



---

## The Implementation of Hamming Code Using Video Steganography

Alhussain Akoum<sup>a\*</sup>, Wael Moughrabi<sup>b</sup>

<sup>a</sup>Lebanese university, Faculty of technology, Saida, mojamaa El Bahaa,, Saida and 813, Lebanon

<sup>b</sup>Lebanese university, Faculty of technology, mojamaa El Bahaa,, Saida, Lebanon

<sup>a</sup>Email: [a.akoum@ul.edu.lb.com](mailto:a.akoum@ul.edu.lb.com)

<sup>b</sup>Email: [waelmoughrabi92@gmail.com](mailto:waelmoughrabi92@gmail.com)

### Abstract

As time goes on, the internet world grows larger and larger. The massive number of people involved in the internet means that there are more data flying around in cyberspace waiting for someone to receive it; that is where the need for steganography emerges. The role of steganography is to ensure that the necessary transmitted data does not fall in the hands of the wrong person. It hides the vital data inside an image or video without noticeable changes, where only a key provided by the sender allows the receiver to crack open the cover and see the original data. Steganography is often mistaken as a method of cryptography. In fact, they are two different methods, but they can be used together. In cryptography, the observer can detect that there is a hidden message but he doesn't have the required tools to crack it. Hamming code is a type of cryptography and its use in my paper will strengthen the security of this implementation and make it even harder to decipher.

**Keywords:** Cryptography; Encoding; Hamming code; Steganography.

### 1. Introduction

In this paper, I have developed a secure algorithm to transfer a hidden image from one point to another. The image would be watermarked in a video frame, then undergo several encryptions such as hamming code (7,4), bit shifts and other methods all done by special keys, which strengthens the algorithm even more and makes it nearly impossible to crack.

---

\* Corresponding author.

Then the procedure for steganography would begin where the encoded image would be embedded inside the video. To extract this message, the receiver would have to obtain the algorithm in addition to the needed keys from the sender to decipher. All encoding techniques will take place whilst keeping the quality of the message as clear as possible and keeping the changes done to the steganography video undetectable. The word "steganography" is defined as hidden writing and it was originally based in Greece. The term steganography includes two Greek words "steganos" which is translated to protected and "graphein" which means written. [1,2]. The motion of computer graphics characters could be derived from the motion of real people; it results in more realistic computer character animation than if the animation were created manually [3]. Most of the famous steganography methods are based on the usage of images as the cover data for the secret messages [4]. Steganography could be expressed as "secure hiding information technique within a noise; a way to supplement encryption, in a way that the hidden data could not be traced". [5]. They proposed a method that provides a higher similarity between the cover and steganographed pictures is achieved that also yields a better imperceptibility and prevents the possibilities of steganalysis [6]. They proposed a new approach to place hidden information in either LSB of Green or Blue matrix of a specific pixel in carrier image which is decided by the secret key and efficient LSB of Red matrix [7]. They proposed an efficient BCH coding for steganography which is embedding the secret information inside a block of cover data by changing some coefficients. Authors have improved the computational time of the system and the complexity becomes low because of the system's linearity [8]. They proposed a robust data hiding scheme in H.264 compressed video stream, where they have prevented a drift of intra-frame distortion. To give the system more robustness, authors have encoded the message using BCH code before making the embedding process. The host data is the DCT coefficients of the luminance I-frame component. The obtained results have a high quality and robustness [9]. They have produced a high robust method for colour images in transform domain on the base of Reversible Integer Haar Wavelet Transform (RIHWT) and Graph Theory [10]. They proposed a novel image steganography Least Significant Bit (LSB) based technique for RGB images using color space exchanging from RGB to Hue-Saturation-Intensity (HSI). The secret data are embedded in Intensity Plane (I-Plane) of HSI color model using LSB method. Finally, the resultant image is transformed to RGB color model after embedding. [11].

