www.ietdl.org

Published in IET Image Processing
Received on 10th February 2013
Revised on 19th June 2013
Accepted on 14th July 2013
doi: 10.1049/iet-ipr.2013.0109

ISSN 1751-9659

One-third probability embedding: a new ±1 histogram compensating image least significant bit steganography scheme

Saeed Sarreshtedari, Mohammad Ali Akhaee

Department of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran
E-mail: s.sarreshtedari@ut.ac.ir

Abstract: A new method is introduced for the least significant bit (LSB) image steganography in spatial domain providing the capacity of one bit per pixel. Compared to the recently proposed image steganography techniques, the new method called one-third LSB embedding reduces the probability of change per pixel to one-third without sacrificing the embedding capacity. This improvement results in a better imperceptibility and also higher robustness against well-known LSB detectors. Bits of the message are carried using a function of three adjacent cover pixels. It is shown that no significant improvement is achieved by increasing the length of the pixel sequence employed. A closed-form expression for the probability of change per pixel in terms of the number of pixels used in the pixel groups has been derived. Another advantage of the proposed algorithm is to compensate, as much as possible, for any changes in the image histogram. It has been demonstrated that one-third probability embedding outperforms histogram compensating version of the LSB matching in terms of keeping the image histogram unchanged.

1 Introduction

Steganography is a delicate technique in transmitting hidden data through highly-used media in order not to attract any unauthorised person's attention [1]. Since images are nowadays the most widespread digital media on the Internet, most of steganographic algorithms are implemented on images. The least-significant bit (LSB) steganography algorithms work on LSB of cover images. These algorithms are widely used because of high embedding capacity, high quality and low computational complexity.

The most simple and primitive algorithm of image LSB steganography is LSB replacement, in which the LSB of each pixel is replaced with the message bit. In this way, even pixel values are incremented by one or remain unchanged. On the other hand, odd values are decremented by one or kept with no change. As a result, every two consecutive values in the histogram of the pixel values converge to the same number and the histogram of the image will be in a 'pair-wise' format. This pair-wise appearance known as pair of values (PoV) can be detected by chi-square [2] and RS [3] steganalysis tests.

Referring to this Achilles' heel, an improvement called LSB matching (LSBM) was proposed [4]. Here, in order to overcome the PoVs issue, when the LSB of the cover image pixel and the message bit are not matched, the pixel value is decreased or increased by one randomly. More recent and complicated steganalysis techniques have been proposed to detect LSBM embedding. An important group of these analysis techniques are based on the centre of mass of the histogram of the characteristic function (HCF-COM) features proposed by Harmsen and Pearlman [5]. Since HCF-COM method was ineffective for grey-scale images, Ker proposed adjacency and calibrated HCF-COM [6]. Several steganalysis methods have been proposed since then for detecting LSB matching [7–9]. Performance of LSBM was also investigated through universal steganalysis techniques [10–12].

In LSB matching scheme, it is assumed that the message is encrypted, the input bit stream is a random sequence and hence, each bit may match to the LSB of the pixel value with one-half probability. Therefore, in the LSB matching method, the probability of change per pixel is 0.5. A revisited version of LSB matching (LSBMR) was proposed by Mielikainen, which greatly improved the performance by lowering the probability of modification per pixel from 0.5 to 0.375 [13]. Our contribution in this regard is to reduce this probability to one-third. This is why we call our method one-third probability embedding algorithm. As expected, the proposed embedding method will exhibit better performance than LSBM and LSBMR, against HCF-COM steganalysis.

