Digital Image Steganography Using Bit Flipping

Aditya Kumar Sahu¹, Gandharba Swain²
¹²Department of Computer Science & Engineering, K L University, Andhra Pradesh, India
¹Department of Computer Science & Engineering, GMRIT, Andhra Pradesh, India

ABSTRACT

This article proposes bit flipping method to conceal secret data in the original image. Here a section consists of 2 pixels and there by flipping one or two LSBs of the pixels to hide secret information in it exists in 2 variants. The variant-1 and variant-2 both use 7th and 8th bit to conceal the secret data. Variant-1 hides 3 bits per a pair of pixels and the variant-2 hides 4 bits per a pair of pixels. Our proposed method notably raises the capacity as well as bits per pixel that can be hidden in the image compared to existing bit flipping method. The image steganographic parameters such as, peak signal to noise ratio (PSNR), hiding capacity, and the quality index of the proposed techniques has been compared with the existing bit flipping technique.

1. INTRODUCTION

Today information hiding became a dominant area in every aspect of life. The real threat to data is in the field of digital data communication. The internet is the principal entity for carrying the digital data. The security to the data is a matter of real interest. In this aspect, cryptography and steganography are the prominent fields for achieving surveillance to secret data. Cryptography misleads the unapproved entity by manipulating the original information [1-3]. Steganography, other side shields the data so that the unapproved entity cannot even predict the survival of data. It comes in diverse ways such as implanting the data in the image, video, audio etc. [4, 5]. Image steganography has gained the attention of many researchers during the years. Implanting text data in the original or cover image is image steganography, the new image which carries the information is resulting image [6].

The efficacy of any image steganographic method depends on various parameters. Parameters like capacity, bits per pixel (BPP), peak signal to noise ratio (PSNR), Quality index (QI) determines the impression of the method [2-4]. The amount of information a resulting image can hide is the capacity for the method. The bits per pixel (bpp) is the measure of the number of secret data in bits that are concealed in the pixel of an original image. The PSNR is a metric for finding the quality of a resulting image. The more the PSNR is the better the method. Q.I tell about the closeness between the original image and resulting image.

Least significant- bit (LSB) is the familiar and simpler technique gives better capacity but the relative reduction in the quality of resulting image, [6]. The use of Moderately-Significant bit (MSB) by pixel adjustment process of the cover image for concealing information has been suggested by Wang et al. [7]. A genetic algorithm based on perceptual modeling with a combination of rightmost LSB in order to achieve large capacity has been proposed by Wang et al. [8]. Although the capacity increases by using Wang et al. [8] method but it the price of hike in the complexity. To avoid the scenario and to reduce the complexity Chan and Cheng [9] suggested an optimal pixel adjustment process. The LSB bit is not altered straightforward by which...
the error can be cut down drastically. The combination of cryptography and steganography can achieve a better security for the data [10]. LSB methods is valuable for hiding the larger magnitude of data. But, Fridrich et al. [11] found the weakness of LSB methods. In 2003, Wu and Tsai [12] suggested a unique method in the ground of image steganography which is known as pixel value differencing (PVD). Here the difference between two consecutive pixels is taken into account and a new difference value is found to concealing the secret data. The amount of secret data will be hidden is depends upon the difference value between the consecutive pixel. A larger difference value indicates a higher amount of secret data hidden in the image. The image can be segregated into either smooth area or edge area. The edge areas can be utilized to hide more number of data compared to smooth areas. A large variety of diversified techniques in combination with PVD and LSB has been suggested in literature [13:23]. Swain [24] given a method for inserting secret information by using group of bits substitution (GBS). He suggested 1GBS and 2GBS techniques. The 1GBS technique hides 1 bit and 2GBS hides 2 bits of secret data. A novel track in image steganography called LSB array has been suggested in [25]. The above works have been further continued in [26].

