An Enhanced Mechanism for Image Steganography Using Sequential Colour Cycle Algorithm

Lip Yee Por[1], Delina Beh[2], Tan Fong Ang[1], and Sim Ying Ong[1]
[1]Faculty of Computer Science and Information Technology, University of Malaya, Malaysia
[2]Malaysian Institute of Information Technology, Universiti Kuala Lumpur, Malaysia

Abstract: *Several problems arise among the existing LSB-based image steganographic schemes due to distortion in a stego-image and limited payload capacity. Thus, a proposed scheme has been developed with the aims to help in improving the payload of the secret data at the same time retaining the quality of the stego-image produced within an acceptance threshold. This study has led to the modification of the current LSB substitution algorithm by delivering a new algorithm namely sequential colour cycle. For achieving a higher security, multi-layered steganography can be performed by embedding a secret data into multiple layers of cover-images. The performance evaluation has been tested and proven that the improvement of embedding ratio at 1:2 for the proposed algorithm can be achieved and the value of the image quality is not falling below the threshold of distortion.*

Keywords: *Image steganography, steganography, least significant bit, data hiding, information hiding.*

Received August 18, 2010; accepted October 24, 2010

1. Introduction

With the vigorous boost of computer network, the Internet revolutionised the boom and dissemination of digital information in our society making it increasingly essential for computer users to find better ways to safeguard personal and confidential information. One common method of securing information is encrypting sensitive data into forms of unreadable strings to prevent unauthorised access and viewing. Unfortunately, encrypted data is bound to draw unnecessary attention assuming that data worth encrypting is likely to be secretive and confidential. Therefore encrypted data is often targeted hence vulnerable to illegal interception and unauthorised tampering during data transmission via public communication channels. To conceal the presence of confidential information, steganography is introduced as an alternative data protection scheme.

Steganography refers to a process of concealing secret data in various forms of digital media such as text, image, audio or video [10]. For instance, if a digital image is utilised as a medium, it is consequently known as a cover-image whereas the altered image after the concealing process is called a stego-image. There are numerous image-based steganographic schemes in existence, but each suffers from stego-image distortion, limited payload capacity and restrictions on embedding file types.

In this research, we investigate ways to improve the embedding ratio in comparison with current image-based steganographic schemes without sacrificing the quality of the stego-image to the extent of noticeable visual distortion. As a result, a new algorithm has been devised primarily to increase capacity of the payload. By employing sequential colour cycle algorithm, our proposed steganographic scheme is able to embed up to four LSBs in a 24bit Windows Bitmap (BMP-24) as well as perform multi-layered encoding and decoding. Most importantly, the proposed scheme assures no increase in the size of the final stego-image and no data loss when retrieving original data.

The remainder of this paper is organised as in the following sections. We will describe the structure of a 24bit BMP-24, various bits for LSB-based method and related LSB-based steganographic tools in section 2. Section 3 will present the design of the proposed scheme as well as the multi-layered encoding and decoding mechanism. In section 4, the experimental result and analysis are discussed. Finally, a conclusion is presented to summarise the deliverables of the proposed scheme.

2. Related Work

Since the aim of the proposed scheme is to improve the embedding ratio in comparison with the current existing method, Morkel *et al*. [19] recommended using lossless images such as BMP and GIF for steganography because lossless compression will not remove any information from the image. In fact, using lossless compression on digital bitmap images keep the original information intact. Lempel-Ziv-Welch (LZW) data compression reduces the size of GIF files without degrading the image visually [16]. Other GIF graphic formats like GIF-16 and GIF-24 are generated by a library known as ANGIF without LZW compression. Both of these GIF extensions mentioned are rarely

used since they are not supported by web browsers [23]. Apparently, the most widely used GIF format supports up to 8bits per pixel which limits its palette to just 256 colours. In steganographic perspective, GIF restricts the types of data file (i.e., only text message) during the embedding process. Since compression plays a crucial role in determining which steganographic algorithm to be used, lossy compression format such as JPEG is not chosen for cover-images [19]. With the exception of transform domain based steganography, lossy compression is avoided because it discards excessive amount of image data consequently destroying properties of the stego-image. BMP-24 on the other hand supports up to 16,777,216 colours giving it an advantage in manipulating colours to embed secret data. Therefore BMP-24 is a favourable format choice for cover-images in the proposed

2.1. BMP-24 As a Cover-Image

BMP file is created by Microsoft and IBM to store Bit-Mapped (BMP) images [3]. Based on analysis from [8] and [29], a BMP file consists of a bitmap-file header, a bitmap-information header, a colour Table and an array of bytes that define the bit-mapped bits. The bitmap-file header contains information about the type, size and layout of a device-independent bitmap file. Each byte in a bitmap file has 8bits that contains colour information on two pixels. Four Most Significant Bits (MSB) define the left pixel and four Least Significant Bits (LSB) define the right pixel.

The composition of the file header as shown in Table 1 has to be identified to perform bit-level manipulation on BMP files so that any incorrect modification of the bit structure can be prevented. The magic number for a BMP file in the offset 0×00 is 0×42 and 0×4D as indicated in Table 2. In the context of computer programming, a magic number refers to a constant used to determine a file type but the significance of the magic number is not apparent without some additional information [15].

Manipulating bits in an image block affects the colours of a BMP image without distorting or degrading image quality. Therefore, the proposed scheme takes advantage of this particularly BMP image file structure trait and embeds secret content into data fields such as the LSB of RGB colour byte. Allowing us to hide an entire file between the information and pixel data without altering the image at all [14].

