Information Hiding in RGB Images Using an Improved Matrix Pattern Approach

Amirfarhad Nilizadeh
Department of Computer Science,
University of Central Florida
Orlando, FL 32816 USA
af.nilizadeh@knights.ucf.edu

Wojciech Mazurczyk
Institute of Telecommunications,
Warsaw University of Technology
Warsaw, Poland
wmazurczyk@tele.pw.edu.pl

Cliff Zou
Department of Computer Science,
University of Central Florida
Orlando, FL 32816 USA
czou@cs.ucf.edu

Gary T. Leavens
Department of Computer Science,
University of Central Florida
Orlando, FL 32816 USA
leavens@cs.ucf.edu

Abstract

In this paper, a novel steganography algorithm based on an improved "Matrix Pattern" (MP) method is presented. In this process, firstly, an RGB image is divided into the non-overlapping square-sized blocks. Next, 95 dynamic-sized unique matrix patterns are automatically generated using the 4th and 5th bit layers of the green layer of each block, which are assigned to 95 English keyboard characters. Then, the blue layer of each block is used for embedding secret messages by adding matrix patterns which are assigned to the characters of the secret message. The results show that this algorithm has a high resistance against steganalysis attacks, including Regular Singular (RS), Sample Pair (SP), and PVD based attacks. Furthermore, the proposed algorithm not only improves capacity by over 27% when compared to the existing method, but also results in a slightly better transparency of the stego-image.

1. Introduction

Transferring a secret message in an unsecure network channel is an important topic in information security. Two main techniques for improving security in unsecure communication are cryptography and information hiding. In cryptography, a secret message is changed in a way that a third party cannot read the message, and in information hiding, the secret message is hidden in a way that the third party cannot detect the existence of the secret message [1]. Steganography and watermarking are two subsets of information hiding. The first one is used for providing a secure channel in a network communication [2], while watermarking is used for preventing copyright infringement by hiding an invisible identification or a visible logo [3, 4].

The main components of a digital media information hiding system are the "cover," "secret message," "key" and "stego-media." In information hiding, a "cover" is an innocent digital media file that is used for carrying the secret message. The classic digital media files "covers" are video, image, voice and text [5]. However, in some new methods, smart phones [6], cloud storage services [7, 8, 9] and the Skype video traffic [10] are used for carrying the message. A message that the sender wants to send to the receiver side in a secure way through an unsecure channel is a "secret message." Also, "key" is an optional component which is used in some for increasing the security of the system. Lastly, "stego-media" is a media which contains the secret message [11]. In an information hiding system, the "secret message" will be hidden in the "cover" by using an embedding algorithm. Then, at the receiver side, the "secret message" will be extracted by using an extracting algorithm [11].

Images are one of the most popular covers in information hiding, because they have enough capacity for hiding a secret message (the "capacity" is the maximum size of message which can be embedded into the cover). Also, transmitting images through the Internet and local networks is common. Steganography algorithms in images are classified into three types: spatial domain, frequency domain, and adaptive steganography methods [11]. In spatial domain methods, the secret message is embedded by modifying pixel values. The most well-known steganography algorithm in this area is LSB (Least Significant Bit) [12, 13], which hides the secret image in the least signification bit layer of the image. In frequency domain methods, the secret message is hidden by modifying frequency coefficients of the cover image, like Outguess [14], and F5 [15]. In adaptive steganography, a pre-processing statistical analysis, like medical image processing [16], detects the most suitable areas of the cover image for hiding the secret message in both spatial and

frequency domains [11, 17, 18] like the Edge Adaptive method [19].

In this paper, an improved version of Matrix Pattern (MP) method [20] is presented, which operates on RGB images and embeds the secrets in the form of a text message. The main difference between the original and improved methods is based on matrix pattern generation. In previous MP work [20], 49 English keyboard characters are supported, while in this work, 95 characters can be used. Also, the capacity is improved by utilizing the image's green layer for generating a matrix pattern and the whole of the blue layer for hiding the secret message. While in previous work [20], some parts of the blue layer were used in the matrix pattern generation step, and the secret message cannot be embedded in those blue layer parts. In addition, in matrix pattern generation, instead of using four bit layers, two bit layers (the 4^{th} and 5^{th}) are used, which has a direct relationship with improving both the capacity and transparency of the stego-image. Transparency is a measure for comparing cover image and stego-image. Also, there are other differences, which will be discussed in the comparison section (section 5).

The rest of this paper is structured as follows. In section 2, related works are discussed. Then, the specifics of the proposed method are described in section 3. In section 4, evaluation results for selecting the best bit layers for hiding data, and capacity and resistance of the presented work against steganalysis attacks are shown. Next, the new MP algorithm is compared with the previous work. Finally, section 6 concludes our work.

2. Related work

In this paper, a novel image steganography algorithm is presented, based on MP work for hiding text message in an RGB image. In this part two related works including simple MP [20] and LSB-MP [21] are discussed briefly.

