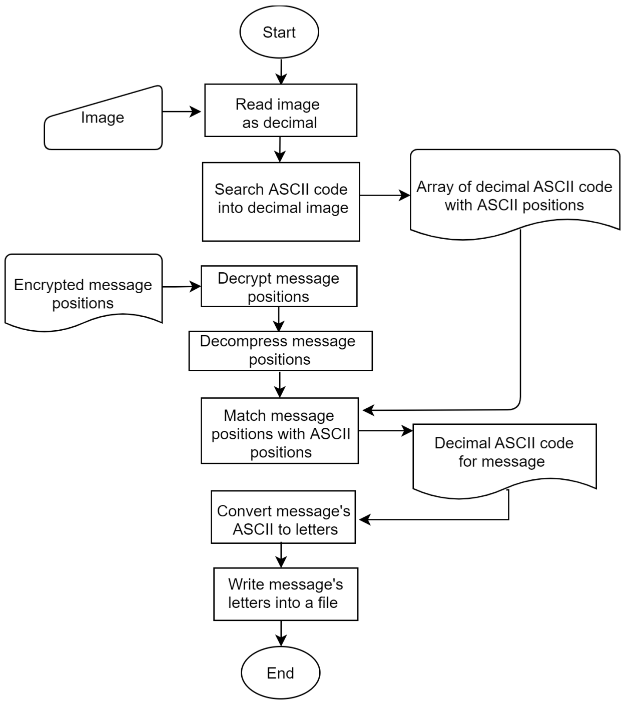
Start
Image
Read image as decimal
Search ASCII code into decimal image
Array of decimal ASCII code with ASCII positions
Encrypted message positions
Decrypt message positions
Decompress message positions
Match message positions with ASCII positions
Decimal ASCII code for message
Convert message's ASCII to letters
Write message's letters into a file
End

FIGURE 6. Extraction using ASCII Based Matching Image Steganography

Algorithm 2: The Extraction Algorithm

```
Input: P : the encrypted compressed secret message positions array
       CF: image cover file
Output: M : the secret message to be hidden

1 read CF as decimal
2 read P

3 decrypt P using AES algorithm /* Decrypt the compressed the secret message
    positions using AES algorithm                                          */
4 uncompress the decrypted P using Huffman algorithm /* Uncompress the secret
    message positions using Huffman encoding                               */
5 for $i = 1$ to $n$ do
    /* n is the length of the secret message positions[ ][ ][ ]             */
6   if secret message positions[ ][ ][ ] == ASCII positions[ ][ ][ ] then
7     return ASCII decimal value

8 convert the ASCII decimal values to characters

9 print the secret message M
```

Figure 7 shows the cover file, Baboon image, of size 243×207, used in the implementation of the proposed method for a 10 bytes secret message. Table 1 shows an example of a hiding process in the proposed model. The secret message is "*HELLO WORLD*" where the characters are converted to the corresponding ASCII code and the comparison is done in the cover file, Baboon Image. All ASCII values (from 0 to 255) are first searched in the cover file, and stored as shown in the mapping table 1. The character positions (ASCII code positions) are extracted and noted in the *positions array* table 2. In this example, the secret message was only 10 bytes in size. The output size after finding the hidden characters positions was 30 bytes, as shown in table 2. Here, no compression was needed as the input size was small. These hidden characters positions are then encrypted using the AES-128 algorithm, where the output is always a multiple of 16, see table 3.