A BMP colour palette is an array of structures specifying red, green, and blue intensity values for each colour in a display device's colour palette. Each bitmap pixel stores a single value that is used as an index to the colour palette. The colour information stored in the element at that index specifies the colour of that pixel. BMP files have different bit per pixel (bbp), that is the number of bits in each pixel. The values 1, 4, 8, 16, 24, 32 are the only values considered legal by Windows 4x API [17].

Based on bitmap analysis in [3, 17], 24bit BMP has a maximum of 224 colours and the palette field does not contain any element. Therefore, a 24bit BMP file is always stored as 3byte RGB values. Unlike BMP-16 and BMP-32, different Windows platforms support different bitmap colour masks requiring bitmap files to be converted to 24bit for steganography use.

Table 1. BMP file header format - windows BMP files begin with a 54byte header (adapted from [4, 22]).

Offset #	Hex Value	Value	Contents
0	42 4D	"BM"	The magic number used to identify the BMP file
2	46 00 00 00	70 Bytes	The size of the BMP file in bytes
6	00 00	Unused	Reserved.
8	00 00	Unused	Reserved.
10	36 00 00 00	54 Bytes	The offset (starting address) of the byte where the bitmap can be found.
14	28 00 00 00	40 Bytes	Size of Bitmapinfoheader structure, (40 bytes)
18	02 00 00 00	2 Pixels	Image width in pixels (signed integer)
22	02 00 00 00	2 Pixels	Image height in pixels (signed integer)
26	01 00	1 Plane	Number of colour planes being used, must be set to 1
28	18 00	24 Bits	Number of bits per pixel, which is the colour depth of the image. (1, 4, 8, 16, 24, 32)
30	00 00 00 00	0	Compression type (0 = none, 1 = RLE-8, 2 = RLE-4)
34	10 00 00 00	16 Bytes	Size of the raw image data in bytes (including padding)
38	13 0B 00 00	2835 Pixels/meter	Horizontal resolution in pixels per meter
42	13 0B 00 00	2835 Pixels/meter	Vertical resolution in pixels per meter
46	00 00 00 00	0 Colours	Number of colours in image, or zero
50	00 00 00 00	0 Important colours	Number of important colours in image, or zero

Table 2. Magic number of BMP.

Hexadecimal	0×42	0×4D
Symbol	B	M

2.2. Least Significant Bit Substitution

According to Katzenbeisser and Petitcolas [11, 12], using least significant substitution for image steganography is able to optimise the capacity of the payload and not results in a perceptible amount of degradation to the human visual senses [26]. Since LSB appears at the lowest order bit in a binary value, therefore the altered bits can only be discovered with hex editor.

Images is the most wide-spread media in use [2, 25] and when digital images are being utilised as cover in

steganography, the cover-image is generally manipulated by changing one or more bits of a single or multiple bytes in image pixels. The secret data can be stored in the LSB of one (e.g., blue colour) out of the three RGB colour bytes or in the parity of the entire RGB [2]. Various bits for LSB-based steganography have currently been implemented in the existing tool.

2.2.1. Stego One Bit LSB

Stego one bit LSB approach changes only noticthe last bit of the colour byte. Therefore, changing the LSB will only change the integer value of the byte by one. By manipulating the LSB of one of the RGB colour, the effect on the appearance of the image is indiscernible [20]. Figure 1 shows an example of using stego one bit in the blue colour byte.

Figure 1. Example of stego one bit LSB.

2.2.2. Stego Two Bits LSBs

Stego two bits LSBs is an approach that manipulates two LSBs of one of the colours in the RGB value of the pixels to store bits of secret data in the cover-image Please refer to Figure 2. The advantage of stego two bits is the amount of information embedded is twice that of stego one bit LSB approach. Unlike stego one bit, the degradation of stego-image quality using stego two bits LSB is slightly distinct [2].

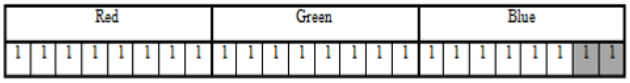


Figure 2. Example of stego two bit LSBs.

2.2.3. Stego Three Bits LSBs

Figure 3 depicts the approach of using three LSBs of one of the colours in the RGB value to hide secret data in the cover-image. The obvious advantage of using more LSBs is increased storage space for secret data. Stego three bits is capable of storing up to three times more secret data than stego one bit LSB approach [2]. Unfortunately, the stego-image quality suffers more detectable degradation in this approach than stego two bits.

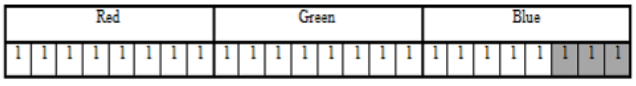


Figure 3. Example of stego three bit LSBs.

2.2.4. Stego Four Bits LSBs

Figure 4 shows the approach of using four LSBs of one the colours in the RGB value to hide secret data. The stego-image quality degrades even more than stego three bits after secret data is embedded, experiencing a disadvantage consistent with the practise of using more LSBs to store secret data. Figure 4 shows an example of using four LSBs to encode secret data.

Figure 4. Example of stego four bit LSBs.

2.2.5. Stego Colour Cycle (SCC)

Using SCC, bits of secret data is embedded in rotating RGB colour values. Bailey and Curran [2] claimed that by using SCC, the presence of hidden data is more challenging to detect and need not subject a single colour to constant change. For instance, the first data bit could be hidden in the LSB of the blue colour byte, the LSB of the red colour byte stores the second data bit and subsequently the third data bit is embedded in the green value. An example of SCC approach is shown in Figure 5.

Figure 5. Example of SCC.