2.1. Simple Matrix Pattern

There are several steganography methods for hiding a message in an image. One of the methods is letter mapping [22], which considers a constant mapping table for 32 characters including 26 lower case alphabets and six non-alphanumeric symbols, which replace pixel values of the cover image by constant values of the table. Simple MP [20], is a steganography method which hides text message in a blue layer of an RGB image, as modification to blue layer is less perceptible for the human visual system. In this work, first the image is divided into $B \times B$ non-overlapping blocks. Then, instead of hiding the bit stream of the secret message in the least significant bits, 49 unique matrix patterns are generated from the texture of each block for 49 characters. After that, the 4^{th} to 7^{th} bit layers of the cover image are used for hiding the secret message. These 49 matrix patterns are then assigned to 49 characters including 26 English characters, 10 digit numbers, 12 keyboard non-alphanumeric symbols, and an end of message character.

In this paper 95 matrix patterns are generated which can support all the text messages which can be typed on a standard keyboard. Thus, in this work the secret message also can be a program, e.g., a compiler. The main algorithm of embedding and extracting steps of prior work [20] is similar. However, there are some differences that will be discussed in the following sections. Also, many changes occurred in the matrix pattern generation for improving the method, which is used at both sender and receiver sides. These changes make the new method more efficient and improve both the capacity and transparency of the proposed method; see the comparisons made in section 5 for more details.

2.2. LSB-MP

LSB-MP method [21], is an image steganography technique for hiding a secret message in an RGB image. This algorithm uses both 3-LSB and a simple MP at the same time for hiding a message in the same cover image. It should be noted that a simple MP and 3-LSB methods hide the secret message in the 4^{th} to 7^{th}, and 1^{st} to 3^{rd} bit layers of the selected cover image, respectively. In the other words, this method is using seven out of eight bit layers of the blue layer of the cover image for hiding two kinds of media messages, or just a text message. The red and green layers of the cover image are used based on the 3-LSB method for hiding more information.

This work shows that any kind of steganography method which uses the LSB part of images for hiding message can be combined with MP algorithm for increasing its capacity.

3. Proposed method

As mentioned before, the proposed steganography method does not use a bit stream, and automatically generates unique matrix patterns for each English keyboard character. In section 3.1, the matrix pattern generation phase is described. Then, in section 3.2 the embedding step is introduced and, finally, in section 3.3 the extraction step is defined.

3.1. Matrix Pattern generation

In the MP method, first, the user selects two fixed sizes for blocks, $B \times B$, and matrix patterns, $t_1 \times t_2$. Then, based on the selected block size, the image is divided into non-overlapping blocks. Next, based on the random generation method, locations of blocks are being placed in a queue. Afterward, through the same steganography embedding phase which is explained later, the two selected size values are embedded into the first 64×64 block in the top left of the image, which has a 3×2 matrix pattern size. In the MP method, by analysis, they illustrated that the best block and

matrix pattern sizes are 64×64 and 3×2 [20]. Also, the locations of the blocks, in order from the queue, are hidden in this first block. These sizes and locations are all numbers; therefore, for generating matrix patterns, 10 numbers, one non-alphanumeric symbol for separating different sizes and locations, one null character and one "end of message" character are generated. In large images, the first 64×64 block is not enough for hiding sizes and locations. This is because each block has a limited capacity and, typically, in large images where many block locations must be hidden, one block is not enough. In this case, the pointer of the 64×64 block will be shifted 64 columns to the right, to hide these locations in the next block. If all the locations could not be hidden in the second block, the pointer will shift again, and this process will continue until all the essential keys, and locations become embedded.

It is important to notice that if any $B \times B$ blocks overlaps with this 64×64 block(s), they are ignored and their location is not hidden in the top left 64×64 block(s). After that, the first block in the queue will be selected and the red, green and blue layers of the block are separated. In the proposed method, green is utilized for generating unique matrix patterns, and blue is utilized for embedding secret messages based on the matrix patterns, each of which is assigned to an English keyboard character. The English keyboard characters that are supported by this method are 52 English alphabet letters, both uppercase and lowercase, 10 numbers, 32 non-alphanumeric symbols, one null character and one "end of message" character. Then, based on the $t_1 \times t_2$ matrix pattern size selected by the user, the first top left matrix in the green layer, is selected for generating the first matrix pattern. In this paper notation $[g(i,j)]_{t1\times t2}$ is used to denote this matrix.

For generating these matrix patterns, the first three bits of the matrix are ignored, because they are used in blind attacks based on bit flipping. Afterward, the first row of the matrix pattern is set to zero and, for producing the second row of the matrix pattern, the subtraction of the second and first row of the green matrix-cover is calculated. This procedure will continue until the subtraction of row t_1-1 from row t_1 of the green matrix-cover is computed. Equation (1) shows a subprogram for generating a matrix pattern.

$$MP(i,j) = \begin{cases} 0 & i = 0 \\ g(i,j) - g(i-1,j) & Other \end{cases} \quad (1)$$

The matrix pattern which is created by equation (1) is denoted by $[MP(i,j)]_{t1\times t2}$.