Table 1 Secret Message ASCII Values for a 10 Bytes Secret Message

Character	**ASCII Code**	**Positions**		
		Row	***Column***	***Byte inside pixel***
H	72	2	31	3
E	69	2	43	3
L	76	2	45	2
L	76	2	45	2
O	79	2	1	1
W	87	2	19	2
O	79	2	11	1
R	82	2	98	3
L	76	2	45	2
D	68	2	8	2

Table 2 Positions Identified in Baboon Image Before Encryption for a 10 Bytes Secret Message

2	31	3	2	43	3	2	45
2	2	45	2	2	1	1	2
19	2	2	1	1	2	98	3
2	45	2	2	8	2		

Table 3 Positions Identified in Baboon Image After Encryption for a 10 Bytes Secret Message

29	88	234	223	235	189	118	0
158	124	22	247	174	59	176	226
30	27	140	242	12	28	23	222
204	85	100	126	4	232	241	32

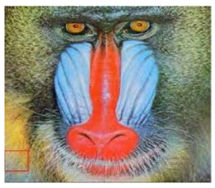

Figure 7. Sample Image of Baboon

5. **Experimental Results and Discussion.** This section details the experimental results of the proposed method. Several results will be discussed to demonstrate the efficiency and performance of the proposed method to evaluate it according to the widely used metrics: Mean Squared Error (MSE) and Peak Signal to Noise Ratio (PSNR). Moreover, the security of the proposed model is discussed including Structural Similarity Index Measure (SSIM) together with the Histogram analysis. Time analysis was also evaluated. Finally, the advantages and disadvantages of the proposed method together with comparisons with some related works were discussed.

5.1. **Imperceptibility.** This aspect measures how much difference (distortion) was caused by data hiding in the original cover, where the higher the stego-image quality, the more invisible the hidden message. The stego-image quality can be judged by using PSNR and MSE. MSE of the stego image is the average squared difference between stego pixels and original cover image pixels using equation 1 [19]. PSNR is the most commonly used metric for evaluation of any steganography. It is the ratio between the maximum signal possible and the influence of modifying noise to the fidelity of its representation. It is calculated using MSE and equation 2, and it is expressed in decibel (dB) [19].

The lower the MSE value, the higher the steganography quality and vice versa. For PSNR, the higher its value, the greater the quality of the concealment, which means better image quality (less distortion).

This proposed method is based on reading the bytes and knowing their positions only, without changing anything in the image cover file, thus the MSE is zero with any possible message size. The PSNR is directly proportional to MSE. That is, the closer the MSE to zero, the higher the PSNR, and hence the better the steganography quality. Therefore, the PSNR value will actually be infinity.

Table 4 and figure 8 show approximate values for MSE, whose real value is, as a matter of fact, zero. These close-to-zero values were taken to measure the quality of the steganography process by PSNR and hence the highest value recorded for the PSNR (which is actually infinity) was calculated by the nearest MSE value to zero.

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(f_{ij} - g_{ij})^2 \qquad (1)$$

$$PSNR = 10log_{10}\frac{L^2}{MSE} \qquad (2)$$

M, N is the row and column for the image cover file.

f$_{ij}$ represents the original image data cover file.

g$_{ij}$ represents the stego file after hiding the secret message inside it.

L is the maximum fluctuation in the input image data type.

(For example, if the input image has a double-precision floating-point data type, then L is 1. If it has an 8-bit unsigned integer data type, L is 255, etc.)

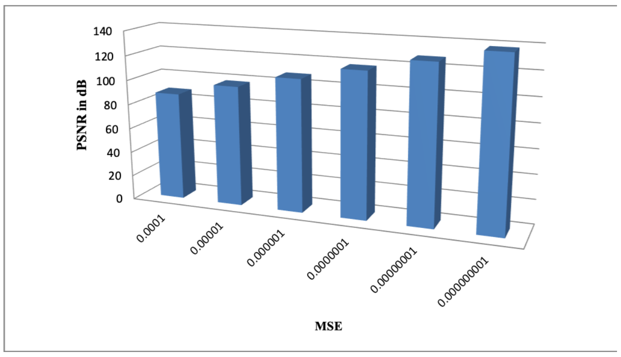

FIGURE 8. PSNR based on decreasing MSE value (approximated from zero)

TABLE 4 Experimental result for a cover file of size 19.4 KB and different message sizes