2.3. Related Steganographic Tools Review

Several LSB-based steganographic tools are available on the web. For example, Steganos Privacy Suite 2008, StegoMagic 1.0, StegaNote v3.2 and so forth. Steganos Privacy Suite 2008 by Steganos GmbH Company, is an LSB-based tool capable of hiding and encrypting data. It is a commercialware that has been improved and enhanced since 1996. The cover-mediums used in Steganos Privacy Suite 2008 are BMP-24, JPG and WAV. The advantage of this tool is it's ability in selecting the appropriate cover-object to hide any kind of secret data without any restrictions. Unfortunately, the payload capacity in Steganos Privacy Suite 2008 at approximately 13% of the cover-image size is fairly limited.

StegoMagic 1.0, published by Varun Surech and Shibin. K, Anoop in 2004, has the ability to hide any kind of files or messages in text, wave or BMP-24 cover-mediums. The embedded data can be encrypted using DES for further protection. The cover-image used must be eight times larger than the secret data. By using LSB insertion in StegoMagic 1.0, embedding secret data into the cover-image does not increase its original file size. However, this is not the case when using text files as cover. For instance, when 80kb of secret data is embedded in a cover-text of 462kb, the size of the resulting stego-text becomes 945kb. The tremendous bloat in output size is perceptually noticeable. Moreover, StegoMagic 1.0 requires users to determine the file extension of secret data during the

retrieval process. This requirement confuses users if he/she does not know the secret data's file extension.

StegaNote v3.2, created in 2005 by Dirk Rijimenants, is an improved version of StegaNote v3.1 and one notable feature is the utilisation of random LSB embedding known as Random Pixel Positioning (RPP), a pseudorandom generator producing a series of bits that are scattered in distributed areas of an image. In order to achieve the required diffusion, the author inserted a certain amount of irrelevant information such as a random header. The secret data will be encompassed with a secret key for additional security. BMP file is used as a cover-image where secret data bits are stored in the red, green and blue LSB of the image pixels. StegaNote v3.2 automatically converts various image file formats such as GIF, JPG or TIFF to BMP before processing. Although this feature provides flexibility for users, the capacity of stego-images becomes relatively large after the conversion to BMP format. This will certainly draw suspicion when small stego-images, after having embedded with secret data, show disproportionate large file sizes. In terms of capacity, StegaNote hides three bits of data per pixel. Cover-images must be approximately eight times larger than the size of secret data. For instance, 32KB of secret data can be hidden in a 263KB BMP image. Therefore, users of Steganote have to take into account the appropriate type and size when choosing cover-images for encoding.

3. The Proposed Scheme

In this section, the proposed scheme which employs sequential colour cycle algorithm is presented. The idea behind the proposed algorithm is to combine stego one, two, three, four bits LSBs and stego colour cycle. The pros and cons for each of the method are aforementioned. If stego one bit LSB is used, the payload will be very limited to encode secret data considering only one of the RGB colours is involved in the substitution.

Similarly, if stego two, three or four bits are used, only one of the colours in the RGB value of the pixels will be used and the remaining two colours are not utilised. For instance, although SCC can optimise the LSB of the entire RGB pixel, bit manipulation using one bit is definitely restricting users from encoding more information. Apparently, the proposed scheme is able to encode one bit LSB, two bit LSBs, three bit LSBs and even up to four bit LSBs at each RGB colour pixel depending on the size of the secret data please refer to Figure 6. For instance, if the size of the secret data is 10% of the cover-image, one bit LSB is used to encode the secret data. If the secret data is half the size of the cover-image, then the maximum four LSBs of the cover-image will be encoded with secret data. By using sequential colour cycle algorithm, the entire LSBs are fully utilised to encode secret data of each pixel sequentially after being converted into binary string. Therefore, data files of any type can be embedded in cover-images as secret data without enlarging the stego-image because LSB (or LSBs) is substituted instead of being added. LSB (or LSBs) substitution effectively prevent the sensitivity of the human visual system from detecting contrast change in an image thereby enabling the proposed algorithm to perform a more secured multi-layered embedding without visually degrading the stego-image [14].

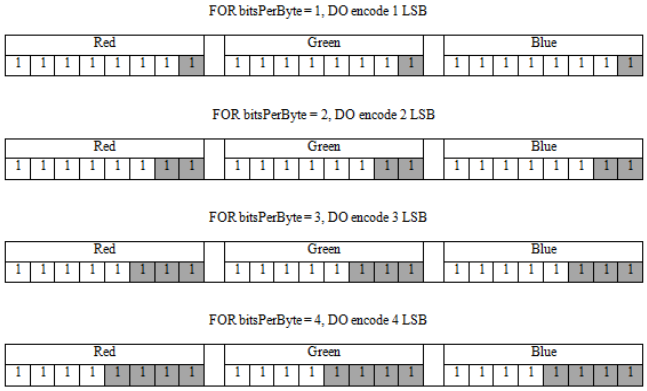


Figure 6. Sequential colour cycle algorithm.

3.1. Encoding Process

The encoding process follows the steps listed below:

- *Step 1:* Calculate the size of the secret data and cover-image.
- *Step 2:* Determine the total bytes from the RGB pixels needed in cover-image to encode the secret data.

$$bits\ per\ pixel = bits\ per\ byte \times 3 \quad (1)$$

$$number\ of\ pixels\ of\ cover = \frac{(cover\ Length \times 8)}{bits\ Per\ Pixel} \quad (2)$$

$$number\ of\ pixels\ of\ data = \frac{(data\ Length \times 8)}{bits\ Per\ Pixel} \quad (3)$$

- *Step 3:* The system reads the bytes of the secret data. Convert the amount of bits of the secret data in (3) from integer into binary string.
- *Step 4:* Segment the pixel into three blocks which are red, green and blue.
- *Step 5:* Get the total LSB or LSBs needed for encoding.
- *Step 6:* Calculate the arrays from the first seven pixels (from the 0^{th} pixel to the 6^{th}pixel) of the bitmap cover-image to store the secret data details as depicted in Table 3. The number of k pixels will be used to store the file name of the secret data.

$$k = n - 7 \quad (4)$$

Denote:

k: Capacity of the file name used in term of pixels.
n: The kth pixel after the 6^{th} pixel. The remaining pixels of the cover-image will be used to store the contents of the secret data.