For producing the second matrix pattern, assigned to the second character, the starting point of the matrix-cover will shift one column to right, and the same process will resume for generating the next matrix pattern. This process will continue until all matrix patterns are generated for the 95 characters, or the pointer reaches the end of the row. If the pointer reaches the end of the row, it shifts one row down, and starts again from the leftmost column, until all the 95 characters get a unique matrix pattern. After each matrix pattern is produced, a checking process will identify if any other matrix patterns with similar values are produced sooner in the block or not. If there are any, they would be ignored and the pointer would shift one column to the right again.

203	200	200	205
200	200	201	208
200	199	199	206

200	200	200	200
200	200	200	208
200	192	192	200

Figure 1: The left section is a cover matrix of green layer; the right one is the same matrix after ignoring first three LSB bits

0	0
0	0
0	-8

0	0
0	0
-8	-8

0	0
0	8
-8	-8

Figure 2: Three 3×2 generated matrix patterns

The evaluation that will be discussed in section 4.2 shows that using the 4th and 5th bit layers for producing matrix patterns produces better results. Therefore, if there is at least one value higher than "24" or lower than "-24" in the generated matrix pattern, it will be ignored. In other words, if the 4th bit, which has value 8, and the 5th bit, which has value 16, change then the maximum value change is "24", or "-24". Notice that, in the first step, the first three bits of each pixel in the green layer of the block are ignored. Thus, in this algorithm for generating matrix patterns, only the 4th and 5th bits of these pixels are used. Also, if the matrix pattern generation algorithm could not assign matrix patterns to all 95 keyboard characters by using the green layer of the block, it means that this block is useless and no characters will be hidden in this block during the embedding phase. Therefore, the next block in the queue will be selected for embedding information.

In the left part of Fig. 1, a sample 3×4 matrix from the green layer of a cover image is shown. The right part of Fig. 1 indicates the matrix after ignoring the first three LSB bits. Three different 3×2 matrix patterns which are generated based on the proposed method from the right part of Fig. 1 are shown in Fig. 2.

3.2. Embedding phase

In the embedding step, the blue layer of the chosen block is selected. Then, the matrix pattern which is assigned to the first character of the secret message is found in the matrix pattern database of the block. For embedding this pattern, the first top left $t_1 \times t_2$ matrix of the blue layer of the block will be chosen; $[b(i,j)]_{t1*t2}$ indicates this matrix. For producing the "stego-matrix," the first row of the blue matrix has no change. To attain the values of the second row, the values of the first row of the "stego-matrix" are

159	160	0	0	159	160
159	159	0	8	159	168
161	159	-8	-8	151	160

Figure 3: Left section is a 3x2 cover-matrix, middle section is the matrix pattern, and right section is the "stego-matrix"

added to the values of the second row of the matrix pattern. This process will continue until the t_1 row of the "stego-matrix" is produced; this is done by adding the t_1-1 row of the "stego-matrix" to the t_1 row of matrix pattern. The subprogram of the embedding step is shown in equation (2). In this equation, the $[MP(i,j)]_{t1\times t2}$ is embedded in the $[b(i,j)]_{t1*t2}$.

$$Em(i,j) = \begin{cases} b(0,j) & i = 0 \\ Em(i-1,j) + MP(i,j) & Other \end{cases} \quad (2)$$

Values of the "stego-matrix" in the blue layer are illustrated by $[Em(i,j)]_{t1*t2}$, which is produced by equation (2). For hiding the second matrix pattern, the pointer will shift t_2 columns to right. This process will continue until it reaches the end of the row. Then, it will shift t_1 row down, and it will move to the leftmost $t_1 \times t_2$ matrix in a new row. This process will continue for embedding all the secret message characters, and if the space of the block is not enough, the process will go on with the next block(s) in the queue with their own unique matrix patterns. Finally, when all the characters are embedded in the blocks, the matrix pattern assigned to the "end of message" will be hidden.

The left part of the Fig. 3 shows a sample of 3×2 cover-matrix. The middle section in Fig. 3 is the matrix pattern which should be hidden. The right section of Fig. 3 is the "stego-matrix" which will be replaced by the sample cover-matrix.

In this embedded system if the difference between the first row in the cover-matrix and other rows would be large (i.e., an edge), then such an addition would modify adjacent rows too greatly. However, this case happens infrequently in natural images and it should be noticed that changing value of pixels in edge part of the cover image is not detectable. There are some steganography algorithms, like PVD (Pixel Value Differencing) [23], which hides the secret message in the edge part of the image by changing pixel's values by a large amount. In this method, if pixel overflow happens during the hiding of a matrix pattern, a special matrix pattern with all zero values is hidden, instead of the real matrix pattern. This is because, it is impossible for a matrix pattern with all zero values to change pixels to values higher than 255 or less than 0. In addition, because it is a pre-defined matrix pattern, if during the matrix pattern generation phase a matrix pattern with all zero values is generated, it will be ignored. This pre-defined matrix pattern is not included in the 95 matrix patterns that are generated during the matrix pattern generation phase.