MSE(approximated)	PSNR (in dB)
0.0001	88.16
0.00001	98.16
0.000001	108.16
0.0000001	118.16
0.00000001	128.16
0.000000001	138.16
0	Infinity

5.2. **Hiding Capacity.** The *hiding capacity* or *payload* is one of the most important factors of evaluating steganography techniques. In the proposed method, each character is represented by 3 positions in the mapping table, as shown in tables 1 and 2. If, for example, the Baboon 243×207 image is used, and having 3 channels (Red, Green, and Blue):

then the total number of positions = 243 ×207 × 3 = 50,301 ×3=150,903

Hence, if each position is used only once, then this implies that 50,301 bytes can be hidden. Nevertheless, the proposed method uses the positions more than once, as long as all characters ASCII codes can be found in the image. Consequently, the proposed method has *unlimited hiding capacity.*

5.3. **Security Analysis of the Proposed Method.**

5.3.1. *Structural Similarity Index Measure (SSIM).* This method is normally employed to quantify the similarity between two given images of the same dimension [19]. Therefore, the mean similarity measure evaluates the quality of stego-image, F with respect to the original image, E, given by equation 3 [20][22].

TABLE 5 Experimental results of SSIM for different cover files of different sizes

Image name	Image size in KB	Pixels size	SSIM
Cat1	14	200 X 200	1
Baboon	12	243 X 207	1
Dog	10	240 X210	1
Eye	23	236 X 225	1
Lena	35	247 X 248	1
Chameleon	43	252 X 253	1
Cat 2	21	211 X 185	1

$$SSIM(c,s) = \frac{(2\mu_c 2\mu_s + v1)(2\sigma_{cs} + v2)}{({\mu_c}^2 + {\mu_s}^2 + v1)({\sigma_c}^2 + {\sigma_s}^2 + v2)} \tag{3}$$

where, μ_c is the average of cover pixels

μ_s is the average of stego pixels

${\mu_c}^2$ is the variance of cover pixels

${\mu_s}^2$ is the variance of stego pixels

μ_{cs} is the covariance of cover and stego

v1 and v2 are two variables having values (k1 L)^2 and (k2 L)^2, where L is 2^8–1, k1 = 0.01 and k2 = 0.03

The best value 1 is achieved if and only if the cover and stego files are the same, and as there is absolutely no change between them in the proposed method, SSIM value is 1 for all images, see table 5.

5.3.2. *Histogram Analysis.* Pixel intensity versus frequency plot of an image is known as a histogram plot. In this research, however, there is no difference between the cover files and stego image, as there is no actual hiding. That is why the histograms illustrated in figure 9 are identical.

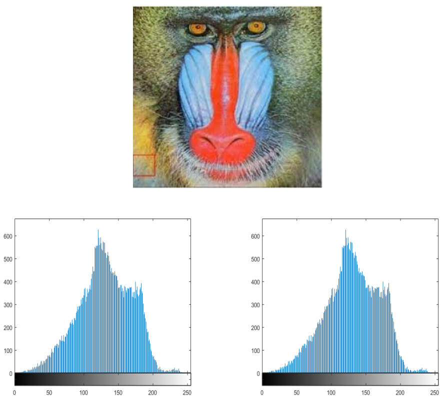

FIGURE 9. Baboon and histograms of Cover and Stego images

5.3.3. *Security of the Proposed Method.* Security in the steganography process is measured in two aspects:

1. Aspect 1:
   No doubt about the existence of the message in the cover file
   This is measured by the extent of the impact of the cover file by the hiding process and the size of change that occurs in the cover file and measured using the criteria MSE and PSNR, as proven effectively in table 4.
2. Aspect 2:
   If you know that the cover file is suspected of having a secret message
   In this case, according to the proposed model, there are three different scenarios:
   - Scenario 1:
     Know the cover file only
     If a third party obtained a single cover file, it cannot retrieve the secret message because it does not own the positions array through which the bytes are extracted from the cover file.
   - Scenario 2:
     Know positions array only
     In this scenario, it is assumed that if any third party has a positions array, even if decrypted, there must be that cover file on which the hiding process occurred on it (comparison) to extract the secret message from it.
   - Scenario 3:
     Know the cover file and the positions array
     In this last scenario, assuming that there is a third party that owns the cover file and the positions array, it will collide with positions array compressed and encrypted with AES algorithm (with a 128-bit key). The probability of breaking the AES encryption key of a length of 128 bits with a brute force attack is 2^{128}.

