IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 5, 2020, accepted April 13, 2020, date of publication April 29, 2020, date of current version May 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2991130

A Multiple-Format Steganography Algorithm for Color Images

ARSHIYA S. ANSARI[1], **MOHAMMAD S. MOHAMMADI**[2], **AND MOHAMMAD TANVIR PARVEZ**[3]

[1]Department of Information Technology, College of Computer and Information Sciences, Majmaah University, Al Majma'ah 11952, Saudi Arabia
[2]Department of Information Technology, College of Computer, Qassim University, Buraidah, Saudi Arabia
[3]Department of Computer Engineering, College of Computer, Qassim University, Buraidah, Saudi Arabia

Corresponding author: Arshiya S. Ansari (ar.ansari@mu.edu.sa)

The work of Arshiya S. Ansari was supported by the Majmaah University's Deanship of Scientific Research under Project R-1441-117.

ABSTRACT This paper presents an image Steganography algorithm that can work for cover images of multiple formats. Having a single algorithm for multiple image types provides several advantages. For example, we can apply uniform security policies across all image formats, we can adaptively select the most suitable cover image based on data length, network bandwidth and allowable distortions, etc. We present our algorithm based on the abstract concept of image components that can be adapted for JPEG, Bitmap, TIFF and PNG cover images. To the best of our knowledge, the proposed algorithm is the first Steganography algorithm that can work for multiple cover image formats. In addition, we have utilized concepts like capacity pre-estimation, adaptive partition schemes and data spreading to embed secret data with enhanced security. The proposed method is tested for robustness against Steganalysis with favorable results. Moreover, comparative results for the proposed algorithm are very promising for three different cover image formats.

INDEX TERMS Data embedding, data hiding, image steganography, JPEG steganography.

I. INTRODUCTION

The area of Steganography generally covers techniques that try to hide information in some media file being transmitted. The goal of Steganography is to ensure that an adversary should not suspect the presence of a hidden message. Text, image, audio or video files can be used as media to conceal the secret message. In *Image Steganography*, after Steganography operations, nobody should be able to notice the visual difference between original image (called *cover* image) and the resultant image (called *stego* image). Text, image, audio or video files can all be treated as sequence of *data bits* and can be transmitted by hiding in cover images using image Steganography. Data bits are also called payload data, secret data, data object and hidden data.

With the best of our knowledge, the reported results for image Steganography focus on cover images of some particular format. Out of the numerous image Steganography algorithms, there are works that use cover images of type JPEG [7]–[9], [18], [21], PNG [11]–[14], [30] and RGB [1]–[6], [19], [20]. In this paper, we present a Generic Steganography Algorithm (called GSA) that is described for abstract image components. The novel aspect of GSA is that, we can implement GSA for all these types of cover images (i.e. JPEG, Bitmap or PNG) with trivial adaptations. In addition, we have implemented GSA for JPEG, Bitmap and PNG formats with very promising results compared to other reported methods.

At this point, the following query may come up in the mind of the reader: what is the benefit of having an image Steganography algorithm that can work for multiple cover image formats? We can answer this question by noting the following points, which also provide the motivations for the current work.

- Having the option to use any types of cover images using one algorithm provides flexibility and simplicity for a user.
- Capacity of a cover image can vary based on the image format. Based on the data length, network bandwidth and allowable distortions, GSA can adaptively select the best cover image format (for the same image) to hide data.
- Since the same algorithm is used for various cover image formats, security levels can be enhanced by modifying

The associate editor coordinating the review of this manuscript and approving it for publication was Vicente Alarcon-Aquino.

only a few parameters (like, using data spreading technique in GSA).

In summary, the proposed method works as follows. We consider a cover image as a sequence of *components* C_i. Each of these components C_i consists of a number of parameters P_j^i, of which we only consider three: $P_{j1}^i, P_{j2}^i, P_{j3}^i$. Out of these three parameters, one is selected at random as the *indicator parameter*. Data bits are stored in one of the remaining two parameters from $P_{j1}^i, P_{j2}^i, P_{j3}^i$ based on the concept of partition schemes. Specific adaptations of the above mentioned process for various cover image types are discussed later in this paper. As an example, a pixel in a Bitmap cover image can be considered as a component C_i with $P_{j1}^i, P_{j2}^i, P_{j3}^i$ being the R, G, B intensity values.

Additionally, we use capacity pre-estimations of cover images to select the best cover image and the concept of data spreading to reduce distortions. Experimental results show that the proposed algorithm performs better compared to other reported works for various cover image types. Thus, this paper contributes to advance the state-of-art in image Steganography in two ways:

- Introducing an algorithm (called GSA) that can be trivially adapted for multiple cover image formats.
- Adaptations of GSA showing promising and better results when compared with other methods.

The rest of the paper is organized as follows. Section II focuses on literature review. Section III gives the abstract description of the proposed method. Section IV presents the implementation issues of our algorithm. Section V describes the adaptations of the proposed algorithm for different cover image formats. Section VI deals with experimental results. Finally, we conclude the work in Section VII.

II. RELATED WORKS

There are a lot of works reported in image Steganography. In this section, we briefly review some of the works related to our approach.

Before we discuss different works, it is good to discuss some factors that we should consider while evaluating a Steganography algorithm. Fig. 1 illustrates some essential factors in Steganography. In *confidentiality*, the Stego image transmission must be confidential, where only the authorized person must be able to read the message, while others should not even suspect it. *Integrity* means that only an authorized person would be able to modify or change the message. *Robustness* is the ability of stego image to resist manipulations like filtering or compression. *Hiding Capacity* is the amount of information hidden in the cover image media. It is measured as bpp (bits per pixel) or bpc (bits per coefficient). In *Authentication*, the origin of the message is recognized correctly with an assurance that identity is not false. *Perceptibility* implies that the Stego image must be same like cover image without visual difference and hidden message should not to be detectable by eyes.

PSNR (Peak Signal to Noise Ratio) is a visual quality estimator for stego images. It is used to measure the

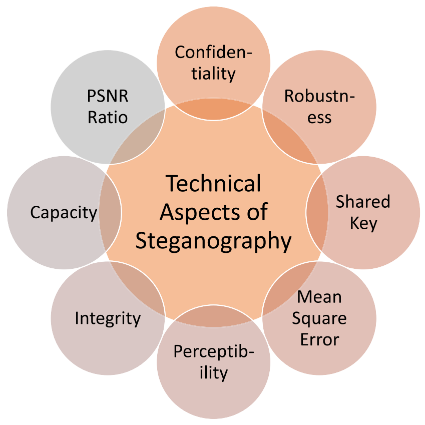


FIGURE 1. **Major technical aspects of an image Steganography algorithm.**

perceptible quality of the modified Steganography image in decibels (dB). A high value for PSNR indicates higher quality of an image. *MSE* (Mean Square Error) is an estimate for error between the original cover image and the output stego (reconstructed) images. There is an inverse relationship between PSNR and MSE, a lower MSE value indicates lower difference between input and output images. If MSE value is less than one, it implies that modified image has been properly restored. *Stego key* is a key that is embedded into the cover media along with the secret data to retrieve the embedded secret data back correctly (explained in detail in Section III). The various image Steganography algorithms attempt to optimize one or more of these factors.

Previous works present various Steganography algorithms for specific image formats [1]–[24], [26]. Algorithms for image formats like JPEG [7]–[9], [18], [21], [23], [24], [26], PNG [11]–[14], and RGB [1]–[6], [19], [20] are discussed here, the most common ones. Table 1 summarizes the various image Steganography algorithms.

A. SPATIAL DOMAIN (RGB/BITMAP) STEGANOGRAPHY

Algorithms for bitmap images work in spatial domain, using direct modifications in the cover image pixels. RGB algorithms provide high capacity but less security because image pixels can be modified directly as per the scene's curves and edges. Examples of RGB algorithms are LSB (least significant bit) substitution method, pixel indicator technique, optimal pixel adjustment procedure, secure key based image realization Steganography, etc.

The techniques in [3], [4], [6], [33], [34] are all LSB substitution based methods, where the basic idea is to embed the message into the right most bits of the pixel array sequentially or randomly without disturbing the original pixel values much. Pixel indicator techniques are based on the concept of an indicator channel and an embedding channel [1]. Amirtharajan *et al.* [2] used both LSB and pixel indicator techniques to enhance security.

TABLE 1. A summary of some recent image Steganography algorithms.