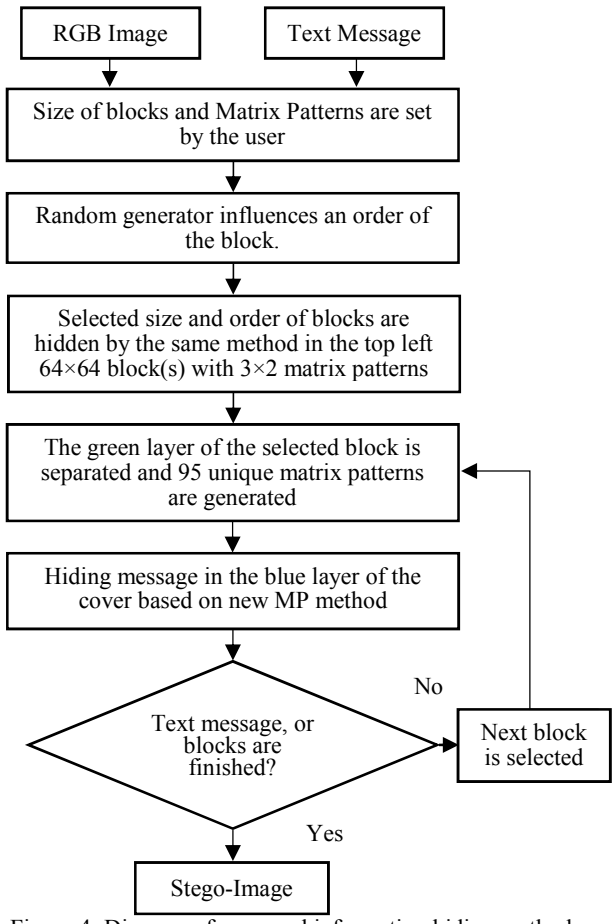

Figure 4: Diagram of proposed information hiding method

Finally, if the entire secret message cannot be embedded completely in all the blocks of the image, the program will show how many characters are hidden and how many are left. The secret message embedding process is shown in Fig. 4.

3.3. Extracting phase

To extract the secret message from the stego-image on the receiver side, first, the program detects the size of both the blocks and the matrix patterns, as well as the order of the block locations. These values are embedded in the first 64×64 block(s) on the top left of the image by 3×2 matrix patterns. Extracting these values is same as the extraction phase which is explained ahead. Next, the extracted block locations are placed in order in a queue, the first block in the queue is chosen, and the three layers of the RGB block are separated. Then, using the green layer of the block and the size of the matrix patterns that were detected from the first 64×64 block(s), the 95 unique matrix patterns of this block are generated by using equation (1), which was discussed in the "matrix pattern generation" part, section 3.1.

To detect the hidden message, after generating the matrix patterns, first the blue layer of the "stego-block" is separated. Next, to detect the first hidden matrix pattern, the first $t_1 \times t_2$ matrix on the top left of the blue layer of the "stego-block" is selected. Here, the first row of the matrix pattern is assumed to be zero and, to detect the 2nd row of the hidden matrix pattern, the subtraction of the 2nd row from the 1st row of the "stego-block" is calculated. Then, to attain the 3rd row of the matrix pattern, the subtraction of the 3rd row from the 2nd row of the "stego-matrix" is calculated. This process will continue until the subtraction of t_1-1 from t_1 row is calculated, to extract the values of the t_1 row of the hidden matrix pattern. Equation (3) indicates a subprogram that is used for extracting matrix patterns assigned to an English, number or non-alphanumeric symbol keyboard character.

$$Ex(i,j) = \begin{cases} 0 & i = 0 \\ Em(i,j) - Em(i-1,j) & Other \end{cases} \quad (3)$$

In Equation (3), "*Ex*" refers to the extracted matrix pattern, and "*Em*" is the "stego-matrix" shown as $[Em(i,j)]_{t1*t2}$. Also, the notation $[Ex(i,j)]_{t1*t2}$ indicates the extracted matrix pattern, which is same as one of the generated matrix patterns in a block.

Then, the pointer of the matrix will shift t_2 columns to right, to extract the second matrix pattern and the hidden character. This process will continue until the pointer reaches the end of the column. Afterward, it will shift t_1 rows down and start again from the leftmost $t_1 \times t_2$ matrix in the new row of the block. In this work, each extracted matrix pattern from the blue layer is assigned to the keyboard character that is hidden in each $t_1 \times t_2$ matrix. If a matrix pattern with all zero values is extracted, it will show that no character is hidden in this matrix, and the pointer will move t_2 columns to right to extract the next character. This trend will continue until all the hidden characters in the block are extracted. Then, the next block in the queue will be selected, and the same process will go on until it detects the "end of message" character in one of the blocks. If all the blocks of the image that were in the queue be used in the embedding phase, the "end of message" character would not be hidden in any matrix of the blocks.

4. Implementation and evaluation

In the first part of this section there is a discussion about the implementation, inputs and outputs of the proposed steganography method. In the next section, the most suitable bit layers for producing matrix patterns are evaluated through different analyses. Then, the maximum number of characters is embedded in some sample images, to show the maximum capacity and resistance of this work against steganalysis attacks.