2. EXISTING TECHNIQUE

Kumar and Chand [2] proposed a bit flipping method for concealing 1 bit of secret data in a pair of pixel. A block of 2 pixels taken which hides single bit only. The last LSB bit has been exploited for concealing the secret data at the first layer of embedding. In the existing work, the block of 2 pixels hides only 1 bit of secret information. This motivates us for providing 2 variants of bit flipping which can conceal 2 and 3 bits of secret data respectively in a segment or block of two pixels. The embedding and extraction process of the said variants has been proposed below.

3. PROPOSED BIT FLIPPING TECHNIQUE

This technique divides the cover image into sections. Each section (S) consists of 2 pixels say, $S_x$ and $S_{x+1}$, for $x=1$ to $N-1$. Where ‘N’ is the total number of pixel elements of the image. The bit flipping technique exists in two variants, (i) Bit flipping-1 and (ii) Bit flipping-2.

3.1.1 Embedding Algorithm

Step-1: Change the data to be hidden into binary. The dimension of secret data is also changed to 18 bit. This is placed at the top of the binary message. The message is now combination of both.
Step-2: Read the original image ($I_x$). Convert it to binary also read the last 2 bits i.e 7th and 8th bit to form the location map from all the pixels of the original image $I_x$, $x=1$ to $N$, where each $I_x$ is a pixel of 8-bit length. Compress the location map and send it to the receiver.
Step-3: Read the secret data in binary. Divide the secret data into a block of three bits of length. So, the three bits can range from 000 to 111.
Step-4: Let the block of secret data is $k_i$ where $i = 1$ to $\frac{n}{3}$, where $n$ is the length of the secret message.
Step-5: For each section (S) of the original image ($I_x$), let $S_x$ and $S_{x+1}$ be the two consecutive pixels of the section. For each section (S) of original image ($I_x$),
- If $k_i=000$, No change to any pixel of the section.
- Else if $k_i = 001$, Flip the 8th LSB bit of $S_x$ only.
- Else if $k_i = 010$, Flip the 7th LSB bit of $S_x$ only.
- Else if $k_i = 011$, Flip the 8th LSB bit of $S_{x+1}$ only.
- Else if $k_i = 100$, Flip the 7th LSB bit of $S_{x+1}$ only.
- Else if $k_i = 101$, Flip the 8th LSB bit of $S_{x+1}$ only.
- Else if $k_i = 110$, Flip the 7th, 8th LSB bit of $S_{x+1}$ only.
- Else if $k_i = 111$, Flip the 7th, 8th LSB bit of both the section.
Step-6: Transmit the obtained resulting image ($G$) along with the compressed location map to the receiver. The embedding process is successful.

3.1.2 Extraction Algorithm

The extraction algorithm is opposite of embedding. The various steps for extracting the secret data are as follows.
Step-1: Decompress the compressed location map and find the 7th and 8th bit of original image. Initialize $M_i$ to empty and initialize the counter, $\text{count}=1$.

Step-2: For each section (S) of the resulting image (G) repeat step-3.

Step-3: Compare 7th and 8th bit of each section (S) of resulting image with the location map. If 7th and 8th bit of both pixels $S_x$ and $S_{x+1}$ are same as corresponding bits of location map then extract 000 and concatenate to $M_i$.

Else if only 8th LSB bit of $S_x$ has changed then extract 001 and concatenate to $M_i$.

Else if only 7th LSB bit of $S_x$ has changed then extract 010 and concatenate to $M_i$.

Else if only 8th LSB bit of $S_{x+1}$ has changed then extract 011 and concatenate to $M_i$.

Else if only 7th LSB bit of $S_{x+1}$ has changed then extract 100 and concatenate to $M_i$.

Else if only 7th, 8th LSB bit of $S_x$ has changed then extract 101 and concatenate to $M_i$.

Else if only 7th, 8th LSB bit of $S_x$, $S_{x+1}$ have changed then extract 110 and concatenate to $M_i$.

Else if 7th and 8th bit of both pixels $S_x$ and $S_{x+1}$ have changed then extract 111 and concatenate to $M_i$.

Step-4: Set count=count+3. If count < 18 then go to step3, otherwise go to step5.

Step-5: Convert the 18 bits in $M_i$ to decimal and then multiply by 7, which is the length of the embedded message in bits, let it be n.