5.4. **Time Analysis.**

5.4.1. *Effect of the Compression Process.* The performance of the hiding process is better for smaller files. Thus, table 6 and figure 10 show the effectiveness of the compression process on the output of the hiding process.

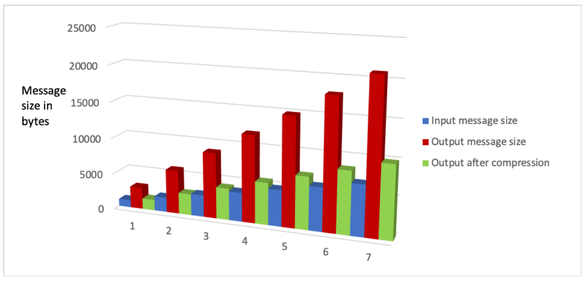

FIGURE 10. Result of compressing file

TABLE 6 Experimental result of compression for a cover file of size 19.4 KB and different message sizes

Input Message Size	**Output Message Size**	**Output File Size After Compression**
1000	3000	1479
2000	6000	2917
3000	9000	4350
4000	12000	5775
5000	15000	7204
6000	18000	8609
7000	21000	10020

TABLE 7 The execution time of hiding different message sizes in different images with different dimensions and sizes.

Input secret message size in Bytes	**Cat1 (14KB)** 200X200	**Baboon (12KB)** 243X207	**Dog (10KB)** 240X210	**Eye (23KB)** 236X225	**Lena (35KB)** 247X248	**Chameleon (43KB)** 252X253	**Cat2 (21KB)** 211X185
1000	6.2224	6.849439	6.7506	6.294	6.2076	6.7324	7.0878
2000	9.5026	9.961385	10.5165	9.5726	9.3793	10.2626	11.0553
3000	12.6297	13.27577	14.1755	12.6469	12.6539	13.989	14.8868
4000	15.7437	16.70098	17.8258	15.7411	15.6	17.3801	18.6238
5000	18.7442	18.59312	21.2194	18.7386	18.6979	20.8714	22.4525
6000	21.645	21.32168	24.8873	21.5874	21.9496	24.2032	26.1453
7000	24.5692	24.16203	28.3069	24.6644	24.9003	27.6058	30.0141

5.4.2. *Time Analysis of Hiding Different Message Sizes on Different Images.* The results displayed in table 7 show that the execution time of the proposed method is directly proportional to the size of the hidden message.

The images in figure 11 is ordered according to the size of images. Hence, from table 7 and figure 11, we noted the dispersion and differences in the time it takes with the different sizes of the different images in which a secret message of size 7000 bytes was hidden. We concluded that there is no clear relation (proportional or reverse) between the time and the size of the image (cover file). That is a logical conclusion if we look

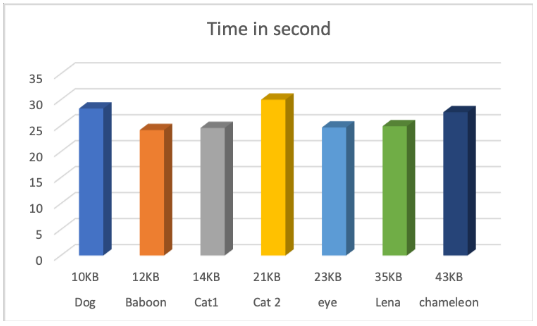

FIGURE 11. The execution time of hiding message size of 7000 Bytes on different cover file sizes.

at the nature of the proposed method, which is based on the principle of comparison; if the correspondence between the message bytes and image bytes in any positions, there is no need to search again in the remainder of the positions if the same character of the message is repeated next time. Actually, that is why the proposed method has an unlimited capacity of hiding.