4.1. Proof of concept implementation

MATLAB R2016b package is used for implementation and evaluation of our proposed method. The inputs of this steganography algorithm are two values, which are selected by the user on the sender side. The first value is the size of the square block, which is $B \times B$, and the other value is the size of the matrix pattern, which is $t_1 \times t_2$. Then, order of blocks based on the random generator are placed in a queue. Also, the user selects an RGB image as a cover, and a secret message for embedding in the selected cover image. In this work, the secret message can be a text message composed of both the English and non-alphanumeric symbols of the keyboard, which can be even a program or compiler. The output of this system is an image, stego-image. At the receiver side, the produced stego-image is the input to the steganography system, and the output of the program would be a visible secret message, if existing, extracted from the stego-image.

In this program, the input image can be in any available image format which has three layers. However, the stego-image can only be in the PNG, Bitmap or TIFF format. The program cannot produce a stego-image with JPEG or JPEG2000 formats, because this steganography method hides data in the spatial domain of the image and, during lossy compression, the embedded message would be destroyed.

4.2. Matrix Pattern evaluation

In the paper where MP was originally introduced [20] the 4th through the 7th bits of pixels were used for producing matrix patterns. It should be noticed, that using less significant bits in MP work can improve transparency, because the stego-image is changed less frequently when compared to the "cover-matrix." Furthermore, in this method, using less significant bits can also develop the capacity of the proposed algorithm; because, as discussed in section 3.2, the values of the matrix patterns are added to the pixels of the blue layer of the cover image and, during this process, it is possible that the value of one of the pixels may become higher than 255 or less than 0. This would cause that matrix to become useless for embedding secret characters. In general, if the values of the matrix patterns change in higher significant bits, the probability of pixel overflowing will increase. However, if the number of bit layers decreases, it is probable that generating 95 unique matrix patterns will not be possible and the whole block will become ineffective.

In this evaluation, 13 different images, including "Airplane," "Arctichare," "Baboon," "Cat," "Fruits," "Girl," "Lena," "Monarch," "Peppers," "Pool," "Sails," "Tulips" and "Watch," are evaluated. These images, in PNG format, are selected out of the image database "Public-Domain Test Images for Homeworks and Projects" at the University of Wisconsin Madison [24], as these

images were used as sample images in earlier LSB-MP work [21]. In this evaluation, the values of the block and matrix pattern size, are established as 64×64 and 3×2, respectively. As mentioned earlier, the best block and matrix pattern sizes are 64×64 and 3×2 based on previous results [20]. The secret message in this evaluation is large enough that all the blocks are used for hiding it.

Then, our algorithm is tested with different bit layers, by embedding the maximum number of characters that can be hidden in these 13 images. Firstly, this method is evaluated by using the 4th to 7th bits of pixels, like the simple MP work [20]. Then, it is analyzed by using the 4th through 6th bit layers. In the next analysis, the method is evaluated by using the 4th to 6th bit layers again, but this time if one change happens in the 6th bit layer of the pixels, the other two bit layers cannot change. In the next analysis, only the 4th and 5th layers are used for generating matrix patterns. Finally, only one of the 4th or 5th bit layers can change. The results of these experiments are shown in Table (1). As can be seen, the maximum possible changes (maximum value of matrix patterns' elements) for these steps are 120, 56, 32, 24, 16 and their negative equivalents, respectively. For comparing this steganography method with different bit layers, the average maximum capacity and average PSNR (peak signal to noise ratio) play key roles.

To compute PSNR, initially, the brightness is derived by equation (4) [25, 26], which is famous as a luminance formula [27], and was used in previous MP and LSB-MP works [20, 21]. In this equation "*r,*" "*g*" and "*b*" are respectively the red, green and blue layers of an image, and "*B*" is the brightness.

$$B(P(r,g,b)) = (0.2126*r) + (0.152*g) + (0.0722*b) \quad (4)$$

Then, the mean square error (MSE) of the brightness of the cover image and stego-image is calculated by using equation (5). In this equation, "n" and "m" are the column size and row size of the images, while "*I*" and "*S*" represent the "cover image" and stego-image, respectively.

$$MSE = \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m}\left[B\big(I(i,j)\big) - B\left(S(i,j)\right)\right]^2 \quad (5)$$

Finally, PSNR is calculated by equation (6).

$$PSNR(I,S) = 20.log_{10}(255/\sqrt{MSE(I,S)}) \quad (6)$$

PSNR measures transparency in an image; for example, if PSNR value is more than 30, any changes in the image cannot be identified by the human eye [26, 28, 29].

In addition, resistance to "Regular Singular" (RS) [30] and "Sample Pair" (SP) [31, 32] attacks by using "Steganography_Studio1.0.1" [33] is computed. The measures of the RS and SP attacks are based on the probability that these attacks predict an image as a stego-image. Also, the number of blocks that were useless and could not hide any messages is divided by the total number of blocks in the image which is shown in Table (1) under the "Fail Block" column. The results shown in Table (1) are rounded to two decimal points.