## **2. Procedure**

### ***2.1. Steganography Approach***

In my paper, a video selected by the sender would be cut into frames, then one of these frames would be randomly selected and the message sent will be watermarked inside it by modifying the least significant bits of the frame bytes. The selected video would be cut into YUV-frames, their pixels would be scrambled by a key, and a frame from the video would be randomly selected. The message would then be watermarked inside the random frame using LSB method. The resulted image would be transformed into a binary matrix which would be shifted using a special key to keep the hidden message unreadable. Hamming code (7, 4) would then be applied, each 4 bits of the message would be multiplied by the generator matrix G and the result would be of modulo 2. Although all these algorithms are almost impossible to crack, the resulted bits are to be XORed using key3, to make it even more secure. We will then take each 7 bits of the encoded message and store them in the YUV frames of the video, 3 bits in frameY, 2 in frameU, and 2 in frameV. Now to reconstruct the video holding

the secret message, the frames will be scanned then combined back to RGB frames. The RGB frames are then put together to reform a video containing a secret message. However, no modification to the video is noticeable. To reobtain the message, the receiver would have to receive the video, scramble the pixels by the same key1, then scan the frames to find the data. From this point on, the decryption steps would occur opposing the steps taken to encrypt it. The message will first be XORed back to its original form by key3, then multiplied by the parity check matrix HT. The data would be scanned to find any error in the bits of the message to correct it. Following the hamming correction, the data would then be shifted back to its original position using the same key1, then the matrix would be transformed back to two dimension, and therefore the frame containing the message would be obtained. The bits of the original message would then be extracted from the least significant bits of the frame, and the image is then obtained. After the message is extracted from the frame, it would then be changed to a binary form where it would be shifted by a special key, encoded by Hamming code (7,4), then XORed and embedded into the video using special keys. These intact algorithms would make it nearly impossible for anyone to crack, not to mention that the steganography done to the image would show no trace of modification, so no one would probably know there is a hidden message. All these steps would make sure that the embedded image would not be intercepted by anyone other than the desired receiver.

## 2.2. Watermarking

LSB substitution is the most well-known technique to data hiding in images; It is basically the application of changing the least significant bit in each byte of the file with a single bit for the hidden message so that it could not be detected by the casual observer. It replaces some of the information in the given pixel with information from the data in the image. The embedding capacity increases with the increase of the LSB used. However, the increase of changed bits in the original message may degrade its fidelity and thus the steganography done might be detected. To begin with, a video selected by the sender would be cut into frames, then one of these frames would be randomly selected and the message sent will be watermarked inside it by modifying the least significant bits of the frame bytes. The video I selected 'small.mp4' was cut into 166 YUV-frames, each of them would be scanned row by column and their pixels would be shifted by key1; the U-frame number 32 was randomly selected. The hidden message would be rescaled to be slightly smaller than the frame size, and then its bits would replace the least significant bit of each byte of the selected frame to be watermarked inside it. Here is a proposed scheme on how an image is watermarked into another using LSB method. The following binary representations of the pictures are not accurate since the image representations are very large; the following tables are randomly chosen, used for explanatory purposes. This *Figure 1* is the message to be hidden, it is transformed into binary.



**Figure 1:** Message to hide.

Consider that this is the binary form of the hidden message.

**Table 1:** Binary form of the message to hide.

|          |          |          |          |          |
|----------|----------|----------|----------|----------|
| 10010000 | 10011010 | 10011100 | 10010010 | 10010110 |
| 10100000 | 10011011 | 10011111 | 10100010 | 10000101 |
| 10010000 | 10001101 | 10001101 | 10001010 | 00111101 |

This is the cover image *Figure 2* we wish to hide the data in. It is also converted into binary.



**Figure 2:** Cover image.

Consider this as part of the binary form of the cover image.