5.5. **Comparison with Related Works of** [14], [23], [24] **and** [28]. The related works of [28], [23], [14] and [24] are closely related to the proposed method. Hence, a comparative analysis of our method and the related works is presented in table 8. Their differences lies in the following:

1. Umamaheswari et al. [14] use one component in RGB (green) and the first LSB (8^{th} bit) from each pixel. Hussein et al. [24] use one gray scale (1 byte). Al-Husainy [23] method uses the unutilized space at the end of the stego-image file, but search in all RGB components. On the other hand, the proposed method uses all RGB components (24 bits). Moreover, as the proposed method can reuse the positions, it therefore has unlimited capacity if all ASCII values can be found in the mage. Kaur et al.[28] also use all RGB components but one at a time.
2. Umamaheswari et al. [14], Hussein et al. [24] and Kaur et al.[28] methods all read secret messages and images as binaries, this takes more time to search all pixels. However, our method and Al-Husainy [23] read both of secret messages and images as bytes.
3. The matching processes of the Umamaheswari et al. [14] and Hussein et al. [24] compare bit by bit, whereas the proposed model, Kaur et al.[28] and Al-Husainy [23] compare byte by byte.
4. The related work model of Umamaheswari et al. [14] represents any character in 7 bytes, while our method represents each character by 3 bytes of image position (i, j & k), see table 1. On the other hand, Kaur et al. [28] represent any character in 2 bytes. Al-Husainy [23] represents any character in 1 byte.
5. Neither Umamaheswari et al. [14], nor Hussein et al. [24] methods use compression for the positions and this results in an exponential output, unlike ours and the Al-Husainy [23] output are much less as they are compressed using Huffman algorithm. Kaur et al. [28] also uses compression with a lossless algorithm.
6. In the proposed model and Kaur et al. [28], the positions are encrypted using the AES algorithm, hence it is more secure than the related works of Al-Husainy [23], Umamaheswari et al. [14] and Hussein et al. [24] as no encryption is used.
7. In the Umamaheswari et al. [14] model, the positions cannot be used again, which limits the size of the secret message. Similarly, Hussein et al. [24] set flag for the matches' location so that they cannot be used again. Likewise, Al-Husainy [23]

TABLE 8 Comparative Analysis of Proposed Method and the Related Works of [14], [23], [24] and [28]

Criteria	**Proposed method method**	**Kaur et al.** [28] December 2010	**Al-Husainy** [23] November 2011	**Umamaheswari** [14] January 2017	**Hussein et al.** [24] February 2018
Usage of RGB Component	Use all RGB components, (24 bits) hence have more capacity	Use all RGB components but one at a time	Use all RGB components	One component in RGB (green) and the first LSB (8^{th} bit) from each pixel	One gray scale (1 byte)
Reading message/image	Reading image as bytes	Reading image as binaries	Reading image as bytes	Reading image as binaries, this takes more time	Reading image as binary may take long time to search all pixels
Matching process	Byte by byte	Byte by byte	Byte by byte	Bit by bit	2 bits by 2 bits
Matching Output Character representation	Represents any character in 3 bytes	Represents any character in 2 bytes	Represents any character in 1 byte	Represents any character in 7 bytes	Represents any character in 8 bits
Compression	Yes Huffman algorithm	Yes lossless algorithm	Yes Huffman algorithm	No	No
Encryption	Yes AES (128)	Yes AES (128)	No	No	No
Usage of positions	Can use the positions more than once, hence more flexible for any possible message size	Can use the positions more than once	Embeds a limited length of the secret message depending on the size of unused bytes at the end of the stego image file	Positions cannot be used again, which limits the size of the secret message	Positions cannot be used again
Invisibility	No change	No change	Change in in cover file but no distortion appears in the pixels values of the stego-image	No change	No change
Independence of file format	Any format readable in MATLAB	-	.bmp format	-	-

method embeds a limited length of the secret message depending on the size of unused bytes at the end of the stego-image file. Elseways, in the proposed model and Kaur et al. [28] one can use the positions more than once, hence they are more flexible for any possible message size.