Reference / Target Image format	Basic approach	Data used
Parvez & Gutub [1] /Bitmap	Change channel value based on intensity.	Cover image size 640 × 480, Bit depth: 24, No of pixels = 307200. Data File bitmap 150 × 117, Bit depth: 24 Data length = 150896 bits
Amirtharajan, et al. [2] /Bitmap	Channel selection method, LSB insertion method with modified version of pixel indicator method.	Lena, Baboon, Gandhi and Temple (256 × 256 pixels) as cover images, data size were not defined.
Singh et al. [3] /Bitmap	Combined different techniques, used descriptor SBD to identify the blocks. LSB layer is used to hide data and masked for more protection. Low silence region chosen for embedding secret data.	Cover image size: 259 ×194, secure data size: 3286 bits, and block size: 16.
Sengul Dogan [4] /JPEG	Pixel pair algorithm by using chaotic map for JPEG images to insert the bits in coefficients.	Cover images: Grayscale 512x512 dimension, Lena, Pepper Baboon, Barbara, Boat, House, Sailboat, Elaine, Tiffany, Gold hill, Toys and Zelda images.
Reddy, V. Lokeswara [5] /Bitmap	Canny edge detection method and matrix encoding for the Steganography technique.	Images of different dimensions, such as 32 × 32, 60 × 60, 64 × 64, 80 × 80 and 100 × 100.
Srinivasan et al. [6] /Bitmap	Algorithm designed for Android application like smart phone, tab or portable device using LSB technique.	Bitmaps as cover images, MMS (Multimedia Messaging Service) messages as data bits.
Huang et al. [23] /JPEG	A channel selection rule based on three parameters, perturbation error, quantization and magnitude of quantized DCT to be modified, was used to find DCT coefficients that may introduce low detectable distortion for data hiding.	5000 images converted into grayscale, cropped into size 512× 512, compressed by JPEG5.
Holub et al. [24] /JPEG	Universal design for distortion called UNIWARD (universal wavelet relative distortion) that can be applied for embedding in arbitrary domain.	BOSS base database, containing 10,000 8-bit grayscale images of dimension 512 × 512; embedded with fix payload.
Arshiya .T and Abdul Rahim [26] /JPEG	Reversible data hiding in encrypted image Steganography. Considered patch-level sparse representation for hiding data.	Lena, Airplane, Man, and Crowd transformed into gray-scale with dimension 512 × 512 and BOSS Base gray- scale images with dimension 512 × 512.
Zhang et al. [7] /JPEG	Compression resistant adaptive Steganography algorithm based on STC coding (Syndrome Trellis Coding) and distortion function.	Lena image (512 × 512).
Yang et al. [8] /JPEG	JPEG RDHC (Reversible Data Hiding Scheme) method to insert confidential data into zero coefficients in a zigzag sequence of 8 × 8 DCT blocks. They used and altered only AC coefficients block of sequence in the middle frequency	Six grayscale test images, with 512 × 512 dimension: Lena, Peppers, Airplane, Boat, Baboon, and Zelda. A series of pseudo random binary numbers were used as the secret data.
Hiney et al. [9] /JPEG	For Facebook images. Only the high resolutions images were used to hide some secret text. Only the high-resolution carriers were capable to successfully transfer image payloads.	Different JPEG images of different sizes and resolutions.
Wang, C. and Ni, J, [18] /JPEG	Used block entropy of DCT coefficients and STCs.	Uncompressed grayscale images from Core Draw database for embedding.
Fridrich Jiri [11] /PNG	Palette based Steganography method inserted only one bit in one pixel of its pointer to the Palette. They selected pixel randomly using seed and shared key and searched palette's closest color to insert a bit to embed.	"Mandrill" (baboon) image of dimension 512 × 512.
Zin, Wai [13] /PNG	Combining the LSB technique with RC4 algorithm and BBS (Blum Blum Shub) generator	One PNG image as a cover image and plain text as a secret message.
Chen et al. [14] /PNG	Used K-means algorithm for 'training the palette'. Euclidean distance is used to measure the dissimilarity between the pixels (vectors and clusters). The secret message get inserted into the true color value of palette in raster scan method from left to right, top to bottom.	Lena , Pepper, Baboon (dimension 512 × 512)

B. FREQUENCY DOMAIN (JPEG) STEGANOGRAPHY

JPEG image format algorithms work in the frequency transform domain (they work on the rate at which the pixel values are changing in the spatial domain). Frequency transform domain is divided into two categories: high-frequency domain (deals with edges) and low-frequency domain (deals with smooth and plane area). Changes in low frequencies can be more apparent. Both DCT (Discrete Cosine Transform) and DWT (Discrete Wavelet Transform) can be used to embed the secret data into the image coefficients. The frequency domain Steganography approaches were shown be more secured compared to the spatial domain methods [22].

The methods reported in [2], [7], [18], [21], [23] are based on DCT. These methods uses relationships between the

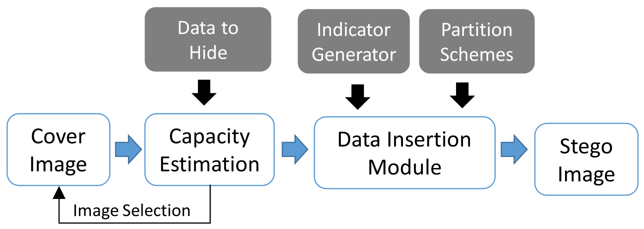


FIGURE 2. **Block diagram of the main phases in Generic Steganography Algorithm (GSA).**

DCT coefficients and STCs. Yang *et.al.* [8] proposed simple DCT method to insert confidential data into zero coefficients in a zigzag sequence of 8×8 DCT blocks. In [17], the method uses bit-plane encoding procedure multiple times and a redundancy evaluation approach to increase hiding capacity. The work proposed in [11] is based on Integer Wavelet Transform (IWT). A careful review of various image Steganography algorithms can reveal that algorithms for JPEG cover images performs better in security aspects compared to all other image formats [22].

C. PNG (PALETTE BASE/IMAGE DATA BASE) STEGANOGRAPHY

There are generally two approaches to embed data bits in PNG images: one can insert data bits into palettes or one can insert data bits into the image data [11]. The first method (palette-mode) is probably the easier to implement, but has less capacity to store data depending on the palette size. In this approach, it is difficult to store even one bit per image component, since it is easy to distinguish images with and without secret message. In contrast, the second method (image-mode) offers more capacity, although this approach has difficulty in ensuring security. It is possible to embed one bit, 2 bits, 3 bits, and up to 7 bits per pixel of image data without distorting the image. In our work, we will use image data based approach to insert data bits into PNG image to have increased capacity (explained later). As discussed later, we try to overcome the security issues by spreading data unevenly and using variable bits partition scheme. Works proposed in [11], [12], [14] are palette-mode methods; whereas [13] is an image-mode method.

III. ALGORITHMIC FORMULATIONS

In this section, we present the proposed algorithm in abstract mathematical terms. Specific implementations for different image types will be discussed later. Fig. 2 illustrates the main modules in the proposed method. A cover image passes through capacity pre-estimation procedure to select the best cover image for the secret data bits. Once a cover image is selected, concepts like partition schemes are used to store data in image components. In the following sections, each of these modules will be described in details.

A. BASIC DEFINITIONS

An image is a visual representation of an entity. We can consider an image I as a two dimensional matrix of finite number of elements or components. Let $C_{x,y}$ be a *component* of the image I, $x = 1, 2, \ldots, r$ and $y = 1, 2, \ldots, c$, where r and c denote the number of rows and columns in the matrix representation of I. For example, this $C_{x,y}$ can be an intensity value in a gray scale image, or it can represent color intensity values in some color image formats or can be some sort of coefficient values in other image formats.

Each of the $(r \times c)$ components $C_{x,y}$ consists of a number of parameters $P^i_{x,y}, i = 1, 2, \ldots, p$. For example, parameters $P^i_{x,y}$ could be the color component values in some image formats. In JPEG images, $C_{x,y}$ consists of coefficient values as parameters.

In our approach, we divide the image matrix I into a series of $n \times n$ blocks. Through experimentations, we found that $n = 8$ gave the best results, as blocks with size less than (8×8) does not have enough capacity for hiding data. On the other hand, blocks with size greater than (8×8) may sharply reduce the overall capacity of the cover image. Therefore, for $n = 8$, we can have a total of $((r \times c) / 64)$ blocks.