**Table 2:** Binary form of the cover image.

|          |          |          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 01111000 | 01111011 | 10000011 | 10010000 | 00110010 | 00111101 | 01001010 | 01011100 |
| 10101010 | 10100111 | 10100111 | 10100110 | 00111101 | 00111011 | 00111000 | 00111011 |
| 01111000 | 01111101 | 10000011 | 10000100 | 00111101 | 00111011 | 00111011 | 00111011 |
| 01111100 | 10000101 | 10000111 | 10000011 | 01011000 | 01001100 | 01001101 | 01001100 |
| 10001010 | 10011001 | 10100111 | 10011010 | 10001011 | .....    | .....    | .....    |

- Take the first byte from the hidden message, [10010000], it is divided into 8 separate bits [1 0 0 1 0 0 0]. The first bit of data to be hidden here is [1].
- 8 bytes from the cover image are taken [0111000 01111011 10000011 ...]
- The first bit of the hidden message data replaces the least significant bit of the first byte of the cover image as shown in the table below:

• Hidden message data:

1 0 0 1 0 0 0 0

• First bit of data to hide\*

• The first byte of the cover image data:

0 1 1 1 1 0 0 0

• Least significant bit\*

- The least significant bit of the byte is replaced with first bit of hidden data

0 1 1 1 1 0 0 1

This process is repeated for all the bytes of the cover image until all the bits of the hidden message have been successfully embedded. Cover image before and after *Figure3*:



**Figure 3:** .(a) Cover image before and (b) after watermark.

When this process is done, the frame containing the secret message will be transformed into binary to continue the encryption processes.

### 2.3. Hamming Code

Hamming code, proposed by Richard Hamming, is a widely used linear code, famous for its intact structure, its ability to detect up to two simultaneous bit errors and for correcting single-bit errors. In this paper, the (7, 4) hamming code is used. It works by encoding four bits of message data (m1 m2 m3 m4) by adding three parity bits (p1 p2 p3) to stack up to a code-word of length seven. The respective data bits assign each parity bit. The hamming correction is done by comparing each parity bit with its respective data bit; the two should achieve an even parity (1).

$$2p \geq p+m+1 \tag{1}$$

Where p is the parity bit and m is the message bit.

The relation (2) between the parity bits and the message bits in hamming code (7, 4).

**Table 3:** Table showing the relation between the parity and message bits in hamming code (7, 4).

|    | m1 | m2 | m3 | m4 |
|----|----|----|----|----|
| p1 | x  | X  |    | x  |
| p2 | x  |    | x  | x  |
| p3 |    | X  | x  | x  |

$$p1 = m1 \oplus m2 \oplus m4$$

$$p2 = m1 \oplus m3 \oplus m4 \tag{2}$$

$$p3 = m2 \oplus m3 \oplus m4$$

The equations (3) used for hamming corrections are:

$$c1 = p1 \oplus m1 \oplus m2 \oplus m4$$

$$c2 = p2 \oplus m1 \oplus m3 \oplus m4 \tag{3}$$

$$c3 = p3 \oplus m2 \oplus m3 \oplus m4$$

If all C bits are zero, then there is no error.

However, if for example we got  $c1=0, c2=1, c3=0$ , we get (010) which is the binary representation of the decimal 2. It is then deducted that the error is in the second bit, so it is switched.

- An example of hamming code (7, 4):

If a message code=1110 then  $m=4$

Using the previous equation,  $2^p \geq p+4+1$  we deduct that  $p=3$

**Table 4:** Message and parity bits of the message code.

|    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|
| 1  | 2  | 3  | 4  | 5  | 6  | 7  |
| p1 | p2 | m1 | p3 | m2 | m3 | m4 |
| p1 | p2 | 1  | p3 | 1  | 1  | 0  |

$$p1 = 1 \oplus 1 \oplus 0 = 0$$

$$p2 = 1 \oplus 1 \oplus 0 = 0 \tag{4}$$

$$p3 = 1 \oplus 1 \oplus 0 = 0$$

Thus, we deduce that the code is 0010110.

There are two famous matrixes used in hamming code (4).

The first one is the generator matrix G used for encoding. After the image is transformed into binary, each 4 bits of the message are multiplied by the generator matrix and there result is applied to a modulo of two (5).