Table 3. Pixels used for storing secret data.

Pixel (Array [])	Details Stored
0	Number of LSB used
1	The length of the file name
2 to 6	The length of the secret data
7 to *n*	The file name of the secret data
n+1 until *n+k,* where *k* is the capacity of the cover-image (in term of pixels)	The contents of the secret data

- *Step 7:* Calculate the number of LSB or LSBs in the colour pixels of the cover-image that are being used to encode data. Standard integers are stored in four bytes allowing a range of 2^{32} possible values in variations of 2^{8}, 2^{16}, 2^{24} and 2^{32} [9]. Since RGB consists of three channels, the proposed scheme corresponds only with the last three byte values 2^{16}, 2^{24} and 2^{32}. Initially, the unused bits in the RGB colour pixel will be discarded. Then, the remaining bits are used to encode the secret data. For instance, if the selected cover-image is a colour image containing RGB pixels, an equation is computed as below to discard the unused bits:

$$x=8-m \quad (5)$$

Denote:

x: Unused bits after encoding.
m: Bit used to encode data; m=1, 2, 3 and 4.

After obtaining the unused bits from RGB pixels in 5, the value of the original byte will deduct the unused bits which is *x* in order to identify the encoded bits. Therefore, to compute the value of a new pixel with encoded data, the original value of the pixel and the encoded bits will be added together to obtain a new pixel containing embedded data. The following computation obtains the total encoded LSBs and the value of a new pixel with encoded data:

$$Total\ encoded\ LSBs=(s_1- x_1)+(s_2- x_2)+(s_3- x_3) \quad (6)$$

New pixel value with encoded data=original byte+total encoded LSBs.
Denote:

X_1: Unused bit from red pixel.
X_2: Unused bit from green pixel.
X_3: Unused bit from blue pixel.
S_1: Original bit from red pixel.
S_2: Original bit from green pixel.
S_3: Original bit from blue pixel.

Figure 7 demonstrates the flow of the encoding process in the proposed scheme.

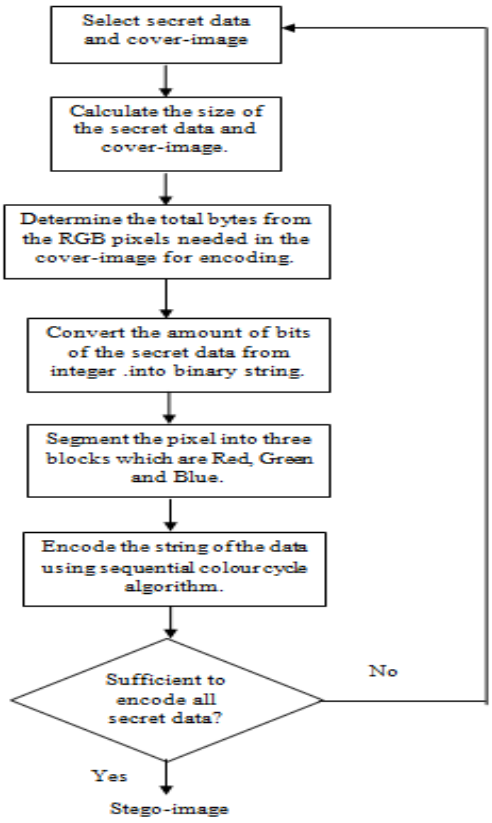


Figure 7. Encoding process.

3.2. Decoding Process

After encoding secret data into the cover-image, the secret data has to be decoded for retrieval. The following are steps for decoding:

- *Step 1:* Calculate the size of the stego-image.
- *Step 2:* The system reads the bytes of the stego-image. Convert the amount of bits of the stego-image from integer to binary string.
- *Step 3:* Determine encoding method for the number of LSBs to be used
- *Step 4:* Calculate and identify the number of pixels in the stego-image and also the embedded secret data respectively using 1, 2 and 3.
- *Step 5:* Use the last three byte values 216, 224 and 232 to obtain the offset of the string. Each of the values mentioned will deduct the LSB used in order to discard unused bits for secret data embedding.

$$first\ offset=16-encoded\ LSB \quad (7)$$

$$second\ offset=24-encoded\ LSB \quad (8)$$

$$third\ offset=32-encoded\ LSB \quad (9)$$

- *Step 6:* Extract and combine the remaining LSB at each pixel in the BMP-24 file in order to retrieve back the original secret data.

$$decoded\ string= (7)+(8)+(9) \quad (10)$$

From 10, the decoded string is the secret data retrieved. Figure 8 shows the decoding process of the proposed scheme when a receiver decodes a stego-image.

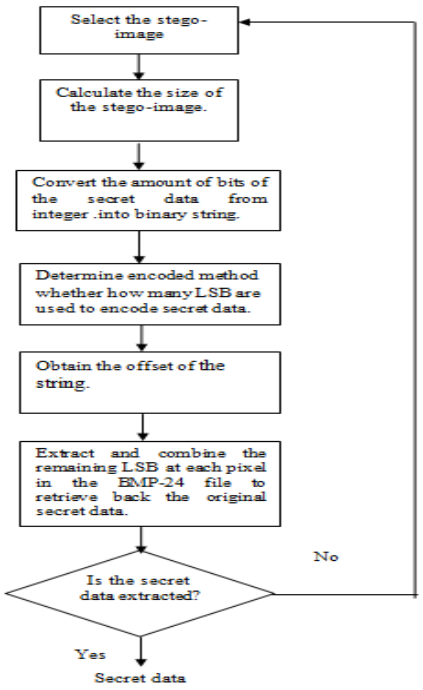


Figure 8. Decoding process.