8. In our method, Kaur et al. [28] and Hussein et al. [24] there are no changes on the image. On the other hand, Al-Husainy [23] method changes the cover file but no distortion appears in the pixels values of the stego-image.
9. Our method is applicable on any format readable in MATLAB, while Al-Husainy [23] works on .bmp formats.

5.6. Evaluation of the Proposed Method.

The most important requirement is that a steganographic algorithm has to be imperceptible. Arun K Mohan [20] and Shelke et al [15] proposed criteria to test imperceptibility of a stegnographic algorithm:

1. **Invisibility:** The invisibility of a steganographic algorithm is the first and foremost requirement, since the strength of steganography lies in its ability to be unnoticed.

Table 9 Evaluation of the Proposed Method

Criteria	Evaluation
Invisibility	No any change at all in the image
Payload capacity	Unlimited hiding capacity
Robustness against statistical attacks	Very high (zero distortion as no visible change in image)
Robustness against image manipulation	Very low (cropping or rotating will change the positions)
Independent of file format	Any format readable by MATLAB
Unsuspicious files	Very high as no change in image so no suspicion

2. **Payload capacity:** Steganography requires sufficient embedding capacity.
3. **Robustness against statistical attacks:** Statistical steganalysis is the practice of detecting hidden information through applying statistical tests on image data. A steganographic algorithm must not leave such a 'signature' in the image when hiding information that can be easily detected through statistical analysis.
4. **Robustness against image manipulation:** Image manipulation, such as cropping or rotating, can destroy hidden messages. It is preferable for steganographic algorithms to be robust against both malicious or unintentional changes to the image.
5. **Independent file format:** With many different image file formats used on the Internet, it might seem suspicious that only one type of file format is continuously communicated between two parties. The most powerful steganographic algorithms thus possess the ability to embed (or hide) information in any file type. This also solves the problem of not always being able to find a suitable image at the right moment, and in the right format to use as a cover image.
6. **Unsuspicious files:** This requirement includes all characteristics of a steganographic algorithm that may result in images that are not used normally and may cause suspicion. Abnormal file size, for example, is one property of an image that may cause suspicion.

Using the criteria of Arun K Mohan [20] and Shelke et al [15], table 9 shows the evaluation of the proposed method accordingly.

6. Conclusion.

Steganography is embedding secret data in an unsuspicious carrier with no doubts. This paper introduces an information hiding method based on ASCII Mapping Technique image steganography. Its strength lies in the fact that the image cover file is actually not changed completely. After reading the image cover file as decimal, the ASCII codes are searched in the image bytes, and after checking its availability, it is stored in a mapping table. Next, the proposed method converts the secret message to decimal numbers to match the secret message ASCII codes with those of the cover image (using the mapping table), and stores their characters' positions in an array. These characters' positions were first compressed by Huffman algorithm and then encrypted by AES before being sent. The sender sends the image cover file separately from the compressed and encrypted characters' positions, so any interception of any one cannot provide desired objective. The proposed method was implemented and tested for widely used metrics: MSE, PSNR, limitless hiding capacity, SSIM and steganography histogram analysis. Security and time analysis have been performed. Furthermore, a comparative analysis with closely related research were conducted showing its advantages and disadvantages. Hence, the proposed method gave excellent results showing its success.

REFERENCES

[1] B. Swathi, K. Shalini, K. Prasanthi, *A review on steganography using images*, Asian Journal of Computer Science and Information Technology. 2(8), 234-237, 2012.