Now, we define a *triode* T as a group of three *elements*, where a triode element can be either a parameter in a component or a component itself. In the first case, we denote the triode T as T_P, whereas in the second case, T is denoted by T_C. Assume that the three elements of a triode T are denoted as e_1, e_2 and e_3. Our proposed method works at the triode level, either using T_P or T_C, based on the cover image format. The proposed method selects one triode at a time and makes one element in that triode as an *indicator* randomly (discussed next). Then secret data bits are stored in one of the two remaining elements in that triode based on element values. Fig. 3 illustrates the use of triodes for data hiding in image component blocks. Note that, not all possible triodes in a component block may be used for hiding data. For example, in our algorithm for JPEG cover images, only four triodes per 8×8 component block are used (Fig. 3(b)). In other case, 64 triodes per component block can be used using T_P. In general, our algorithm considers m number of different triodes in each component block, labelled as triode 1, triode 2, …, triode m.

In summary, Generic Stenography Algorithm (called *GSA*) hides data by altering some element values in each triode of each component block for a given cover of a particular format.

B. TRIODE INDICATORS

As discussed above, out of three elements e_1, e_2 and e_3 in a triode T, one of them is selected as *indicator*. The idea here is that only one of the triode elements will store data. This way, we can keep the distortion level low with increased security. To find out which triode element will store data, we identify one of the e_1, e_2 or e_3 as indicator. Say, e_{in} is the indicator element. This selection of e_{in} is done randomly using a random number generator (RNG) for security purpose. During data extraction, the receiver must know the exact e_{in} used in a triode at the sender side. For this reason, both sender and receiver must agree on a shared key for the RNG.

Now, e_{in} is used to *indicate* which of the other triode elements will store data. Suppose e_j and e_k are the two elements

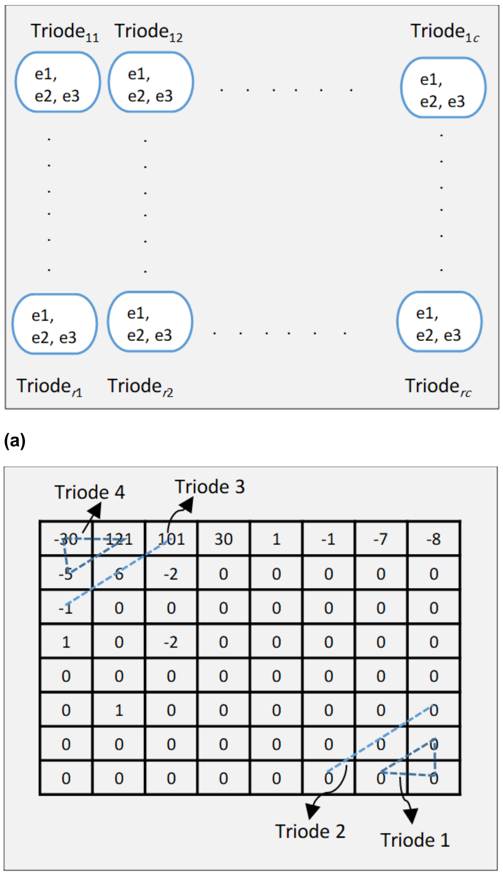


FIGURE 3. Illustration of triodes for embedding data in component block: (a) Triodes in an image of size (rxc) components, (b) triodes in an 8 × 8 JPEG coefficients block (with random coefficient values).

in T other than e_{in}. Data will be stored in e_j if the numeric value of e_j is lower than e_k. Otherwise, e_k will store the data. The number of bits to be stored in either e_j or e_k is decided using the concept of *partition schemes*, to be discussed next.

C. PARTITION SCHEMES

In general, a partition scheme is a mapping that decides the number of bits to be stored in a selected element of a triode in a cover image. For the discussion purpose, we assume that the element in a triode T that will store data is denoted by e_{data}.

We first define partition schemes in general terms. Generic Variable Bit Adaptive Partition Schemes (GVBAPS) can be defined as sequence of schemes (S_i), $i = 0, 1, \ldots, k$, where each (S_i) allows a triode element to store a maximum of $(n - k + i)$ bits of data, for some $k, n > 0$. For a particular element, only one scheme can be utilized based on the element value. Table 2 illustrates GVBAPS for $k = 7$ and $n = 8$.

The goal of a partition scheme is to allow *variable* number of bits to store in a triode element, based on the value of the element. Some partition schemes allow larger number bits to be stored in a triode element compared to other schemes.

TABLE 2. Generic variable bit adaptive partition schemes.

Partition Scheme (S_i)	Range of values in t_{data}		Numbers of bits per element (bpe)
(S_0)	(v_{i+7})	(v_{i+8})	$(n-7)$
(S_1)	(v_{i+6})	(v_{i+7})	$(n-6)$
(S_2)	(v_{i+5})	(v_{i+6})	$(n-5)$
(S_3)	(v_{i+4})	(v_{i+5})	$(n-4)$
(S_4)	(v_{i+3})	(v_{i+4})	$(n-3)$
(S_5)	(v_{i+2})	(v_{i+3})	$(n-2)$
(S_6)	(v_{i+1})	(v_{i+2})	$(n-1)$
(S_7)	(v_i)	(v_{i+1})	(n)

The partition schemes are called *adaptive*, since the best partition scheme is adaptively selected for a given cover image I and some secret data D. To accomplish this, an estimation of the capacity of I is needed. Assume that the estimated capacity of I is I_{cap}. Now, we select a partition scheme (S_i) that can ensure that D can be stored in I with the *maximum* number of components of I being used (called *data spreading* in this paper). Therefore, we next discuss the estimation of capacity of a cover image and the concept of data spreading.

D. CAPACITY ESTIMATION AND DATA SPREADING

For the estimation of the capacity of a given cover image, our algorithm uses a value called *Maximum Constant*, M_C. The value M_C is estimated experimentally as follows. For a cover image I of a particular format (say JPEG) and for some random data bits, we insert d bits into each triode of a component block, starting with $d = 1$. Gradually the value of d is increased until the quality of the stego image gets badly affected (which roughly corresponds to a PSNR value less than 48 dB). Thus, M_C is estimated approximately as 12.5 for each block in JPEG images. With M_C more than 12.5, the PSNR of the stego image reduces significantly (for example, for 'Pepper' image, PSNR is 57.7dB at $M_C{\sim}12.5$, whereas PSNR goes down to 48.45dB at $M_C{\sim}14$). Then, the expected capacity of a I is: $E_{cap} = (r \times c \times M_c)/64$. Here, E_{cap} is the expected capacity of a cover image with $(r \times c)$ components and 8×8 component blocks.

If the amount of secret data is less and capacity of cover image under consideration is high, then instead of embedding secret data in less number of image components, we attempt to scatter it throughout the entire image components. This technique can help reduce the distortions in stego images. This goal is achieved by selecting the partition scheme that allows minimum amount of changes per triode element. To improve security further, our algorithm uses interleaved selection of blocks (alternate blocks) for storing data bits.

Now, let the total number of image components/parameters utilized to hide data with data spreading be C. Let X be the number of utilized components/parameters without data spreading. Then, $C - X$ is the amount of extra components/parameters utilized (called *ECU*), Hence %Extra Component/ parameter Utilised, $ECU = (C - X) \times 100\%$. We attempt to maximize *ECU* for given data bits and a cover image. This is done with the help of *Decision Factor*, defined as follows.

Decision factor D_f is defined as a ratio of the difference between the expected capacity of a cover image and secret data length in bits, to the total number of blocks in the cover image. Therefore, decision factor D_f = floor (Data difference / Total number of 8 ×8 blocks). Here, Data difference = Expected Capacity of Cover Image – Secret Data length and Total number of component blocks = $((r \times c)/64) \times s$, where s is a constant value (1 or 1.5) depending on the image format. Therefore, a lower decision factor value means that the capacity of the cover image is close to the length of the secret data and thus less amount of data spreading is possible.

E. EMBEDDING EXAMPLE

Table 3 illustrates an example for embedding data in the parameter/component using Generic Variable Bit Adaptive Partition Schemes (GVBAPS). Say, x, y, and z denote three parameters/components (i.e. elements) with values 154, 152, and 140 respectively in any one of the triodes. For example, these three parameters can be the RGB values in a pixel or three DCT coefficients. The values of the elements are selected randomly and the following steps are applicable for any values of the elements. To simplify the process, parameters/components are sequenced in a cyclic order, like $x \rightarrow y \rightarrow z \rightarrow x$. In the following, the 'steps' refer to the step numbers mentioned in Table 3.