3.3. Multi-Layered Embedding and Decoding

To attain higher security, the sequential colour cycle LSB algorithm allows a sender to embed secret data into multiple layers of cover-images creating a stealth camouflage. Although multi-layered encoding is possible with Steganos and StegaNote, it is important to note that the proposed scheme is able to improve the embedding ratio at approximately 1:2 when compared to related steganographic LSB-based schemes. Taking into account the size of stego-images increase proportionately to the layer of embedding, the final stego-image of the proposed scheme will always be smaller than the final stego-image produced by related tools. The encoding process of multi-layered encoding is a repetition of the steps shown in Figure 7. When multi-layered encoding is performed, a sample of a video file as illustrated in Figure 9 is embedded into the first cover-image (cover-image A) and produces the first stego-image or first output. The first output is then embedded into the second cover image (cover-image B), then followed by the second output which will be embedded into the last cover-image (cover-image C). The sender can proceed with more layers of embedding as long as the size of the cover-image is two times larger than the previous output. The number of layers embedded is only known by the sender preventing others from retrieving the secret data.

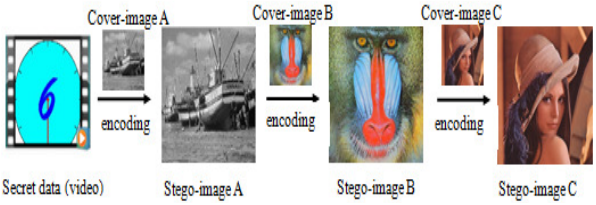


Figure 9. Multi-layered encoding.

When the sender conveys the secret message to a receiver, the sender needs to notify the receiver of the actual layers of cover-images that have been embedded. Only then can the receiver decode the cover-images layer by layer, obtain the original secret data and saving what is produced. The decoding process for multi-layered encoding is an iterative decoding process following steps shown in Figure 8. Figure 10 shows the secret data decoded from multiple layers of cover-images.

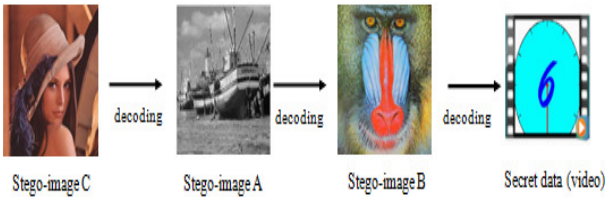


Figure 10. Multi-layered decoding.

4. Experimental Results and Analysis

The proposed scheme has been programmed in J2SE and developed using Java Plug-in technology as well as bundled with Java Runtime Environment (JRE) to provide developers of legacy OLE/COM/ActiveX containers such as word or visual basic the ability to embed and use portable JavaBeans components in the same way they would previously embed and use platform-specific OLE/COM/ActiveX components [21]. The experimental test has been carried out on a 2.0GHz Centrino Duo personal notebook with 2GBs of DDR RAM running Windows Vista operating system.

The proposed scheme is evaluated by two measurements which are the maximum embedding capacity of the payload and the image quality. Embedding capacity of the payload refers to the length of the secret data which can be embedded into a cover-image [5]. Image quality refers to the characteristic of an image that measures the degradation of a perceived image in comparison with the original perfect image [30]. The test images used in the experiment are "Lena" as shown in Figure 11 in sizes 256×256, 512×512 and 1024×1024. Cover-image sizes smaller than 256×256 such as 64×64 and 128×128 are not used in payload testing because rich text format and audio/video files have higher density, therefore unable to be encoded in smaller cover-images. Cover-images which are larger than 1024×1024 are not included in the test because there are no significant changes in the percentage of embedding capacity for each identified schemes beyond this image size.

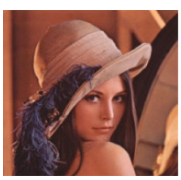


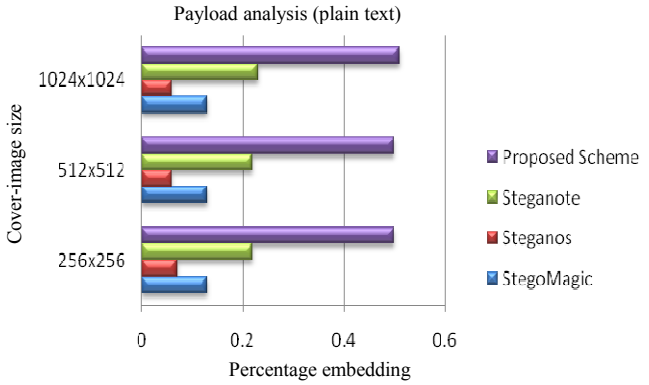
Figure 11. 24Bitmap cover-image used for testing and evaluation.

4.1. Capacity Analysis

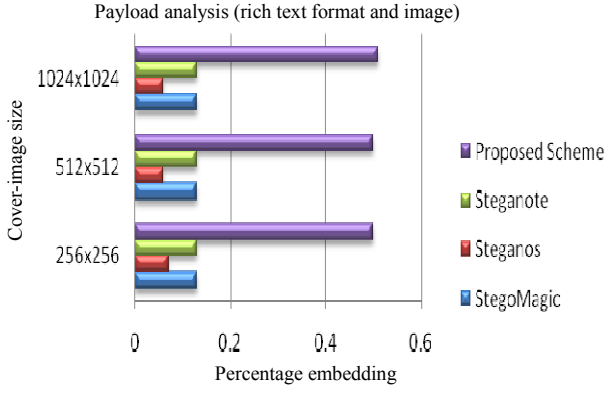
Figure 12 shows the percentage embedding for each identified scheme. In order to measure the maximum capacity, a payload testing has been proceeded with different types of secret data that consists of plain text, combination of rich format text and image as well as audio and video files. Different types of data type regardless non-motion or motion files are tested in the payload experiment to scrutinise the behaviour of the data type after the encoding process. Apparently, the proposed scheme is compared with three other existing LSB-based steganographic tools.