[2] E. Zieliska, W. Mazurczyk, K. Szczypiorski, *Trends in steganography*, Communications of the ACM, 57(3) 86-95, 2014.

[3] N. Hamid, A. Yahya, R. Badlishah, O. Al-Qershi, *Image steganography techniques: an overview*, International Journal of Computer Science and Security (IJCSS), 6(3), pp: 168-187, 2012.

[4] W. Bender, D. Gruhl, N. Morimoto, A. Lu, *Techniques for data hiding*, IBM System Journal, 35(3/4), pp. 313 – 336, 1996.

[5] P. Jayaram, H. Ranganatha, H. Anupama, *Information hiding using audio steganography–a survey*, The International Journal of Multimedia and Its Applications (IJMA), 3(3), PP: 86-96, 2011.

[6] M. Sadek, A. Khalifa, M. Mostafa, *Video steganography: a comprehensive review*, Multimedia Tools and Applications, 74(17), pp: 7063-7094, 2015. DOI: 10.1007/s11042-014-1952-z.

[7] N. Johnson, and S. Jajodia, *Exploring steganography: seeing the unseen*, Computer 31(2), pp: 26-34, 1998.

[8] *Image structure*, http://jun.hansung.ac.kr/DI/Chapter-2.htm doi: 2018/12/10

[9] *Stack overflow website*, http://stackoverflow.com/questions /14690159/is-ascii-code-7-bit-or-8-bit. doi: 2018/12/10

[10] *ASCII code website*, http://www.ascii-code.com. doi: 2018/12/10

[11] S. Singh, G. Agarwal, *Use of image to secure text message with the help of LSB replacement*, International Journal of Applied Engineering Research, Dindigul, 1(1) 200-205, 2010.

[12] S. A. Laskar, and K. Hemachandran, *High capacity data hiding using LSB steganography and encryption*, International Journal of Database Management Systems (IJDMS), 4(6), December 2012.

[13] V. Sharma, and S. Kumar, *A new approach to hide text in images using steganography*, International Journal of Advanced Research in Computer Science and Software Engineering, 3(4), 701-708, 2013.

[14] G. Umamaheswari, C. P. Sumathi, *A new information hiding technique matching secret message and cover image binary value*, International Journal of Computer Science and Information Security (IJCSIS), 15(1), January 2017.

[15] F. M. Shelke, A. A. Dongre, P. D. Soni, *Comparison of different techniques for steganography in images*, International Journal of Application or Innovation in Engineering & Management (IJAIEM). www.ijaiem.org, 3(2), 171-176, February 2014.

[16] Xin Liao, Yingbo Yu, Bin Li, Zhongpeng Li, and Zheng Qin, *A New Payload Partition Strategy in Color Image Steganography*, IEEE Transactions on Circuits and Systems for Video Technology, 2019. DOI 10.1109/TCSVT.2019.2896270

[17] Xin Liao, Zheng Qin, Liping Ding, *Data embedding in digital images using critical functions*, Signal Processing: Image Communication, 58, 146-156, 2017.

[18] Ali, K. Hmood, B. B. Zaidan, A. A. Zaidan and Hamid A. Jalab, *An Overview on Hiding Information Techniques in Images*, Journal of Applied Sciences, 10 (18), 2010.

[19] Giridhar Maji And Sharmistha Mandal, *Secure and Robust Image Steganography Using a Reference Image as Key*, International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075, Volume-8, Issue-7, May, 2019.

[20] Arun K Mohan, *Hybrid Algorithm For Improved RDH Using Dual Imaging And Histogram Shifting*, Dissertation, Master of Technology In Electronics, Department of Electronics Engineering, Pondicherry University Kalapet, Puducherry, 14 May 2015.

[21] Nitin Kanzariya, Ashish Nimavat and Hardik Patel, Security of Digital Images using Steganography Techniques based on LSB, DCT and Huffman Encoding, *Proceeding of International Conference on Advances in Signal Processing and Communication - Elsevier*, 2013.