In Step 1, element 'z' is randomly selected as an *indicator*, which means 'z' will not store any data. This selection of an indicator can be done using a random number generator. Now data bits are stored in one of the element other than the indicator. The selection of element that will store data bits is done in Step 2.

In Step 2, since the value of 'y' is lower than 'x', therefore 'y' is selected to store data. Once we select the element that will store data, we need to decide the number of bits that will be stored in that element. In Step 3, the number of bits to store is found based on the partition scheme and values of the parameters/components. For ease of understanding, parameter/component values in Table 3 are shown in binary. In Step 4, three secret data bits (as calculated by the partition scheme) are inserted in the lower three bits of 'y'. Now, the value of 'y' changes from 10011000 to 10011111. In Step 5, after changing the bits, the value of element 'y' changes from decimal 152 to 159.

As for the detection/retrieval phase, the following points need to be considered. Once we embed data bits in an element, the value of the element may change in a way that the rule used in Step 2 may no longer be valid. In the current example, the new value of 'y' will make the receiver confused, as now the element 'x' is lower in value compared to the new value of 'y'. Thus, in case the value of 'y' becomes greater than 'x' after modification, it will be impossible to retrieve data by the receiver using the rule of Step 2 alone. Therefore, to retrieve the correct data, LSB of 'x' may need to be modified so as to correctly find (at the receiver's side) the parameter/component that stored data. The following rule is used for this purpose.

Suppose, a and b are the two elements other than the indicator element. Also, suppose b stores the data. In the example here, a is the element x and b is the element y. Now, if b comes AFTER a in the cyclic order of the parameters/components, then LSB of a will be modified so that the LSBs of a and b (after embedding) do not match. Conversely, if b comes BEFORE a in the cyclic order of parameter/component, then LSB of a will be modified so that the LSBs of a and b (after embedding) are same (either both 0 and both 1). Thus, at the receiver, only the LSBs of a and b are checked to decide which parameter/component has been used to store the secret data. Referring to Table 3, step 6, the LSB of 'x' is not changed in this case, as 'y' comes after 'x' in cyclic order and the LSBs of 'x' and 'y' (after embedding) are already different.

TABLE 3. An example of storing data in a triode element using adaptive partition schemes.

Step	Action	Element 'x'	Element 'y'	Element 'z'
0	Element values from triode	154	152	140
1	Element 'z' is indicator	154	152	140
2	'y' is selected to store data	154	152	140
3	Select partition scheme (values in binary)	10011010	10011000	140
4	Insert payload data (here 3 bits with values = 111_2)	10110010	10011111	140
5	Equivalent decimal	154	159	140
6	No change in LSB of 'x' in this case	10110010	10011111	140

F. PSEUDO-CODE

Based on the discussions above, the following GSA algorithm captures the process of our proposed Generic Steganography Algorithm (GSA).

IV. IMPLEMENTATION OF PARTITION SCHEMES

Before we describe the specific adaptations of GSA for multiple image formats, we discuss how partition schemes can be devised for different types of cover images.

A. PARTITION SCHEMES FOR JPEG IMAGES

GSA is mapped to JPEG cover images by considering the DCT coefficients as image components. Based on this mapping, we can devise partition schemes for JPEG images. For JPEG images, we call GVBAPS as JPEG Variable Bit Adaptive Partition Schemes or JVBAPS. In this paper,

Algorithm GSA

```
    Input: Cover Image, Secret Data, Shared key
    Output: Stego Image
1   Begin
2     Calculate cover image and data sizes in bits.
3     Estimate the capacity of Cover Image $E_{cap}$
4     Repeat for each $8 \times 8$ interleaved component block and interleaved row from cover image $I$
5       Repeat for each Triode $j$ where $j = 1$ to $v$ ($v = 4$ or $64$)
6         Assume that triode elements are $e_1$, $e_2$, and $e_3$.
7         Use the shared key to find indicator element (say $e_{in}$). Let the other two elements be $e_j$ and $e_k$.
8         Let, $\{q, r\} \in \{e_j, e_k\}$, where value $(q) = \min(\text{value}(e_j), \text{value}(e_k))$ and value $(r) = \max(\text{value}(e_j), \text{value}(e_k))$.
9         Calculate the number of bits $n$ to be stored in $q$ through partition scheme.
10          Hide data in $q$ by replacing the lower $n$ bits of $q$ by the $n$ bits from data.
11          If value $(q) >$ value $(r)$, adjust the last bit of $r$.
12          Save new values of $e_1$, $e_2$, and $e_3$ back into $8 \times 8$ component block.
13          Adjust unmodified elements
14    End
```

TABLE 4. Decision factors and bits per component for JVBAPSP1.

Decision factor range D_f	Partition Schemes (S_i)	Bits per Component (bpc)			
		Triode 1	Triode 2	Triode 3	Total
8	(S_1)	0	0	1	1
7	(S_2)	0	1	1	2
6	(S_3)	1	1	1	3
5	(S_4)	1	1	2	4
4	(S_5)	1	2	2	5
3	(S_6)	2	2	2	6
2	(S_7)	2	2	3	7
1	(S_8)	2	3	3	8
0	$(S_{Default})$	3	3	3	9

TABLE 5. Ranges of DCT coefficients values for JVBAPSP2.

Partition Schemes (S_i)	Range of coefficient values $[Cv_i, Cv_{i+1}]$	Numbers of bpc
(S_0)	[1024, 299]	0
(S_1)	[300, 000]	1
(S_2)	[-001, -099]	2
(S_3)	[-100, -199]	3
(S_4)	[-200, -399]	4
(S_5)	[-400, -499]	5
(S_6)	[-500, -599]	6
(S_7)	[-600, -1023]	7

JVBAPS are of two types: Variable Bit Adaptive Partition Scheme Part 1 (JVBAPSP1) for first three triodes and Variable Bit Adaptive Partition Scheme Part 2 (JVBAPSP2) for fourth triode (ref. Fig. 3(b)). JVBAPSP1 is mathematically defined as a sequence (S_i), $i = 1, \ldots, k$, for the decision factor D_f. By default $D_f = 0$ and partition scheme P_{default} stores $k+1$ bits. A partition scheme (S_i) allows k bits to store per component with $D_f = (k - i + 1)$. Table 4 illustrates JVBAPSP1 for $k = 8$.

As can be seen in Table 4, the number of bits to be stored in the first three triode coefficients of a JPEG image depends on the decision factor value. Decision factor values allow different number of bits to be stored in selected coefficients in each triode. This approach may give more utilization of coefficients of the cover image for entire secret data. This coefficient utilization would be most under maximum estimated capacity of the cover image. In all these cases, our algorithm allows spreading of data over the cover image.

JVBAPSP2 (shown in Table 5) is used for hiding data in the fourth triode of a JPEG cover image. JVBAPSP2 is defined as a sequence (S_i), $i = 0, \ldots, k-1$, for the coefficient value C_v. The scheme (S_i) allows to store i number of bits in the image coefficient when C_v meets the certain condition. Table 5 shows an example of JVBAPSP2 for $k = 8$. For example, as shown in Table 5, partition scheme (S_5) is selected if the coefficient value C_v lies between -400 to -499, allowing 5 bpc from the secret data to be stored in a component. Triode 4 generally has high-valued (mostly negative) DCT coefficients. Based on observations, we found that we could store more bits in more negative valued DCT coefficients without compromising quality, leading to the design of JVBAPSP2 in Table 5.

JVBAPSP2 allows hiding more bits in negatively valued coefficients. As the value of a coefficient goes more negative, JVBAPSP2 allows increasing number of bits to be stored in that coefficient. In contrast, storing more number of bits into the coefficients with positive values can give rise to variation in brightness or color or produce blur effect or dots in the stego image. Positive coefficients can only store a maximum of one bit of data per component.

B. PARTITION SCHEMES FOR RGB, PNG AND TIFF

In Variable Bit Adaptive Partition Scheme (VBAPS) for RGB, PNG and TIFF images, we want to store variable number of bits according to the color intensity values of the channels. Channels with high values will store less number

TABLE 6. Ranges of channel intensity values for VBAPS for RGB, PNG, and TIFF images.