Table 1: Results of the proposed method based on using different bit layers for matrix pattern generation

	Avg. Capacity [chrs]	*Avg. PSNR [dB]*	*Avg. RS [%]*	*Avg. SP [%]*	*Fail Block*
4 to 7	39330.31	45.28	5.88	4.84	126/1062
4 to 6	40468.38	45.92	3.48	6.14	123/1062
(4 to 5) or 6	42083.92	46.87	3.62	6.35	135/1062
4 to 5	43437.92	47.14	3.47	6.23	135/1062
4 or 5	42777.38	47.64	3.66	6.27	171/1062

According to this table, using the "4th to 5th" bit layers not only has the best capacity result, but it also has a better PSNR in comparison with three of the other layers, when the message is hidden in the entire image. The "4th or 5th" group only has a better PSNR than the "4th to 5th" group because it contains less embedded messages. In addition, all the groups have a good and acceptable resistance against these two steganalysis methods. Thus, in this work for matrix pattern generation, the "4th to 5th" bit layers are used; this means that the values in the generated matrix pattern can be 0, 8, -8, 16, -16, 24 and -24.

4.3. Proposed method evaluation

The maximum number of characters that can be embedded in the $B\times B$ block, through the $t_1\times t_2$ matrix pattern size, is calculated by equation (7).

$$Max_{Capacity} = floor\ (BxB/t_1xt_2) \quad (7)$$

If the sizes of block and matrix pattern are 64×64 and 3×2 correspondingly, then 682 characters can be hidden in each block. However, it is possible that fewer characters can be hidden in the block if, during the embedding phase, pixel overflowing happens. Thus, the number of characters which can be hidden in each block is dynamic and it can be between zero to $Max_{Capacity}$ in equation (7).

Table (2) indicates the results of the proposed steganography technique for the selected 13 images, with maximum capacity, PSNR (when the whole image is used for hiding), and probability of detection, based on the RS and SP steganalysis methods for each image; values in this table are rounded to two decimal points. Based on Table (2), RS and SP steganalysis attacks are not successful in detecting these stego-images. Also, it shows that the proposed method has a fine PSNR and does not changes pixels in a noticeable way when the entire image is used for embedding secret messages. Moreover, the average and

Table 2: Results of maximum embedding in 13 different images

	Capacity [chrs]	*PSNR [dB]*	*RS [%]*	*SP [%]*	*Fail block*
Airplane	34618	48	1.19	5.15	11/64
Arctichare	18431	50.78	3.063	2.4	10/54
Baboon	40676	41.2324	2.63	29.72	0/64
Cat	45736	44.89	3.2	2.93	5/77
Fruits	39778	47.36	2.48	2.52	2/64
Girl	42215	49.06	1.36	1.81	26/96
Lena	42336	48.33	5.1	7.19	0/64
Monarch	38808	46.59	0.85	1.97	37/96
Peppers	38147	47.44	5.84	5.69	0/64
Pool	10509	51.39	9.18	9.2	10/35
Sails	63721	44.42	7.55	8.73	0/96
Tulips	59829	45.72	2.41	3.57	0/96
Watch	89889	47.62	0.22	0.14	34/192
Average	43437.92	47.14	3.47	6.23	135/1062

total capacity of these images are “43437.92” and “564693” characters, respectively. This means that, on average, each useful block can hide “609.18” characters.

Another steganalysis attack used for evaluation in this method is an attack which can detect a stego-image based on the PVD steganography method [34]. The PVD method hides data by subtraction of neighboring pixels [23]; therefore, testing the proposed data hiding method, which has the same trait for embedding characters, is necessary. Based on this attack, the difference of neighboring pixels is calculated and illustrated through a histogram. Natural images have a Gaussian-shaped histogram, but if data is hidden in them by the PVD method, the Gaussian shape will be changed. Fig. 5.a, and 5.b show the cover and stego-image of “Fruits” by maximum capacity, respectively. Also, Fig. 6.a, and 6.b indicate the histogram attack based on the PVD, which is applied on the cover and stego-image of “Fruits,” correspondingly. Fig. 6.b shows the stego-image based on this attack has the Gaussian shape. Thus, it seems unlikely that this attack can detect the stego-image when all the blocks are used for hiding.

5. Comparison

In this section, the proposed method is compared to earlier MP work [20]. First, the differences between these two algorithms are listed below:

1) In previous work, 49 matrix patterns were generated, but in this algorithm 95 unique matrix patterns are assigned to 95 English, number and non-alphanumeric symbol keyboard characters. As all the keyboard characters are supported by this method, the secret message can be a program in a computer programming language.

2) In earlier methods, selecting blocks was based on a seed and pseudo-random generator, but in this work, a truly random generator is used. As the order of blocks sent is

Figure 5.a: “Fruits” as a cover image

Figure 5.b: “Fruits” as a stego-image by hiding “39778” characters.