LSBMR and one-third probability embedding work on groups of two and three pixels, respectively. There exist many other steganographic algorithms that work on groups of pixels [14–16]. Increasing the number of pixels used as an embedding group to approach infinity, Li showed that the probability of change per pixel can be reduced to 0.22 [17]. Although the Li's method is elegant in reducing the embedding distortion at the expense of finding suitable groups, it does not have any solution for the histogram

preserving which is the main point to several steganalysers. Several techniques proposed so far are based on LSBM and LSBMR. Sun *et al.* [18] exploited the randomness of changes in LSBM to preserve the histogram of the host image. The main drawback of this method is its low embedding efficiency as it is based on LSBM. Human visual system (HVS) is less sensitive to changes in edges of the image than smooth areas; thus, Luo *et al.* [19] adapted LSBMR to an edge adaptive algorithm. Since LSBM and LSBMR offered the capacity of only one bit per pixel, LSB steganographic algorithms with higher capacity were proposed afterwards [20–25].

Exploiting modification direction (EMD) steganography method introduced a new class of LSB embedding technique [26]. Depending on the size of the pixel groups employed, the capacity of EMD is slightly more than one bit per pixel. Diamond encoding (DE) was proposed to improve the capacity of EMD [27]. Adaptive pixel pair matching steganography method improved the DE in terms of the mean square error (MSE) between the cover and stego-image [28]. Sun *et al.* [29] also proposed another algorithm to improve the capacity of EMD. Several schemes have been presented based on EMD to improve either its payload [30, 31] or imperceptibility [32].

This paper introduces another LSB algorithm called one-third probability embedding, which reduces the probability of change per pixel. The method offers the capacity of exactly one bit per pixel, which can be considered as an improved version of the classic LSBMR. In this case, the stego image is more similar to the cover image, and hence embedding is less detectable via steganalysers like HCF-COM-based techniques. There is also one degree of freedom in the proposed method, which has been exploited to change pixels in proper direction to compensate for mandatory changes and keep the histogram with less variations. Therefore, the proposed technique has high embedding efficiency and also preserves the histogram as much as possible (preliminary results of this work have been partially reported in IEEE Conference on Multimedia and Expo (ICME) 2009 [33]).

The paper is organised as follows: In Section 2, we give a brief introduction to related works. Section 3 details our proposed algorithm and Section 4 presents the effect of the extension of our algorithm to pixel groups of more than three pixels. Numerical results and performance analysis are presented in Section 5. Finally, Section 6 concludes the paper.

2 Related works

2.1 LSB matching revisited

In the LSB matching revisited method [13], the message embedding is performed on a group of two cover pixels at the same time. The grey-level values of those two pixels are assumed to be x_i and x_{i+1}. After the message embedding, the value of ith message bit m_i equals to the LSB of the stego image's ith pixel y_i and the value of the $i+1$th message bit m_{i+1} is a function of y_i and y_{i+1} as given below

$$m_i = \mathrm{LSB}(y_i) \tag{1}$$

$$m_{i+1} = f(y_i, y_{i+1}) \tag{2}$$

where

$$f(y_i, y_{i+1}) = \mathrm{LSB}(\lfloor y_i/2 \rfloor + y_{i+1}) \tag{3}$$

It can be easily proved that this function satisfies the following property (property 1 in [13])

$$f(l-1, n) \neq f(l+1, n) \quad \forall l,\ n \in Z \tag{4}$$

Therefore, when none of m_i and m_{i+1} are matched to LSB(x_i) and $f(x_i, x_{i+1})$, by only one change (add or subtract) in x_i in a way to match the defined function to m_{i+1}, both matching equations can be satisfied. Another property of this function is (property 2 in [13])

$$f(l, n) \neq f(l, n+1) \quad \forall l, n \in Z \tag{5}$$

According to this property, when m_i is matched to LSB(x_i), and m_{i+1} does not match to the function of two pixels, the matching can be realised by adding to or subtracting from x_{i+1} one randomly.