Partition Schemes (S_i)	Channel Intensity value range $[Cv_i, Cv_{i+1}]$	Numbers of bpp
(S_0)	[228, 255]	1
(S_1)	[196, 227]	1
(S_2)	[164, 195]	2
(S_3)	[128, 163]	2
(S_4)	[096, 127]	3
(S_5)	[064, 095]	3
(S_6)	[032, 063]	4
(S_7)	[000, 031]	5

of bits, whereas, channels with low values will store more number of bits using partition schemes. Thus, the selection of a partition scheme depends on the intensity values of a cover image. The capacity of a cover image depends on the color intensity values of the channels being used to store data. Our partition scheme allows different number of data bits to be stored per channel based on the channel intensity values (for example, one color intensity level may allow to store 1 bits per parameter/channel (bpp) whereas, another color intensity level may allow 5 bpp).

The partition scheme decides the number of bits to be stored throughout the cover image depending on the requirements and allowable level of distortion. For example, if secret data bits are less than the estimated capacity of a cover image, only one bit or two bits per channel is used over the cover image. If the number of secret data bits to be stored is high (say, nearly equal to the estimated capacity of the cover image), then more number of bits (may be 3/4/5 bits) will be stored in each channel depending on the size of data and partition scheme. Table 6 illustrates one such VBAPS for RGB/PNG/TIFF images.

V. ADAPTATIONS FOR MULTIPLE IMAGE FORMATS

In this section, we show how GSA algorithm presented in Section III can be readily adapted for some popular cover image formats, namely JPEG, RGB, PNG and TIFF.

A. GSA FOR JPEG IMAGES

Every JPEG image consists of a header. Each header has a field named *coef_arrays* of size 1×3 cells. The 1st cell is of size $r \times c$ where r and c are the row and column dimensions of the image under consideration. The 2nd and 3rd cells are of size $r/2 \times c/2$. Hence, in the formulation of decision factor constant, $s = 3/2$ for JPEG format (as discussed in Section III). Our algorithm selects $n \times n$ coefficients from coefficient matrices of a JPEG cover image to be processed once at a time, where $n = 8$ (ref. Fig. 3(b)). Therefore, in JPEG cover images, image components refer to the coefficient values and a triode refers to a set of three coefficients. For each component block (which is an 8×8 block of coefficients), four triodes are considered (ref. Fig. 3(b)), named as triode 1, triode 2, triode 3 and triode 4. In each triode, we have three coefficients (Fig. 3(b)). For Capacity Estimation of Cover Image E_{cap}, we found the value of M_C as 12.5 bits per block. The following *GSA_JPEG* algorithm is the implementation of GSA for JPEG images.

B. GSA FOR BITMAP (RGB/PNG/TIFF) IMAGES

For Bitmap cover images, each component corresponds to an image pixel, with each pixel having three parameters: R, G and B. Therefore, an 8×8 component block contains a total of 64 pixels. Thus, each triode in this component block contains the parameters R, G, B; making a triode same as a pixel for a Bitmap image. For Capacity Estimation of Cover Image E_{cap}, we found the optimum value of M_C to be 192 bits per component block.

GSA for PNG (Portable Network Graphics) cover images are almost same as for RGB images. PNG is palettes based file format that supports indexed color, gray scale and RGB images. It supports palette-based images of 24 bit RGB or 32 bit RGB colors, grayscale images, and full-color non palette based RGB images. Again, $M_C = 192$ for PNG cover images.

Similarly, TIFF (Tagged Image File Format) supports bi-level, grayscale, palette-color, and RGB full-color images. TIFF images can store 16 to 48 bit images and meta data. TIFF contains property of loss-less compression, uncompressed option, Grayscale, RGB color, 8 to 24-bit color and indexed color. Our proposed method uses the RGB color mode property of TIFF image and store secret bits as like method [2]. Our proposed method does not support TIFF 48 bit RGB data. Again, $M_C = 192$ for TIFF cover images.

The following *GSA_RPT* algorithm illustrates the steps for GSA for Bitmap/PNG/TIFF cover images.

VI. EXPERIMENTAL RESULTS

We have tested the proposed algorithm (GSA) for around 100 different cover images of multiple formats. We first present some results of testing GSA for various cover images with different secret data. We then discuss Steganalysis results for the proposed method. Finally, we compare the proposed method with other reported methods for three types of cover images using some popular cover images used by other researchers.

A. GSA FOR JPEG/RGB/PNG/TIFF COVER IMAGES

We have tested our Generic Steganography Algorithm (GSA) for a number of color cover images of JPEG/BMP/PNG/TIFF formats. The output stego images for three cover images (each of them in four different formats) are shown in Fig. 4. For experimentations, we have taken the 'soldier' image (contains 35,160 bits) as a secret data (ref. Fig. 4). The detailed results for different cover images are given in Table 7. A comparative graph is shown in Fig. 5.

The data hiding capacity of a colored cover image depends on its dimension, colors, edges and fine details. For the same data length and cover image size, the PSNR values for Lena,

Algorithm GSA_JPEG

```
     Input: Cover Image, Secret Data, Shared key
     Output: Stego Image
1   Begin
2     Calculate cover image and data sizes in bits.
3     Estimate the capacity of Cover Image $E_{cap}$ and Calculate Decision Factor $D_f$
4     Repeat for each 8 × 8 interleave component block and interleaved row from image $I$.
5       Repeat for each Triode $j$ where $j = 1$ to 4.
6         Let $p1, p2$, and $p3$ be the parameters (coefficients) in triode $j$
7         Use the shared key to find indicator coefficient (say $p1$).
8         Let, $\{q, t\} \in \{p2, p3\}$, where value $(q) = \min$ (value $(p2)$, value $(p3)$) and value $(t) = \max$ (value $(p2)$, value $(p3)$).
9         Calculate the number of bits $n$ to be stored in $q$ through $VBAPSP2$ ($j = 4$) or $VBAPSP1$ ($j = 1, 2, 3$).
10          Hide data in $q$ by replacing the lower $n$ bits of $q$ by the $n$ bits from data.
11          If value $(q) >$ value $(t)$, adjust the last bit of $t$.
12          Save new values of $p1, p2$ and $p3$ back into the coefficients and save 8 × 8 coefficient block
13          Adjust unmodified elements $ei$.
14    End
```

Algorithm GSA_RPT

```
     Input: Cover Image, Secret Data, Shared key
     Output: Stego Image
1   Begin
2     Calculate cover image and data sizes in bits.
3     Estimate the capacity of Cover Image $E_{cap}$
4     Repeat for each 8 × 8 interleave component block and interleaved row from image $I$.
5       Repeat for each Triode (a pixel) $j, j = 1$ to 64
6         Let $p1, p2$, and $p3$ be the parameters of a component (RGB values of pixel $j$)
7         Use the shared key to find indicator coefficient (say $p1$).
8         Let, $\{q, t\} \in \{p2, p3\}$, where value $(q) = \min$ (value $(p2)$, value $(p3)$) and value $(t) = \max$ (value $(p2)$, value $(p3)$).
9         Calculate the number of bits $n$ to be stored in $q$ through partition scheme.
10          Hide data in $q$ by replacing the lower $n$ bits of $q$ by the $n$ bits from data.
11          If value $(q) >$ value $(t)$, adjust the last bit of $t$.
12          Save new values of $p1, p2$ and $p3$ back into component (pixel) and save 8 × 8 pixel block.
13          Adjust unmodified elements $ei$.
14    End
```

Pepper and Baboon images are slightly different based on color or other parameters (ref. to Table 7).

As can be seen in Fig. 5, the difference between average PSNRs of JPEG cover images with any other format is nearly13 dB. A possible reason behind this difference is that in JPEG cover images, the DCT coefficients cannot be altered much. However, the capacity utilization (% component/parameter utilization) being same due to the same size of the cover images.

For the cover image number four ('doll' in Table 7), for the same data length, utilization of parameters reaches 88.4%. This high utilization of parameters can be explained as follows. The estimated capacity of the 'doll' image is 120,000 bits. In this case, our algorithm selects to maximize the data spreading throughout the image; hence, it selects to store 1 bit/component with VBAPS partition scheme to be set for only 1 bit. In this way, we get % Extra Component/parameter Utilized as given previously as: $ECU = 88.4 - 28.5$. Here, value of X is found by experimentation with VBAPS. With $ECU = 59.86$, we achieve a high level of data spreading. Similarly, 96.65% and 97.14% utilizations of parameters are achieved (with data length of 4,72,928 bits) for cover images 'Earth' and 'Cake' respectively, as shown in Table 7 (serial number 10 and 11).

However, in 'Birds' image (serial number 7 in Table 7), for the data length of 7,57,953 bits, the utilization of parameters is 65.27%. This is because, for this particular cover image and given data length, the maximum level for data spreading (which is 1 bit/component) was not possible.