Figure 12-b shows that Steganos Privacy Suite 2008 has the lowest payload at an average of 6%. StegaNote and StegoMagic earn 13% and 16% of payload respectively. The proposed scheme defeats the rest of the existing LSB-based proposed scheme in terms of payload as the proposed scheme has gained an average of 50% even though various types of secret data have been included during the payload testing. Due to the sequential colour cycle algorithm are employed in the proposed scheme, the entire LSBs are fully utilised to maximise the payload capacity at an embedding ratio of 1:2.

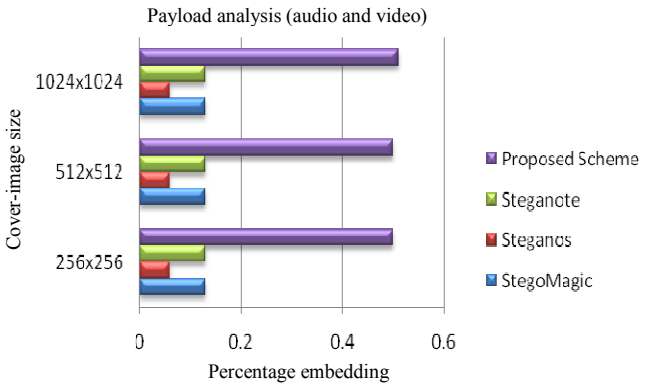
Subsequently, the proposed scheme is tested with the related tools such as StegoMagic 1.0 and StegaNote v3.2 to prove that the proposed scheme is able to outperform among the related tools by obtaining an improvement in terms of capacity at 50% embedding rate.



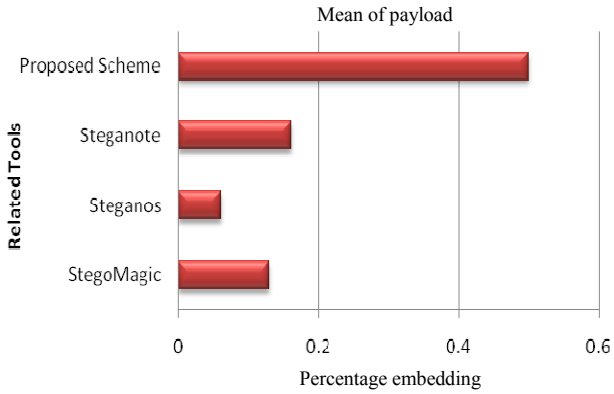
a) Payload analysis using plain text.



b) Payload analysis using rich text format and image.



c) Payload analysis using audio and video.



d) Mean of payload analysis.

Figure 12. Comparison results in terms of maximum capacity for a set of pre-defined image sizes using.

4.2. Stego-Image Distortion Analysis

Peak Signal-to-Noise Ratio (PNSR) is used to measure the quality of the stego-image that produced by the current LSB-based steganographic tool. PNSR is the most popular metric to measure the distortion between an original image and a reconstructed image [5, 6, 13, 18]. Stego-images with sizes of 256×256, 512×512 and 1024×1024 have been tested with the original image in MATLAB using PNSR formula computed by Ponomarenko [24] for measuring a 24-bitmap image with RGB channels. It is defined as follows:

$$MSE = \Sigma_{i=1}^{M} \Sigma_{j=1}^{N} \left(f_{ij} - g_{ij} \right)^2 MN \tag{11}$$

$$PSNR = 10 log_{10} \left(\frac{L^2}{MSE} \right) \tag{12}$$

In order to identify the PNSR of a stego-image, the Mean Square Error (MSE) has to be measured [27]. From 11, wherein M and N are the number of rows and number of columns of the cover-image respectively, f_{ij} is the pixel value from the cover-image, g_{ij} is the pixel value from the stego-image, and L is the peak signal value of the cover-image. For colour images with three RGB values per pixel, the definition of PSNR is the same except the value of MSE is the sum over all squared value differences divided by the image size and by three [24, 28]. According to Shajeemohan [28], a PNSR value that falls below 30dB indicates a fairly

low quality image which is most likely caused by distortion during embedding process. A high quality stego-image should measure 40dB or higher.

Based on the evaluation in Figure 13, stego-images produced by the proposed scheme measures 30dB and above when plain text and combinations of rich format text and images are used in the test as secret data. However, due to higher density of the combination of audio and video files, the quality of stego-images encountered a slight degradation indicated by a fall below threshold value. The degradation phenomenon also occurs when 256×256 or smaller cover-images are used because more pixels are needed for bit manipulation during the encoding process, hence influencing a slight drop in peak signal-to-noise ratio falling below threshold value. However, the quality of the stego-image is still visually indistinguishable to the original image as depicted in Table 4.

Despite the trade-off between embedding capacity and stego-image quality in the proposed method, the results in Figure 14 show that the proposed scheme has obtained an average PNSR value of 31.49dB by embedding the three categories of data type as depicted in Figures 13 and 14. By keeping the quality of stego-images above the threshold of fairly low quality, the visual differences will appear indistinguishable for human visual senses.