$X=M*G$  where:

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

M: message matrix

X= 7 bit code-word

The second matrix is the parity check matrix H used for decoding. The received message would be multiplied by the transpose of the matrix H, and then the message would be decoded (6).

$$Z = R * HT \text{ where } HT = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \quad (6)$$

The vector Z is composed of three bits (z1,z2,z3). The vector Z should be all zeros ( 0 0 0) for the transmitted message to be correct. Otherwise, a change in the message will require the need for an error correction process. The binary form of the frame containing the secret message is separated into blocks of 4 bits, each matrix of 4 is multiplied by the generator matrix G, where the result of the message bits will be 7 bits including the parity bits. These 7 bits will then be embedded inside the video frames. As for decoding, each block of 7 bits -4 message bits, 3 parity bits- will be multiplied by the transpose of the parity check matrix H. The resulted 3 bits will be checked and undergo hamming correction if necessary.

#### 2.4. Video Embedding

Video embedding is an extra layer of security used to keep our data a secret. Instead of just encrypting the data and keeping them in plain sight, we hide the data in other files so then almost no one knows it's there. To embed data in video, the video should be cut into images called frames, where the bits of the secret data will be distributed among these frames. Then, the frames holding the data would be combined to reform the video, which holds the data, but does not show any difference in the video. For starters, the video selected by the sender will be cut into frames, of type YUV. In my case, the video I've chosen 'small.mp4' was cut into 166 frames, each has a role in carrying the bits of the secret message. The video selected to hold the data is cut into Y U and V frames, then saved in BMP format. They are calculated with the following equations (7):

$$Y = 0.299 * R + 0.587 * G + 0.114 * B$$

$$U = (-0.14713 * R) - (0.28886 * G) + (0.436 * B) \quad (7)$$

$$V = 0.615 * R - 0.51499 * G - 0.10001 * B$$

The number of frames is calculated by the given equation:

- Number of frames= (frame rate \* Video duration)-1

U-frame.



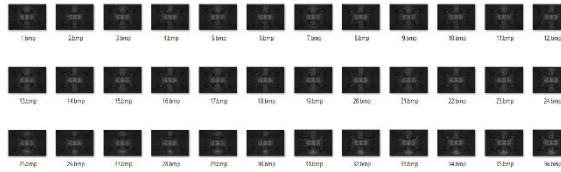
**Figure 4:** Screenshot showing the U-frames.

Y-frames:



**Figure 5:** Screenshot showing the Y-frames.

V-Frames:



**Figure 6:** Screenshot showing the V-frames.

Following the encryption of the data, the resulted 7 bits of the hamming code will be distributed to the video frames for embedding. 3 bits in frame U, 2 bits in frame V and 2 in frame Y. They will be lodged in the frame based on a specially chosen key<sub>3</sub>, which will be needed by the receiver to extract the hidden data. Now that the video frames carry our message, the reverse process is in motion. The YUV frames are combined and transformed into RGB frames. The conversion formulas from YUV to RGB are:

- $R = Y + 1.139834576 * V$
- $G = Y - .3946460533 * U - .58060 * V$
- $B = Y + 2.032111938 * U$

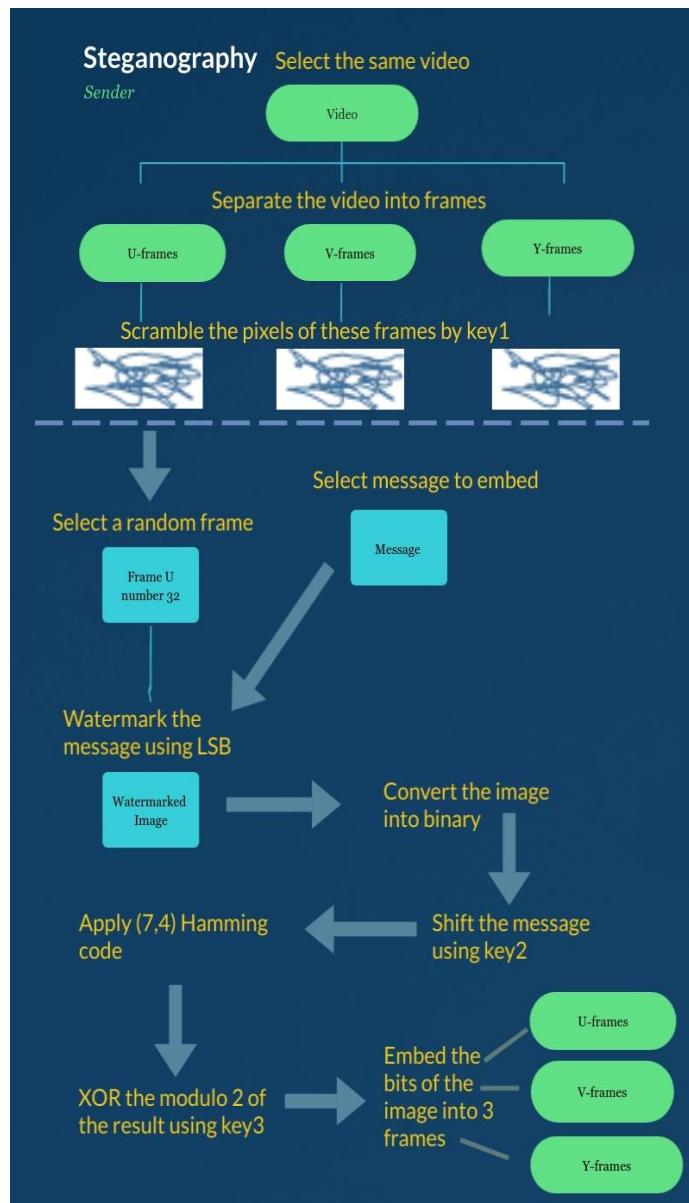




**Figure 7:** Screenshot showing the RGB-frames.

- The frames are scanned to find the embedded data, the same keys are used to decipher them.
- The message would be shifted back by the same key, decoded by hamming code, Xored by the same key2 and then reconstructed.

**Data embedding phase:**



**Figure 8:** Flowchart showing how the message is sent.

- The sender chooses a video.
- The video is split into YUV frames.
- Key1 scrambles the pixels of the frames.
- The sender selects a message to hide, and a random frame is selected from the video.
- The message is watermarked using LSB method.
- The watermarked image converted into binary.
- The message is shifted using a key2.
- Hamming code (7, 4) is applied to the block of code. Each block is transposed then multiplied by the generator matrix G.
- The answers would then be XORed by key3, then there modulo 2 result is taken.
- The embedding process is then launched for each linear block of the code; the message is of 4 bits, which become 7 after the 3 parity bits are added. Three bits are embedded into the U-frames, two into the V-frames and two into Y-frames.

**Data extracting phase:**

- The receiver selects the same video.
- The video is cut into YUV frames.
- Key1 scrambles the pixels of the frames.
- The frames are scanned to find the hidden data.
- The hidden data is XORed with key3.
- Hamming syndrome would be calculated, the r matrix (which has the size of the frames) would be multiplied by  $H^T$ , to decode the message.
- Hamming correction would be applied, the message would be scanned to see if any of its bits are flipped.
- Key2 shifts the message back to its original position.
- The message matrix would then be constructed, and then converted to a 2-Dimensional matrix. In this step, we have obtained the watermarked frame containing the message.
- The Least significant bits of the frame are scanned to regain the message. The secret message is then extracted.