B. ROBUSTNESS AGAINST STEGANALYSIS

In this section, we discuss the security aspects of the proposed method based on Steganalysis. In addition, we present ideas to handle a few of the possible attacks.

Seganalysis detects the stego image when it finds some similarities or irregularities or heavily loaded areas in the

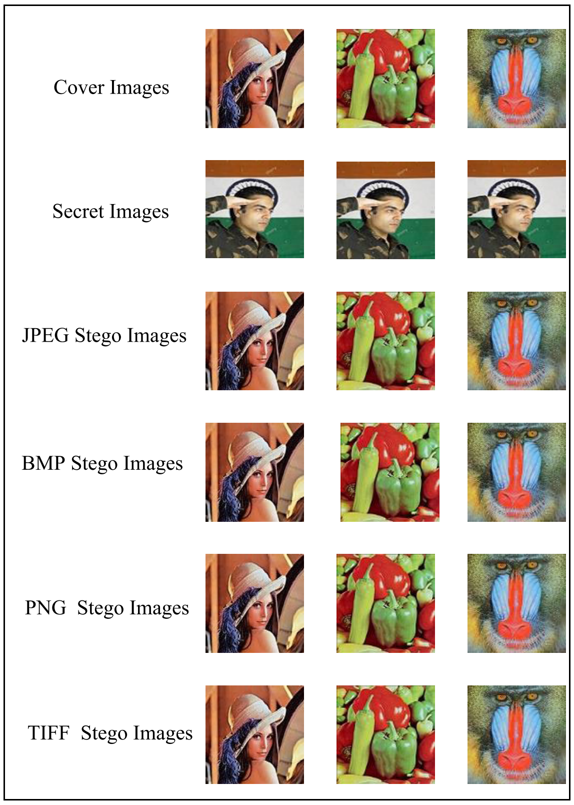


FIGURE 4. Stego output results of the proposed GSA using the JPEG/BMP/PNG/TIFF cover images (Lena, Pepper, Baboon of size (512 × 512)).

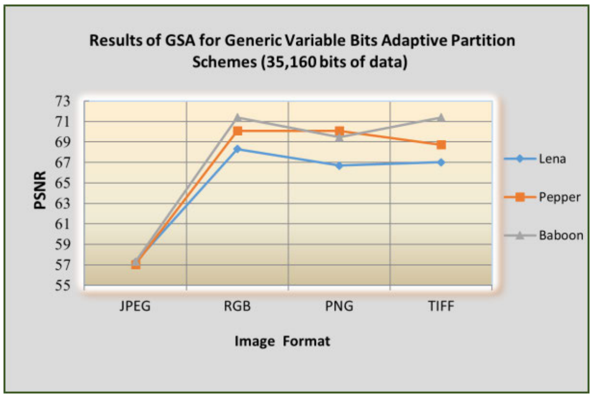


FIGURE 5. Illustrations of PSNR values for GSA for JPEG/RGB/PNG/TIFF formats for 35,160 bits of secret data.

image. To achieve robustness against Steganalysis, the stego image should not have any bulky area of similarities or should not have irregularities which can be recognized by Steganalysis. In our algorithm GSA, data embedding is based on values of image elements; for example, DCT elements values ranging from -1023 to 1024, whereas bitmap image pixel values range from 0 to 255 (8 bits/pixel). We worked directly on color images without any pre-processing steps (like color conversion). Let e_i denote the number of data bits embedded in an element i. In GSA, the average secret data length per element is estimated by experimentation and

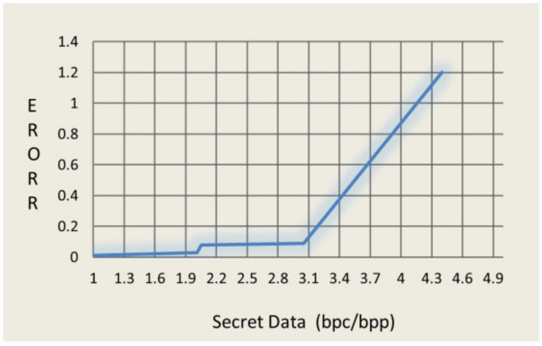


FIGURE 6. Error indices across payload size (bits per element).

found to be around 3.1 bits. To test robustness of the proposed method, we vary the value of e_i and test the robustness against Steganalysis attack.

In our experimentations, for $e_i = 0.1$ to 3.1, error value $ER < 1$. After embedding high amount of secret data bits/element with $e_i > 3.1$, ER becomes more than one. Therefore, the quality of the stego image starts to reduce when more than 3.1 bpc / bpp are used. For e_i less than 3.1 bpc / bpp, error less than one can be tolerated and does not affect the quality of the image much. The error ER increases more after embedding more secret data bits in more positive element values as shown in Figure 6. Since the attacks are more effective on high frequency elements, so we tested our modified high frequency elements against the attack and fixed the maximum number of bits/element to be 3.1 bpc/bpp. Embedding was done in interleaved blocks for the selected triode in an element. To compensate the effect of this modification, we used the technique of counter embedding bits in other remaining elements. For ease of discussions, we refer to the elements selected by GSA as 'modified' elements and the other elements as 'unmodified' elements. Suppose, we embed α bits in each unmodified element to balance the weight of elements so that the Steganalysis does not recognize the difference between the modified and unmodified elements.

Then we compared the unmodified elements error value ER with modified element error value, AER. We have tried to keep the balance between the modified and unmodified elements to achieve the balanced error. The error value of modified elements should be nearly equal to the error value of unmodified elements so that the Steganalysis will not be able to detect the difference between modified and unmodified elements; that means the error for the α bits AER should be $\sim = ER$.

We also tested the robustness of the proposed algorithm against image processing attacks. For this purpose, we used format independent rich Steganalysis model [27]. Rich model works on the coefficients/pixel perceptual weight difference error and the distortion error at different level of sensitivity. For secret data length/element $e_i < 3.1$ bpc/bpp, the Steganalyzer found $ER < 0.5$. For, $e_i < 0.5$, we got $ER < 0.25$ as shown in Figure 6. It means that our generic algorithm is robust against Steganalysis with $e_i < 3.1$. We also tested the

TABLE 7. GSA outputs using generic variable bits adaptive partition schemes for different cover image formats/sizes with various secret data size.

Serial No.	Cover Image Size	Data Length Size	JPEG		BMP		PNG		TIFF	
	Image (R×C)	Bits	% Comp. utilization	PSNR (dB)	% Param. utilization	PSNR (dB)	% Param. utilization	PSNR (dB)	% Param. utilization	PSNR (dB)
1	Lena (512×512)	35160	13.41	57.2579	13.60	68.30	13.60	66.67	13.60	67.02
2	Pepper (512×512)	35160	13.41	57.0316	13.60	70.09	13.60	67.16	13.60	68.71
3	Baboon (512×512)	35160	13.41	57.2975	13.60	71.40	13.60	69.45	13.60	71.40
4	Doll (200×200)	55135	13.46	55.98	36.77	64.62	36.77	63.51	36.77	64.12
5	Football (800×800)	55135	12.54	59.67	23.56	63.20	23.56	62.90	23.56	63.09
6	Garden (750×750)	44832	11.13	59.94	26.8	63.62	26.8	62.82	26.8	62.91
7	Birds (560×600)	48504	13.66	56.10	43.05	63.09	43.01	62.09	43.05	63.01
8	Cat (450×450)	50134	13.24	57.92	74.29	58.47	74.29	57.89	74.29	58.08
9	Flower cake (653×621)	36020	14,111	55.77	30.90	66.28	30.90	61.93	30.90	63.32
10	Earth (700×700)	45832	12.57	59.76	26.87	68.88	26.87	67.02	26.87	68.62
11	Cake (650×750)	50134	13.53	57.32	30.90	56.37	30.90	55.01	30.90	56.05
12	Baby (594×428)	23000	12.59	62.08	35.23	67.01	35.23	66.87	35.23	66.91
13	Dairy milk (720 ×720)	50672	12.135	56.78	59.23	61.08	59.23	59.10	59.23	60.10
14	Rose (540 ×771)	50136	13.06	55.95	36.15	65.33	36.15	63.81	36.15	65.01
15	Fruit plate (640 ×640)	50152	13.46	56.12	36.77	64.98	36.77	62.19	36.77	64.02

JPEG and TIFF stego images generated in our experimentations using Ben-4D specific Steganalysis tool. The results are very promising for JPEG and TIFF images as shown in Table 8.