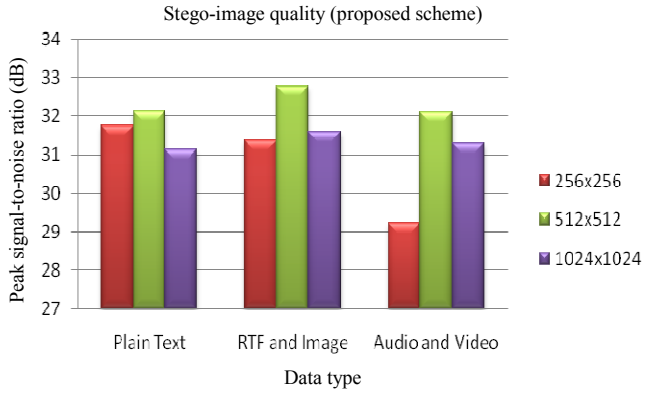


Figure 13. Comparison results in terms of image quality for a set of pre-defined image sizes (256×256, 512×512 and 1024×1024) with different kinds of data type.

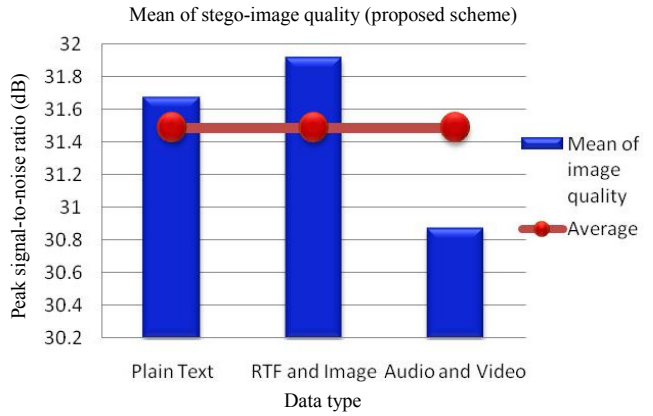
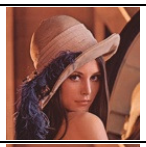
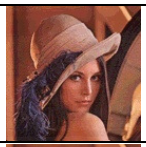
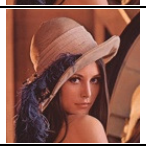
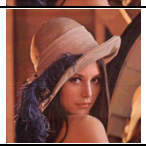


Figure 14. Mean value of the stego-image quality in the proposed scheme.

Table 4. Comparison of an original image and a stego-image using 128×128 and 256×256 cover-image.

Cover-Image Size(Pixel)	Original Image	Stego-Image
128x128		
256x256		

5. Conclusions and Future Work

In this paper, a new algorithm using sequential colour cycle is proposed to optimise the current LSB mechanism by utilising and integrating stego one LSB to stego four LSBs and also stego colour cycle LSB. As a result, the proposed scheme is able to encode up to four LSBs in the each of the RGB pixels according to the contents of the secret data without visually degrading the stego-image.

The proposed scheme contributes a multi-layered embedding feature that enable senders to encode secret data into several cover-images sequentially to create a stealth camouflage to avoid intruder's unwanted attention. Another advantage is the implementation of bit substitution using sequential colour cycle algorithm to ensure the capacity of stego-images remain unchanged despite having multiple layers of encoding and decoding embedded. The proposed scheme also contributes to the flexibility in the selection of any secret data file types as long as the size of secret data is approximately 50% of the cover-image size. Having reviewed comparison with related work, the contributions mentioned in the proposed scheme is proven to outperform most of the current existing LSB-based tools. In a nutshell, the proposed scheme has achieved its envisioned objectives by developing a novel prototype to increase embedding capacity to 1:2 ratio as well as alleviate problems of distortion in stego-images production and restrictions in secret data file types.

Future work will focus on diversifying the various types of cover-medium and to increase the capacity of payload using other alternative methods such suggested in [1]. We strive to explore and discover more options for users in the selection of cover mediums as well as ways to embed more secret data. Besides, we would also like to focus on the integration of information hiding method and authentication method [31, 32, 33].

References

[1] Al-Ataby A. and Al-Naima F., "A Modified High Capacity Image Steganography Technique Based

on Wavelet Transform," *The International Arab Journal of Information Technology*, vol. 7, no. 1, pp. 358-364, 2010.

[2] Bailey K. and Curran K., "An Evaluation of Image Based Steganography Methods Using Visual Inspection and Automated Detection Techniques," *Multimedia Tools and Applications*, vol. 31, no. 3, pp. 55-88, 2006.

[3] BMP Format, available at: http://atlc.sourceforge.net/bmp.html, last visited 2009.

[4] BMP Windows Header Format, available at: http://www.fastgraph.com/help/bmp_header_format.html, last visited 2009.

[5] Chao R., Wu H., Lee C., and Chu Y., "A Novel Image Data Hiding Scheme with Diamond Encoding," *EURASIP Journal on Information Security*, vol. 2009, pp. 1-9, 2009.

[6] Cheddad A., Condell J., Curran K., and Mckevitt P., "Biometric Inspired Digital Image Steganography," *in Proceedings of 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, Belfast, pp. 159-168, 2008.

[7] EasyBMP C++ Bitmap Library, available at: http://easybmp.sourceforge.net/steganography.html, last visited 2009.

[8] Graphics File Formats, available at: http://www.digicamsoft.com/bmp/bmp.html, last visited 2009.

[9] JAR File Specification, available at: http://java.sun.com/j2se/1.5.0/docs/guide/jar/jar.html#Intro, last visited 2009.

[10] Johnson N. and Jajodia S., "Exploring Steganography: Seeing the Unseen," *IEEE Computer Society*, vol. 31, no. 2, pp. 26-34, 1998.