Step-6: Reinitialize $M_i$ to blank, and for $i=1$ to $\frac{n-18}{3}$ repeat step3. Then we get $M_i$ with n-18 bits length.

Step-7: Convert the bits of $M_i$ to characters. The extraction is successful.

3.2. Bit Flipping-2:

The 7th and 8th bits of the original image ($I_x$) is used to hide the secret data. Each section (S) hides 4 bit of secret data sequentially. The embedding and extraction procedures are as follows.

3.2.1 Embedding Algorithm

Step-1: Change the data to be hidden into binary. The dimension of secret data is also changed to 18 bit. This is placed at the top of the binary message. The message is now combination of both.

Step-2: Read the original image ($I_x$). Convert it to binary. Where, $x=1$ to N, where each $I_x$ is a pixel of 8 bit length. Read the last 2 bits that are 7th and 8th bit from all the pixels of the original image ($I_x$) and form the location map. Compress the location map and send it to the receiver.

Step-3: Read the secret data in binary. Convert it into blocks of four bits in length. The four bits can range from 0000 to 1111.

Step-4: Let the block of secret data is $k_i$, where $i=1$ to $\frac{n}{4}$, where n is the length of secret message.

Step-5: For each section (S) of original image (I), let $S_x$ and $S_{x+1}$ be the two consecutive pixels of the section.

If $k_i=0000$, No change to any pixel of the section.

Else if $k_i=0001$, Flip the 8th LSB bit of $S_x$ only.

Else if $k_i=0010$, Flip the 7th LSB bit of $S_{x+1}$ only.

Else if $k_i=0011$, Flip the 7th, 8th LSB bit of $S_{x+1}$ only.

Else if $k_i=0100$, Flip the 8th LSB bit of $S_x$ only.

Else if $k_i=0101$, Flip the 8th LSB bits of both $S_x$ and $S_{x+1}$.

Else if $k_i=0110$, Flip the 8th LSB bit of both $S_x$ and 7th LSB bit of $S_{x+1}$.

Else if $k_i=0111$, Flip the 8th bit of $S_x$ and 7th & 8th LSB bits of $S_x$ and $S_{x+1}$.

Else if $k_i=1000$, Flip the 7th LSB bit of $S_x$ only.

Else if $k_i=1001$, Flip the 7th LSB bit of $S_x$ and 8th LSB bit of $S_{x+1}$.

Else if $k_i=1010$, Flip the 7th LSB bits of both $S_x$ and $S_{x+1}$.

Else if $k_i=1011$, Flip the 7th LSB bit of $S_x$ and both 7th & 8th LSB bits of $S_{x+1}$.

Else if $k_i=1100$, Flip the 7th, 8th LSB bit of $S_x$ only.

Else if $k_i=1101$, Flip the 7th, 8th LSB bits of $S_x$ and 8th LSB of $S_{x+1}$.

Else if $k_i=1110$, Flip both the 7th & 8th LSB bits of $S_x$ and 7th LSB bit of $S_{x+1}$.

Else if $k_i=1111$, Flip the 7th and 8th LSB bits of $S_x$ and $S_{x+1}$.

Step-6: Transmit the obtained resulting image (G) along with the compressed location map to the receiver. The embedding process is successful.

3.2.2 Extraction Algorithm

The extraction algorithm is opposite of embedding process. The various steps of retrieving the secret data are outlined below.

Step-1: Decompress the compressed location map and find the 7th and 8th bits of original image. Initialize $M_i$ to empty and initialize the counter, $\text{count}=1$.