In the following, we suggest further modifications to GSA to counter some other attacks. Other common attacks on stego images include image processing attacks, like image cropping attack, orientation (rotation) attack or clipping attack, etc. Clustering and duplication are two approaches that can be applied for each image element to get rid of orientation attack (rotation attack) and image cropping attack. Duplication of information in the image can be a good solution to tackle the cropping or cutting attack on the image. Although this is a costly approach, we can divide the cover image into four quadrants and then replicate the secret data four times in each quadrant of the cover image. Another solution against cropping attack is to embed the secret data in the 'central' area of image so that if any attacker crops the corner sides of image, secret data can be still available in the center. Also we can combine these two methods of clustering and duplication to make our algorithm GSA more secure.

It is challenging to achieve three aspects of Steganography simultaneously, such as secret data embedding capacity, perceptibility of the stego image and its robustness against Steganalysis or attack. GSA is a step towards this goal, as it provides high perceptibility, high PSNR after embedding, high capacity secret data bits and promising results against Steganalysis.

C. COMPARATIVE RESULTS

Comparative results for PSNR values of some existing works with the proposed algorithm (GSA) are illustrated in Table 9 (JPEG cover images), Table 10 (Bitmap cover images) and Table 11 (PNG cover images). In all of these comparative results, we have used Lena, Paper and Baboon as cover

TABLE 8. Steganalysis results using Ben-4D Steganalysis tool.

File type	No. of images tested	Subsample match results
JPEG	20	No. Editor doesn't appear in list above
TIFF	20	No change detected

TABLE 9. Comparative results measuring performance in terms of PSNR (JPEG images).

Methods	Images (512 x 512)		
	Lena	Pepper	Baboon
Proposed GSA Method	57.25	57.03	57.29
RDHS, results taken from [8]	47.27	44.42	31.5
Chang, results taken from [8]	35.95	41.41	40.49
Hemlata [21]	44.6	44.7	44.8
Comp embedding, results taken from [28]	34.67	34.75	37.64
Jseteg, results taken from [28]	35.36	35.45	38.82
Out Quesses, results taken from [28]	36.37	35.32	38.22
Adaptive PVD [29]	50.89	51.29	52.29
GANs [31]	28.233	28.665	28.29
Adaptive PVD [32]	46.17	47.06	48.49
Average % Improvements	**30.23%**	**29.26%**	**30.08%**

images in different formats (JPEG/ BMP/ PNG) with the same dimensions and for same data length.

Table 9 shows the comparative results with nine other reported works. The proposed method shows 26% - 31% average improvements in terms of PNSR values for three cases. A closer look at the numbers in Table 9 reveals that, out of nine works, only one work [29], [30] reported PSNR values greater than 50 for all three images. Even for these 'best previous results', our method provides 9% - 11% improvements for all three images.

Similar comments can be made from Table 10, where comparisons are made with methods that work for Bitmap cover images. Here again, nine other methods (different than the methods in Table 9) are compared with the proposed method. However, as compared with Table 9, none of the previous nine methods gives uniformly the best results for all three images. Even if we take the best previous result for each image separately, the proposed method provides higher PNSR values for each of the three cover images. Likewise, Table 11 illustrates the comparative results with five other methods for PNG cover images. In all cases, the PSNR values produced by GSA are better compared with the other methods.

All three tables (Tables 9 – 11) combined, we have actually compared the proposed method with 23 other reported works! It should be noted that, each of the previous methods in these tables focuses on a single cover image type, thus optimizing

TABLE 10. Comparative results measuring performance in terms of PSNR (BITMAP images).

Methods	Images (512 x 512)		
	Lena	Pepper	Baboon
Proposed GSA method	68.30	70.09	70.40
Channel Selection [2]	51.09	51.42	51.15
Adaptive Partition [1]	46.94	49.22	46.74
Kareem's [19]	42.62	17.39	48.55
HIS Colour model [20]	42.63	62.96	61.87
Nadeem's stego [33]	56.12	58.21	57.26
LSB Matching [34]	54.53	54.48	54.15
Secure MSB [35]	48.00	54.64	61.79
Chaotic map [36]	44.28	44.30	44.29
Shen & Huang [37]	42.46	40.15	38.88
Average % Improvements	30.3%	31.4%	26.7%

TABLE 11. Comparative results measuring performance in terms of PSNR (PNG images).

Methods	Images (512 x 512)		
	Lena	Pepper	Baboon / Fruit
Proposed GSA method	69.45	69.09	69.45
Shamir's Threshold [38]	50.11	55.33	53.65
Rajoli [10]	51.3	51.8	59
Yung, results taken from [14]	36.95	35.86	34.09
EZ_Stego, results taken from [14]	14.23	14.55	21.68
Fridrich [11]	31.28	0.64	25.98
Average % Improvements	47.0%	54.2%	44.0%

their parameters for the target image type. This means, each method needs a particular image type for data hiding; a lack of cover image of such image type requires either to convert the cover image (which is not feasible sometimes, due to quality issues) or to use another algorithm. In contrast, the proposed method can utilize any types of cover images that are available.

VII. CONCLUSION

The proposed algorithm (called GSA in this paper) features the use of different types of images as cover media by utilizing an algorithm that depends of the abstract definition of image components. In addition, we used concepts like capacity pre-estimation, partition schemes and data spreading to embed high amount of secret data. The proposed method can be further enhanced by using experimentally estimated partition schemes.

When compared with other reported works based on PSNR values, the proposed method achieved at least 26% improvements. This result is very encouraging, although only the most common cover images were used. However, the proposed method provides other benefits that are possible only due its nature. For example, to avoid Steganalysis, different parts of the data can be sent using different image types. Also, in case of necessity, larger cover images can be used to spread the data adaptively.

To the best of our knowledge, the proposed algorithm is the first Steganography algorithm that can utilize cover images of type JPEG, Bitmap, TIFF and PNG. This way, the applicability of the proposed method is much wider that other reported works.

ACKNOWLEDGMENT

Dr. Arshiya S. Ansari would like to thank Deanship of Scientific Research at Majmaah University for supporting this work under the Project No. R-1441-117.

REFERENCES

[1] M. T. Parvez and A. A.-A. Gutub, "Vibrant color image steganography using channel differences and secret data distribution," *Kuwait J. Sci. Eng.*, vol. 38, no. 1B, pp. 127–142, 2011.

[2] R. Amirtharajan, S. K. Behera, M. A. Swarup, M. Ashfaaq, and J. B. B. Rayappan, "Colour guided colour image steganography," 2010, *arXiv:1010.4007*. [Online]. Available: http://arxiv.org/abs/1010.4007

[3] R. K. Singh and B. Lall, "Saliency map based image steganography," in *Proc. 28th Int. Conf. Image Vis. Comput. New Zealand (IVCNZ)*, Nov. 2013, pp. 430–435.

[4] S. Dogan, "A new approach for data hiding based on pixel pairs and chaotic map," *Int. J. Comput. Netw. Inf. Secur.*, vol. 10, no. 1, pp. 1–9, Jan. 2018.

[5] V. L. Reddy, "Novel chaos based steganography for images using matrix encoding and catmapping techniques," *Inf. Secur. Comput. Fraud*, vol. 3, no. 1, pp. 8–14, 2015.

[6] A. Srinivasan, J. Wu, and J. Shi, "Android-stego: A novel service provider imperceptible MMS steganography technique robust to message loss," in *Proc. 8th Int. Conf. Mobile Multimedia Commun.*, 2015, pp. 205–212.

[7] Y. Zhang, X. Luo, C. Yang, D. Ye, and F. Liu, "A JPEG comparison resistant adaptive steganography based on relative relationship between DCT coefficients," in *Proc. 10th Int. Conf. Availability, Rel. Secur.*, 2015, pp. 461–466.

[8] C.-N. Yang, C. Kim, and Y.-H. Lo, "Adaptive real-time reversible data hiding for JPEG images," *J. Real-Time Image Process.*, vol. 14, no. 1, pp. 147–157, Jan. 2018.

[9] J. Hiney, T. Dakve, K. Szczypiorski, and K. Gaj, "Using facebook for image steganography," in *Proc. 10th Int. Conf. Availability, Rel. Secur.*, Aug. 2015, pp. 442–447.