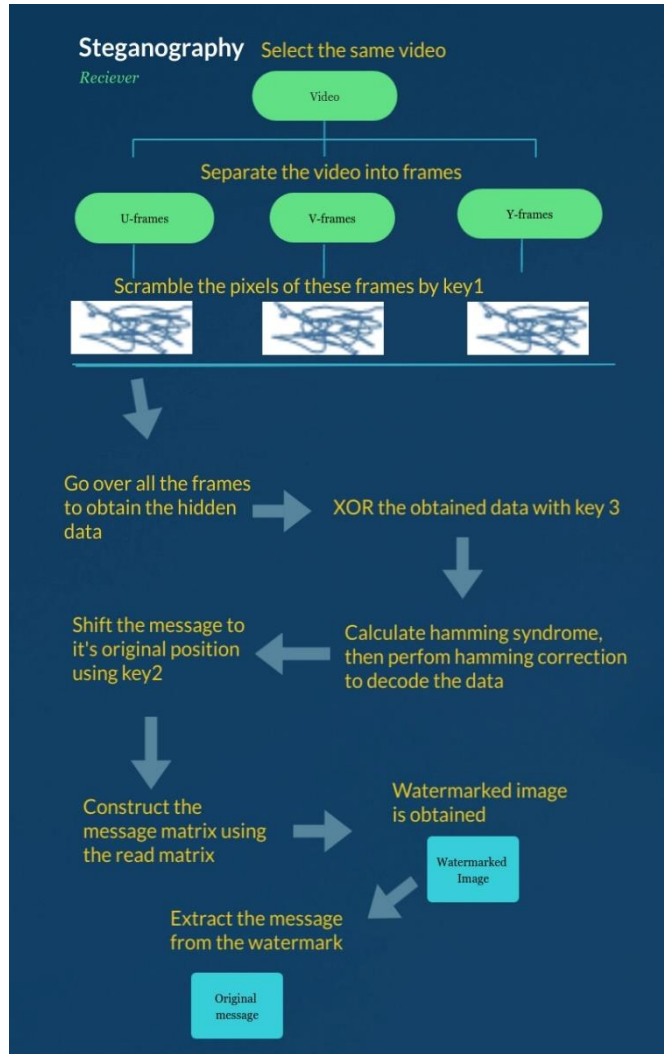


Figure 9: Flowchart showing how the message is received.

### 3. Error Detection

MSE: stands for mean-square error. This ratio also compares between an original image and a reconstructed one. It represents a measure of the peak error. The lower the value of MSE, the lower the error (8).

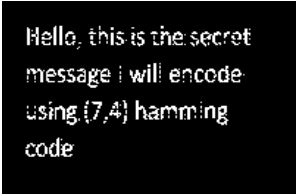
PSNR stands for peak signal to noise ratio. This ratio is used as a quality measurement between an original image and a modified one. The higher is PSNR, the better the quality of the reconstructed image (8).

$$PSNR = 10 \log_{10} \frac{R^2}{MSE} \quad (8)$$

### 4. Results


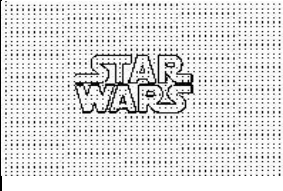
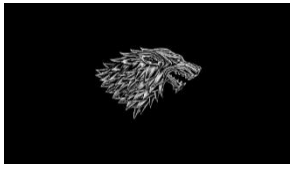
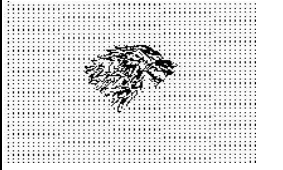
In the first test, I attempted to embed then extract the following text “Hello, this is the secret message I will encode using (7,4) Hamming code” After submitting through the code, undergoing watermark, hamming code, pixel shifting, video embedding then extraction, this message was extracted.

**Table 5:** Secret text message sent and extracted.

| Original image  | Extracted image  |
|---|--|
| <p>Hello, this is the secret message I will encode using (7,4) hamming code</p>             |  |
| <p>Original message sent.</p>   | <p>The message extracted at the end.</p>   |
| <p>The PSNR of the obtained image is 0.3958<br/>The MSE of the obtained image is 0.0694</p> |  |

In the second test, I will attempt to embed an image then extract it. For the code to succeed, the image needs to be a grayscale image. I constructed the code in a way to accept any image, but it will return it in grayscale form.