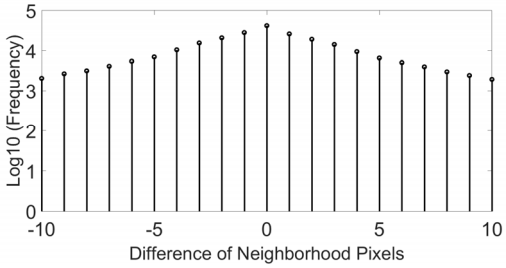

Figure 6.a: Histogram of PVD attack apply on the cover image

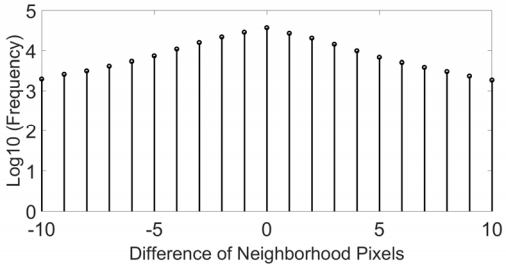

Figure 6.b: Histogram of PVD attack apply on the stego-image

fixed by method, it provides a condition that in the future works, the best blocks can be selected instead of using pseudo random generator in both side.

3) The previous algorithm used four bit layers, the 4th to 7th, for information hiding, while this recent version uses the 4th and 5th bit layers, which has a direct relationship with improving both capacity and transparency simultaneously.

4) In the MP method, the intensity of each block was used for producing different matrix patterns, and then the remainder of the blue layer of selected block was used for embedding. However, in our algorithm, the green and blue layers are applied for matrix pattern generation and hiding secret information, correspondingly. Thus, the whole blue layer is used for hiding purposes.

5) In the earlier method, in the MP work for generating matrix patterns, when one row was finished, it would shift t_1 rows down, but in this algorithm, it just shifts one row down. This process has a vital effect for generating 95 unique matrix patterns.

6) Lastly, block and matrix pattern sizes by the seed of pseudo random generator was hidden in the first 48×48 block and 3×3 matrix pattern size. However, in this work size of selected block and matrix pattern by user and the orders of selected blocks are hidden in the first 64×64 block(s), by 3×2 matrix pattern size.

For comparing these two methods, the maximum possible number of characters which is hidden in these 13 images with the earlier MP method is computed. The average results of maximum capacity, PSNR, detectability by RS, and SP attacks are "34139.15," "46.25," "3.47" and "5.75," respectively. According to the Table (2) and these results, and [21], the new algorithm embedded 27.24% more characters than the earlier method. Furthermore, although it hides more characters, it also has a slightly better transparency and it improves PSNR by approximately 3 percent. Using the entire blue layer for embedding has a direct relationship with capacity improvement, and utilizing two bit layers instead of four bit layers plays a key role in having a better PSNR. Also, both algorithms have high resistance against RS and SP steganalysis attacks, as well as the same resistance against RS.

6. Conclusion and future work

In this paper, a steganography method is presented, based on matrix patterns for embedding secret messages in the form of text. In this method, initially, the image is divided in non-overlapping $B \times B$ blocks. Then, blocks are placed in a queue by using a random generator. Next, instead of hiding bit streams of secret messages, it generates 95 unique matrix patterns from the green layer of each block for 95 English, number and non-alphanumeric symbol keyboard characters. In the embedding phase, based on the secret message, these matrix patterns are added to the blue layer of the blocks through a special process. For extracting, on the receiver side, the same process must be reversed. Evaluation results show that the proposed method has a high capacity and PSNR; also, they show its resistance against some well-known steganalysis attacks, such as RS, SP and the PVD histogram attacks. Finally, as compared to the similar MP method, results illustrate not only that the new method improves capacity by more than 27 percent, but that it also improves PSNR by nearly 3 percent.

In future work, the proposed solution can be extended to select blocks from the most to the least suitable. Also, generating 256 matrix patterns makes the method able to hide all kindes of digital media as a secret message.

Acknowledgments

The work of Amirfarhad Nilizadeh and Gary Leavens was supported in part by the US National Science Foundation under grant SHF1518789. The work of Gary Leavens was also supported in part by NSF grant CNS1228695.

References

[1] W. Mazurczyk, and L. Caviglione. Information Hiding as a Challenge for Malware Detection. In *IEEE Security and Privacy Magazine*, 13(2):89-93, 2015.

[2] D. Lerch-Hostalot, and D. Megías. LSB matching steganalysis based on patterns of pixel differences and random embedding. In *Computers & Security*, 32:192-206, 2013.

[3] G. Bhatnagar, Q. M. J. Wu, and B. Raman. A new robust adjustable logo watermarking scheme. In *Computers & Security*, 31(1):40-58, 2012.

[4] H. B. Golestani, M. Joneidi, and M. Ghanbari. Logo watermarking with unequall strength for improved robustness against attacks. In *7th International Symposium on Telecommunications (IST),* pages 827-832, 2014.

[5] P. Marwaha, and P. Marwaha. Visual cryptographic steganography in images. In *Computing Communication and Networking Technologies (ICCCNT)*, 1(6):29-31, 2010.

[6] W. Mazurczyk, and L. Caviglione. Steganography in Modern Smartphones and Mitigation Techniques. In *IEEE Communications Surveys & Tutorials*, 17(1):334-357, 2015.

[7] W. Mazurczyk, and K. Szczypiorski. Is Cloud Computing Steganography-proof?. In *Third International Conference on Multimedia Information Networking and Security*, pages 441-442, 2011.