[10] S. Rojali and A. Galih, "Website-based PNG image steganography using the modified vigenere cipher, least significant bit, and dictionary based compression methods," in *Proc. Amer. Inst. Phys. Conf. Ser.*, vol. 1867, no. 2, pp. 020059-1–020059-11, 2017.

[11] J. Fridrich, "A new Steganographic method for palette-based images," in *Proc. PICS*, Apr. 1999, pp. 285–289.

[12] P. Sujitha and G. Murali, "Authentication of gray scale document images via the use of PNG image with data repairing," *Int. J. Sci. Res.*, vol. 2, no. 11, pp. 2319–7064, 2013.

[13] W. Zin, "Message embedding in PNG file using LSB steganographic technique," *Int. J. Sci. Res.*, vol. 2, no. 1, pp. 226–230, Nov. 2013.

[14] Y.-F. Chen, S.-W. Chien, and H.-H. Lin, "True color image steganography using palette and minimum spanning tree," in *Proc. Int. Conf. Math. Comput. Sci. Eng. (WSEAS)*, no. 3, X. Lifent Ed. Singapore: World Scientific, and Engineering Academy and Society, 2009.

[15] T. Taburet, P. Bas, J. Fridrich, and W. Sawaya, "Computing dependencies between DCT coefficients for natural steganography in JPEG domain," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur.*, Jul. 2019, pp. 57–62.

[16] A. Febryan, T. W. Purboyo, and R. E. Saputra, "Analysis of steganography on TIFF image using spread spectrum and adaptive Method," *J. Eng. Appl. Sci.*, vol. 15, no. 2, pp. 373–379, 2020.

[17] T. Shahida and C. Sobin, "An efficient method for improving hiding capacity for JPEG2000 images," in *Proc. Int. Conf. Internet Comput. Inf. Commun.* New Delhi, India: Springer, 2014, pp. 159–168.

[18] C. Wang and J. Ni, "An efficient JPEG steganographic scheme based on the block entropy of DCT coefficients," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 1785–1788.

[19] M. K. Ramaiya, N. Hemrajani, and A. K. Saxena, "Security improvisation in image steganography using DES," in *Proc. 3rd IEEE Int. Advance Comput. Conf. (IACC)*, Feb. 2013, pp. 1094–1099.

[20] N. Grover and A. K. Mohapatra, "Digital image authentication model based on edge adaptive steganography," in *Proc. 2nd Int. Conf. Adv. Comput., Netw. Secur.*, Dec. 2013, pp. 238–242.

[21] U. D. Acharya and P. R. Kamath, "A secure and high capacity image steganography technique," *Int. J. Signal Image Process.*, vol. 4, p. 83, Feb. 2013. [Online]. Available: https://arxiv.org/abs/1304.3629

[22] P. Rai, S. Gurung, and M. K. Ghose, "Analysis of image steganography techniques," *Int. J. Comput. Appl.*, vol. 114, no. 1, pp. 11–17, 2015.

[23] F. Huang, J. Huang, and Y.-Q. Shi, "New channel selection rule for JPEG steganography," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 4, pp. 1181–1191, Aug. 2012.

[24] V. Holub, J. Fridrich, and T. Denemark, "Universal distortion function for steganography in an arbitrary domain," *EURASIP J. Inf. Secur.*, vol. 2014, no. 1, pp. 1–13, Dec. 2014.

[25] S. Al-Nofaie, M. Fattani, and A. Gutub, "Capacity improved arabic text steganography technique utilizing 'Kashida' with whitespaces," in *Proc. 3rd Int. Conf. Math. Sci. Comput. Eng. (ICMSCE)*, 2016, pp. 38–44.

[26] T. Arshiya and A. Rahim, "Encrypting images by patch-level sparse representation for high capacity reversible data hiding," *Int. J. Adv. Technol. Innov. Res.*, vol. 9, no. 1, pp. 1–8, 2017.

[27] G. R. Suryawanshi and S. N. Mali, "Universal steganalysis using IQM and multiclass discriminator for digital images," in *Proc. Int. Conf. Signal Process., Commun., Power Embedded Syst. (SCOPES)*, Oct. 2016, pp. 877–881.

[28] C.-L. Liu and S.-R. Liao, "High-performance JPEG steganography using complementary embedding strategy," *Pattern Recognit.*, vol. 41, no. 9, pp. 2945–2955, Sep. 2008.

[29] A. Pradhan, K. R. Sekhar, and G. Swain, "Adaptive PVD steganography using horizontal, vertical, and diagonal edges in six-pixel blocks," *Secur. Commun. Netw.*, vol. 2017, pp. 1–13, 2017.

[30] S. Rojali and A. G. George, "Website-based PNG image steganography using the modified Vigenere Cipher, least significant bit, and dictionary-based compression methods," *AIP Conf. Proc.*, vol. 1867, no. 1, Aug. 2017, Art. no. 020059.

[31] Z. Zhang, G. Fu, F. Di, C. Li, and J. Liu, "Generative reversible data hiding by image-to-image translation via GANs," *Secur. Commun. Netw.*, vol. 2019, Sep. 2019, Art. no. 4932782, doi: 10.1155/2019/4932782.

[32] G. Swain, "Adaptive pixel value differencing steganography using both vertical and horizontal edges," *Multimedia Tools Appl.*, vol. 75, no. 21, pp. 13541–13556, 2016.

[33] N. Akhtar, S. Khan, and P. Johri, "An improved inverted LSB image steganography," in *Proc. Int. Conf. Issue Challenges Intell. Comput. Techn. (ICICT)*, Feb. 2014, pp. 749–755.

[34] A. J. Umbarkar, P. R. Kamble, and A. V. Thakre, "Comparative study of edge based LSB matching steganography for color images," *ICTACT J. Image Video Process.*, vol. 6, no. 3, pp. 1185–1191, Feb. 2016.

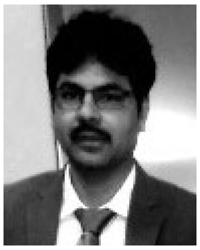
[35] A. Sharma, M. Poriye, and V. Kumar, "A secure steganography technique using MSB," *Int. J. Emerg. Res. Manage. Technol.*, vol. 6, no. 6, pp. 208–214, Jun. 2017.

[36] S. Dogan, "A new approach for data hiding based on pixel pairs and chaotic map," *IJ Comput. Netw. Inf. Secur.*, vol. 10, no. 1, pp. 1–9, 2018.

[37] S. Y. Shen and L. H. Huang, "A data hiding scheme using pixel value differencing and improving exploiting modification directions," *Comput. Secur.*, vol. 48, pp. 131–141, Feb. 2015.

[38] S. Tyagi, R. K. Dwivedi, and A. K. Saxena, "High capacity steganography protected using Shamir's threshold scheme and permutation framework," *Int. J. Innov. Technol. Exploring Eng.*, vol. 8, no. 9, pp. 784–795, Jul. 2019.

ARSHIYA S. ANSARI received the B.E. degree in computer technology from the Yashwantrao Chavan College of Engineering, Nagpur University, India, the M.Tech. degree in computer engineering from NMIMS University, Mumbai, India, and the Ph.D. degree from Noida International University, Noida, India. She has nine years of experience in teaching field. She is currently working as an Assistant Professor with Majmaah University, Saudi Arabia. Her research areas of interests are image processing and data warehousing. She is a Lifetime Member of ISTE.



MOHAMMAD S. MOHAMMADI received the B.E. degree in computer technology from the Yashwantrao Chavan College of Engineering, Nagpur University, India, the M.Tech. degree in computer engineering from NMIMS University, Mumbai, India. He is currently pursuing the Ph.D. degree with Noida International University, India. He has total 16.5 years of experience, including 1.5 years industrial experience in Reliance Petroleum, Mumbai, and 14 years of teaching experience. He is currently working as a Lecturer with the Computer Engineering Department, Qassim University, Saudi Arabia. His research interest includes image processing, information hiding, and information/network security. He is a member of the Saudi Internet Scientific Society from 2017 to 2018.

MOHAMMAD TANVIR PARVEZ received the B.Sc. and M.Sc. degrees in computer science and engineering (CSE) from the Bangladesh University of Engineering and Technology (BUET), Dhaka, and the Ph.D. degree in CSE from the King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, in 2010. He is currently working as an Associate Professor of computer engineering with Qassim University. His research interests include pattern recognition, image processing, and machine learning with the special interest in handwriting recognition using structural approach. He has received several awards including the Best Poster Award in ICFHR 2012. He maintains the research repository platform ideas2serve.net.