Step-2: For each section (S) of the resulting image (G), repeat the step3.
**Step-3:** Compare each section (S) of resulting image with the location map. If 7th and 8th bit of both pixels $S_x$ and $S_{x+1}$ are same as corresponding bits of location map then extract 0000 and concatenate to $M_1$.
Else if only 8th LSB bit of $S_{x+1}$ has changed then extract 0001 and concatenate to $M_1$.
Else if only 7th LSB bit of $S_{x+1}$ has changed then extract 0010 and concatenate to $M_1$.
Else if only 7th, 8th LSB bit of $S_{x+1}$ has changed then extract 0011 and concatenate to $M_1$.
Else if only 8th LSB bit of $S_x$ has changed then extract 0100 and concatenate $M_1$.
Else if 8th LSB bit of $S_x$ and 7th LSB bit of $S_{x+1}$ both have changed then extract 0101 and concatenate to $M_1$.
Else if 8th LSB bit of $S_x$ and 7th, 8th LSB bit of $S_{x+1}$ both has changed then extract 0111 and concatenate to $M_1$.
Else if only 7th LSB bit of $S_x$ has changed then extract 1000 and concatenate to $M_1$.
Else if 7th LSB bit of $S_x$ and 8th LSB bit of $S_{x+1}$ has changed then extract 1001 and concatenate to $M_1$.
Else if 7th LSB bit of $S_x$ and $S_{x+1}$ both have changed then extract 1010 and concatenate to $M_1$.
Else if 7th LSB bit of $S_x$ and 7th and 8th LSB bit of $S_{x+1}$ both have changed then extract 1011 and concatenate to $M_1$.
Else if 7th, 8th LSB bit of $S_x$ and 8th LSB bit of $S_{x+1}$ both have changed then extract 1100 and concatenate to $M_1$.
Else if 7th, 8th LSB bit of $S_x$ and 7th LSB bit of $S_{x+1}$ both have changed then extract 1101 and concatenate to $M_1$.
Else if 7th, 8th LSB bit of $S_x$ and 7th, 8th LSB bit of $S_{x+1}$ both have changed then extract 1110 and concatenate to $M_1$.
Else if 7th, 8th LSB bit of $S_x$ and 7th, 8th, 9th LSB bit of $S_{x+1}$ both have changed then extract 1111 and concatenate to $M_1$.

**Step-4:** Set count $= count + 4$. If count $\leq 18$ then go to step 3, otherwise move to step 5.

**Step-5:** Decimlize the 18 bits in $M_i$ and multiply by 7, which is the magnitude of the total embedded message in terms of bits, let it be $n$.

**Step-6:** Reinitialize $k_i$ to blank and for $i = 1 \rightarrow \frac{n-18}{4}$ repeat step 3. Then we get $M_i$ with n-18 bits length.

**Step-7:** Convert the bits of $M_i$ to characters. The extraction is successful.

4. RESULTS

The given method has been compared with the existing bit flipping method [2], in Table 1. The parameters such as PSNR, hiding capacity, quality index (Q.I), and Bit rate (i.e. bits per pixel, BPP), referred from [28, 29] has been considered for comparison. From Table 1, it is concluded that the capacity of proposed bit flipping technique is more than that of existing technique. The capacity of Bit flipping-2 is 1.5 times larger compared to the existing method. The PSNR of the proposed techniques is acceptable i.e. more than 40. If we compare among the three proposed techniques the bit flipping-2 is preferable for higher hiding capacity.

The cover images are shown in Figure 1. (a)-(h) and the resulting images for bit flipping-1, bit flipping-2 schemes respectively are shown in Figure 2. (a)-(h) and Figure 3. (a)-(h), with 2,45,000 bits of data are concealed in each. It is a clear indication that the resulting images are indistinguishable and do not show any identification areas.

(a) Lena  (b) Baboon  (c) Peppers  (d) Boat

---

IJICT Vol. 7, No. 1, April 2018 : 1–7
Digital Image Steganography Using Bit Flipping (Aditya Kumar Sahu)
5. CONCLUSION

This article proposes a modified bit flipping technique for hiding information in an image called as bit flipping-1 and bit flipping-2. The proposed bit flipping-1 offers a capacity of 3 bits for a pair of pixels. The bit flipping-2 offers a capacity of 4 bits for a pair of pixels. Thus the capacities of the proposed techniques are improved. Furthermore, it is quite clear from the resulting images that it is not susceptible to the invader.

REFERENCES

Digital Image Steganography Using Bit Flipping (Aditya Kumar Sahu)