[8] M. Ahmadian, A. Paya, and D. C. Marinescu. Security of Applications Involving Multiple Organizations and Order Preserving Encryption in Hybrid Cloud Environments. In *Parallel & Distributed Processing Symposium Workshops (IPDPSW),* pages 894-903, 2014.

[9] M. Ahmadian, F. Plochan, Z. Roessler, and D. C. Marinescu. SecureNoSQL: An approach for secure search of encrypted NoSQL databases in the public cloud. In *International Journal of Information Management,* 37(2):63-74, 2017.

[10] W. Mazurczyk, M. Karas, K. Szczypiorski, and A. Janicki. YouSkyde: information hiding for skype video traffic. In

Multimedia Tools and Applications, 75(21):13521-13540, 2016.

[11] A. Cheddad, J. Condell, K. Curran, and P. M. C. Kevitt. Digital image steganography: Survey and analysis of current methods. In *Signal Processing*, 90(3):727-752, 2010.

[12] R. Chandramouli, and N. Memon. Analysis of LSB based image steganography techniques. In *Image Processing*, 3, 2001.

[13] D. C. Lou, and J. L. Liu. Steganographic method for secure communications. In *Computers & Security*, 21(5):449-460, 2002.

[14] N. Provos. Defending against statistical steganalysis. In *Usenix security symposium*, 10:323-336, 2001.

[15] A. Westfeld. F5—a steganographic algorithm. In *International workshop on information hiding*, pages 289-302, 2001.

[16] F. Farahani Vasheghani, A. Ahmadi, and M. H. F. Zarandi. Lung nodule diagnosis from CT images based on ensemble learning. In *Computational Intelligence in Bioinformatics and Computational Biology (CIBCB),* pages 1-7, 2015.

[17] A. D. Ker, P. Bas, R. Böhme, R. Cogranne, S. Craver, T. Filler, J. Fridrich, and T. Pevný. Moving steganography and steganalysis from the laboratory into the real world. In *Proceedings of the first ACM workshop on Information hiding and multimedia security*, pages 45-58, 2013.

[18] T. D. Denemark, M. Boroumand, and J. Fridrich. Steganalysis features for content-adaptive JPEG steganography. In *IEEE Transactions on Information Forensics and Security*, 11(8):1736-1746, 2016.

[19] W. Luo, F. Huang, and J. Huang. Edge adaptive image steganography based on LSB matching revisited. In *IEEE Transactions on Information Forensics and Security*, 5(2):201-214, 2010.

[20] A. F. Nilizadeh, and A. R. Naghsh Nilchi. Steganography on RGB Images Based on a "Matrix Pattern" using Random Blocks. In *International Journal of Modern Education and Computer Science (IJMECS)*, 5(4):8-18, 2013.

[21] A. Nilizadeh, and A. R. Naghsh Nilchi. A novel steganography method based on matrix pattern and LSB algorithms in RGB images. In *1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, pages 154-159, 2016.

[22] M. A. F. Al-Husainy. Image steganography by mapping pixels to letters. In *Journal of Computer science*, 5(1), 2009.

[23] D. C. Wu, and W. H. Tsai. A steganographic method for images by pixel-value differencing. In *Pattern Recognition Letters*, 24(9):1613-1626, 2003.

[24] Public-Domain Test Images for Homeworks and Projects, 2017. [Online]. URL: http://homepages.cae.wisc.edu//~ece533/images. [Accessed: 03- Apr-2017].

[25] O. Holub, and S. T. Ferreira. Quantitative histogram analysis of images. In *Computer physics communications*, 175(9):620-623, 2006.

[26] H. Tozuka, M. Yoshida, and T. Fujiwara. Salt-and-pepper image watermarking system for IHC evaluation criteria. In *Proceedings of the 1st international workshop on Information hiding and its criteria for evaluation*, pages 31-36, 2014.

[27] A. O. Akyüz, and E. Reinhard. Color appearance in high-dynamic-range imaging. In *Journal of Electronic Imaging*, 15(3), 2006.

[28] A. F. Nilizadeh, and A. R. Naghsh Nilchi. Block Texture Pattern Detection Based on Smoothness and Complexity of Neighborhood Pixels. In *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, 6(5):1-9, 2014.

[29] C. M. Wang, N. I. Wu, C. S. Tsai, and M. S. Hwang. A high quality steganographic method with pixel value differencing and modulus function. In *Journal of Systems and Software*, 81(1):50-158, 2008.

[30] J. Fridrich, M. Goljan, D. Hogea, and D. Soukal. Quantitative steganalysis of digital images: estimating the secret message length. In *Multimedia systems* 9(3):288-302, 2003.

[31] S. Dumitrescu, X. Wu, and Z. Wang. Detection of LSB steganography via sample pair analysis. In *IEEE Transactions on Signal Processing*, 51(7):1995-2007, 2003.

[32] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker. Digital watermarking and steganography. In *Morgan Kaufmann*, 2007.

[33] StegStudio project URL: http://sourceforge.net/projects/stegstudio/files/, Last visited 03/24/2017.

[34] X. Zhang, and S. Wang. Vulnerability of pixel-value differencing steganography to histogram analysis and modification for enhanced security. In *Pattern Recognition Letters*, 25(3):331-339, 2004.