[11] Katzenbeisser S. and Petitcolas P., *Information Techniques for Steganography and Digital Watermarking*, Artech House, London, 2000.

[12] Kipper G., *Investigator's Guide to Steganography*, Auerbach Publications, 2004.

[13] Lee C., Chang C., and Wang K., "An Improvement of EMD Embedding Method for Large Payloads by Pixel Segmentation Strategy," *Image and Vision Computing*, vol. 26, no. 12, pp. 1670-1676, 2008.

[14] Lie W. and Chang L., "Data Hiding in Images with Adaptive Numbers of Least Significant Bits Based on the Human Visual System," *in Proceedings of IEEE International Conference on Image Processing*, Japan, vol. 1, pp. 286-290, 1999.

[15] Magic Number Definition, available at: http://www.linfo.org/magic_number.html, last visited 2009.

[16] Memon N. and Rodila R., "Transcoding GIF Images to JPEG-LS," *IEEE Transactions on Consumer Electronics*, vol. 43, no. 3, pp. 423-429, 1997.

[17] Microsoft Windows Bitmap File Format Summary, available at: http://www.fileformat.info/format/bmp/egff.htm, last visited 2009.

[18] Mittal A., "Low Power Motion Estimation Architecture for MPEG-4 AVC," *Master Thesis*, University of Edinburgh, 2006.

[19] Morkel T., Eloff J., and Olivier M., "An Overview of Image Steganography," *in Proceedings of the 5th Annual Information Security South Africa Conference*, South Africa, pp. 1-11, 2005.

[20] Neeta D. and Snehal K., "Implementation of LSB Steganography and Its Evaluation for Various Bits," *in Proceedings of 1st International Conference on Digital Information Management*, Bangalore, pp. 173-178, 2006.

[21] Oracle, Sun Developer Network, available at: http://java.sun.com/products/plugin/1.3/activex.faq.html, last visited 2009.

[22] OS/2 Bitmap, available at: http://netghost.narod.ru/gff/graphics/summary/os2bmp.htm, last visited 2009.

[23] Philip H., Library: Angif [Internet] 1999, available at: http://angif.slashusr.org, last visited 2009.

[24] Ponomarenko N., Lukin V., Egiazarian K., and Astola J., "DCT Based High Quality Image Compression," *in Proceedings of 14th Scandinavian Conference on Image Analysis*, Finland, pp. 1177-1185, 2005.

[25] Por L., Lai W., Alireza A., Ang T., Su M., and Delina B., "StegCure: A Comprehensive Steganographic Tool Using Enhanced LSB Scheme," *Journal of WSEAS Transactions on Computers*, vol. 7, no. 8, pp. 1309-1318, 2008.

[26] Rahman S. and Syed M., *Multimedia Technologies: Concepts, Methodologies, Tools, and Applications Book Description*, Information Science Publishing, 2007.

[27] Sandipan D., Ajith A., and Sugata S., "An LSB Data Hiding Technique Using Prime Numbers," *in Proceedings of the 3rd International Symposium on Information Assurance and Security*, Manchester, pp. 101-106, 2007.

[28] Shajeemohan S., Govindan K., and Vijilin B., "A Scheme for Image Classification and Adaptive Mother Wavelet Selection," *in Proceedings of International Conference on Advanced Computing and Communications*, Surathkal, pp. 308-313, 2006.

[29] Structure of BMP File, available at: http://www.ue.eti.pg.gda.pl/fpgalab/zadania.spartan3/zad_vga_struktura_pliku_bmp_en.html, last visited 2009.

[30] Torres-Maya S., Nakano-Miyatake M., and Perez-meana H., "An Image Steganography

Systems Based on BPCS and IWT," *in Proceedings of 16th International Conference Electronics Communications and Computers*, USA, pp. 51-51, 2006.

[31] Yee P-L., and Kiah M-L-M., "Shoulder Surfing Resistance Using Penup Event and Neighbouring Connectivity Manipulation," *Malaysian Journal of Computer Science*, vol. 23, no. 2, pp. 121-140, 2010.

[32] Por L-Y, 2013, "Frequency of Occurrence Analysis Attack and Its Countermeasure," *The International Arab Journal of Information Technology*, *Advance online publication*, available at: http://www.ccis2k.org/iajit/ PDF/, vol.10, no.1/4143-7.pdf, vol. 10, no.2, Last Visited 2012.

[33] Por L-Y., Wong K-S., and Chee K-O., "UniSpaCh: A Textbased Data Hiding Method Using Unicode Space Characters." *Journal of Systems and Software*, vol. 85, no. 5, pp. 1075-1082, 2012.

Lip Yee Por is a senior lecturer for the Department of System and Computer Technology in the Faculty of Computer Science and Information Technology at University of Malaya. He received his PhD, BSc and MSc in computer science at University of Malaya, Malaysia. His current research interests include information security, steganography, image processing, graphical authentication, grid computing, and e-learning framework. He is a senior member of IEEE since 2011. His biography has been included in Marquis Who's Who in the World.

Delina Beh received her BSc degree in Information Technology and MSc in Computer Science in 2008 and 2010 at University of Malaya, Malaysia. She is a lecturer in Malaysian Institute of Information Technology, Universiti Kuala Lumpur since 2010. Her research area includes information hiding, cryptography, biometric authentication, mobile security and USB security.

Tan Fong Ang obtained his PhD from University of Malaya in 2012. He is currently a senior lecturer in the Department of Computer system and Technology, Faculty of Computer Science and Information Technology, University of Malaya, Malaysia. He is a Member of IEEE. His research areas include Web Services, cloud computing and game education.

Sim Ying Ong received her BSc in computer science in software engineering at University of Malaya, Malaysia. She is currently pursuing her PhD at University of Malaya, Malaysia.