**Table 6:** Secret images sent and received.

| Original image:   | Extracted image:   |
|---|--|
|  |  |
| <p>PSNR: 0.3587<br/>MSE: 0.2523</p>   |  |
|  |  |
| <p>PSNR: 0.2795<br/>MSE: 0.9380</p>   |  |

## 5. Conclusion

In this paper, the algorithm has shown the ability to transmit images at a very high quality with a low error rate, proving to be an effective way to send secret messages. In addition, the encoding methods applied to this message will be nearly impossible to intercept which would be very useful in these modern days where privacy is a very vital and fragile aspect. The fact that the video is cut into YUV frames with data hidden inside it makes

it very difficult for anyone to detect any change once it is combined back together in RGB form. Even if somehow, someone managed to get their hands on the video, and they knew that it contained secret data he would not have the required knowledge to extract it. The encryption methods done from the key shifting, to the XOR, to the Hamming code (7, 4) all make sure that the information hidden will remain intact and unreachable. In these times, with all the fast data exchange in the World Wide Web, steganography has become a necessity to maintain the highest probability of security possible. Steganography has a very wide field of study, it does not only apply to video and images, and it could apply to audio such as mp3 files. In the future, I will probably upgrade my paper to hide data inside the audio files as well as the video.

## References

- [1]. Sharma, V., and Kumar, S., "A New Approach to Hide Text in Images Using Steganography", *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol 3, No 4, ISSN: 2277 128X, pp.701-708, April 2013.
- [2]. Thiagarajan1, C., Aarthi2, N., "Novel Algorithm for RGB Image Steganography", *International Journal of Computer Science and Mobile Computing*, Vol. 5, No. 4, pp.261 – 270, April 2016.
- [3]. Rosziati Ibrahim and Teoh Suk Kuan, "Steganography Imaging System (SIS): Hiding Secret Message inside an Image", *Proceedings of the World Congress on Engineering and Computer Science Vol I*, pp. 20-22, October 2010.
- [4]. Firas A. Jassim, "A Novel Steganography Algorithm for Hiding Text in Image using Five Modulus Method", *International Journal of Computer Applications*, Vol.72, No.17, pp. 39-44, June 2013.
- [5]. Bhagat, A.R and A.B. Dhembhare. "An Efficient and Secure Data Hiding Technique – Steganography", *International Journal of Innovative Research in Computer and Communication Engineering*, Vol 3, No 2, pp. 941-949, February 2015.
- [6]. Narayana, S and G. Prasad, "Two new approaches for secured image steganography using cryptographic techniques and type conversions". *Signal & Image Processing: An International Journal (SIPIJ)*, Vol.1, No.2, pp. 60-73, December 2010.
- [7]. Karim, S.M., Rahman, M.S and Hossain, M.I., "A new approach for LSB based image steganography using secret", In: 14<sup>th</sup> International Conference on Computer and Information Technology, ICCIT 2011, pp. 286-291.
- [8]. Rongyue, Z. Sachnev, V. Botnan, M. B. Hyounng Joong, K. and Jun, H. "An Efficient Embedder for BCH Coding for Steganography", *IEEE Transactions on Information Theory*, vol. 58, pp. 7272-7279, December 2012.
- [9]. Liu, Y. Li, Z. Ma, X. and Liu, J. "A Robust Data Hiding Algorithm for H. 264/AVC Video Streams", *Journal of Systems and Software*, Vol 86, pp. 2147-2183, 2013.
- [10]. Thanikaiselvan, V. and Arulmozhivarman, P. "High security image steganography using IWT and graph theory". In *Signal and Image Processing Applications (ICSIPA)*, 2013 IEEE International Conference, pp: 337-342.
- [11]. Muhammad, K., Ahmad, J., Farman, H. and Zubair, M., "A novel image stenographic approach for hiding text in color images using HSI colour model". *Middle-East Journal of Scientific Research* Vol 22, No 5, pp. 647-654, May 2015.