[22] Srilekha Mukherjee and Goutam Sanyal, *A multi level image steganography methodology based on adaptive PMS and block based pixel swapping*, Multimedia Tools and Applications, 78:17607–17622, 2019. https://doi.org/10.1007/s11042-018-7127-6

[23] Mohammed Abbas Fadhil Al-Husainy, *A New Image Steganography Based on Decimal-Digits Representation*, Computer and Information Science, Vol. 4, No. 6, November 2011.

[24] Hussein L. Hussein, Ahmed A. Abbass, Sinan A. Naji, Salam Al-augby and Jasim H. Lafta, *Hiding Text in Gray Image Using Mapping Technique*, IOP Conference Series: Journal of Physics: Conference Series 1003 (2018) 012032. Doi: 10.1088/1742-6596/1003/1/012032. 2018.

[25] Keshav Joshi, *A New Approach of Text Steganography Using ASCII Values*, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181. Vol. 7, Issue 05, May 2018.

[26] Jatinder Kumar M.Tech and Mr. Ashish Oberoi. Image Steganography By Using Mapping Of Letters To Pixels And Indicator Channels, *Proceeding of the International conference on Advanced Computing, Communication and Networks'11*, 2011.

[27] Biswajita Dattaa, Upasana Mukherjeeb and Samir Kumar Bandyopadhyaya. LSB Layer Independent Robust Steganography using Binary Addition, *Proceeding of International Conference on Computational Modeling and Security (CMS 2016).* Procedia Computer Science 85 (2016) 425 – 432, 2016.

[28] Sukhpreet Kaur and Sumeet Kaur. A High Security Approach for Image Steganography by Mapping Pixels to Letters, *AIP Conference Proceedings 1324, 280.* https://doi.org/10.1063/1.3526214 Published Online: 03 December 2010.

[29] Alan A. Abdulla, Harin Sellahewa and Sabah A. Jassim (2020). Stego Quality Enhancement by Message Size Reduction and Fibonacci Bit-plane Mapping, *In the Proceeding of Security Standardization Research - Springer. arXiv:2004.12467*, April 2020.

[30] Pyung-Han Kim, Eun-Jun Yoon, Kwan-Woo Ryu, and Ki-Hyun Jung, *Data-Hiding Scheme Using Multidirectional Pixel-Value Differencing on Colour Images. Hindawi - Security and Communication Networks* Volume 2019, Article ID 9038650, 11 pages https://doi.org/10.1155/2019/9038650. 2019.

[31] Mehdi Hussain, Ainuddin Wahid Abdul Wahab, Yamani Idna Bin Idris, Anthony T. S. Ho, Ki- Hyun Jung, *Image Steganography in Spatial Domain: A Survey*, Signal Processing Image Communication. 65.10.1016/j.image.2018.03.012, March 2018.

[32] Aminou Halidou, Youssoufa Mohamadou, and Georges Delort Olle O., *Robust Steganography based on Matching Pixel Locations*, International Journal of Computer Applications (0975 – 8887) Volume 168 – No.12, June 2017.

[33] S. Anitha and T. Ramaprabha, *Using Reference Image an Alternative Approach of Steganography*, International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 ICATCT – 2020 Conference Proceedings, 2020

[34] Bin Li, Junhui He, Jiwu Huang, and Yun Qing Shi, *A Survey on Image Steganography and Steganalysis*, Journal of Information Hiding and Multimedia Signal Processing, Vol. 2, No. 2, pp. 142-172, April 2011

[35] Dejian Fang and Shuliang Sun, *A New Scheme for Image Steganography based on Hyperchaotic Map and DNA Sequence*, Journal of Information Hiding and Multimedia Signal Processing, Vol. 9, No. 2, pp. 392-399, March 2018

[36] Katandawa Alex Kingsley and Ari Moesriami 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, March 2020