To analyse the performance of this method, we define $\boldsymbol{r} = [r_i, r_{i+1}]$ as

$$\begin{aligned} \boldsymbol{r} &= [r_i, r_{i+1}] \\ &= xor([m_i, m_{i+1}], \quad [\mathrm{LSB}(x_i), f(x_i, x_{i+1})]) \end{aligned} \tag{6}$$

which shows the result of matching prior to embedding. Therefore, there are four possible matching states before data insertion. The examples of these states are shown in Table 1. If $\boldsymbol{r} = [0, 0]$, then both matching equations are satisfied and there exists no need for modification. If $\boldsymbol{r} = [0, 1]$, m_{i+1} must be manipulated. According to Property 2, one change in x_{i+1} is sufficient to satisfy the matching equation. If $\boldsymbol{r} = [1, 0]$, we change x_i such that the function remains unchanged; and if $\boldsymbol{r} = [1, 1]$, we change it in such a way that the function changes as well, according to property 1 in (4).

As shown in Table 1, only three output pixels differ from their corresponding input values through eight. Thus, we can conclude that the probability of change per pixel as the result of embedding is $3/8 = 0.375$.

2.2 LSB compatible substitution steganography

In LSBM, the pixel value is randomly added or subtracted by one, when it is not matched to the message bit. LSBMR exploits this degree of freedom to decrease the probability of change per pixel to 0.375. Another method called LSB substitution compatible steganography (LSCS) [18] uses this randomness to impose less variations into the histogram of the host image. To this end, every pixel value must be changed in a direction to compensate for previous pixel changes if possible. Here, we briefly introduce LSCS

Table 1 Possible states in LSB matching revisited

x_i	4	4	4	4
x_{i+1}	7	7	7	7
m_i	0	0	1	1
m_{i+1}	0	1	0	1
r_i	0	0	1	1
r_{i+1}	1	0	1	0
y_i	4	4	3	3
y_{i+1}	6 or 8	7	7	7

algorithm with which the histogram preserving property of our proposed method will be compared.

LSCS embedding is accomplished after two complete scans of cover images. Comparing the input secret bits with pixel values at the first scan, the number of pixels with value u that must be changed is calculated and recorded in $T(u)$, where u ranges from 0 to 255. We can decide how many of these changes can be made by subtracting or adding. These numbers for each value of u are recorded in two arrays called $L(u)$ and $R(u)$, respectively. $L(u)$ and $R(u)$ are set to zero for all us at the beginning of the second scan, and we will have $T(u) = R(u) + L(u)$ at the end. The LSCS algorithm assigns proper values to $L(u)$ and $R(u)$ in the second scan. These values are set in an optimum way such that $L(u)$ will be as close as possible to $R(u-1)$; that is the number of changes from u to $u-1$, is as close as possible to the number of changes from $u-1$ to u. This property of the algorithm preserves the histogram of the cover image in highest extent possible.

3 One-third probability embedding

3.1 Basics and definitions

In the LSBM, if the message bit does not match to the LSB of the cover pixel, a random change (plus or minus one) in the pixel value will occur. Therefore, with two pixels, there are two of these random states. In LSBMR introduced in Section 2.1, one of these random states is used in order to obtain better embedding efficiency. In case $r_i = 1$, x_i does not change randomly, but it changes in a way to handle the following pixel matching and to prevent the next pixel from being modified, based on the first property of the function. However, when $r_{i+1} = 1$, x_{i+1} is added or subtracted by one randomly. Thus, we still have one random state left which is used in the proposed algorithm to further reduce the change probability. In order to efficiently use the existing randomness in the LSB matching, LSBMR utilises two consecutive pixels where one random state is still left unexploited. To employ this remaining random state, we extend LSBMR to three consecutive pixels. Before introducing the method, we define the minor and major changes as follows:

Definition 1: A minor change in any x is an addition or a subtraction by one, such that $\lfloor x/2 \rfloor$ remains unchanged.

Definition 2: A major change in any x is an addition or a subtraction by one, such that $\lfloor x/2 \rfloor$ changes.

In accordance to these definitions, for any even grey value, minor and major changes are addition and subtraction by one, respectively. On the other hand, these changes are vice versa for odd ones. It can be shown that

$$\text{minor}(x) = \lfloor x/2 \rfloor \times 4 + 1 - x \tag{7}$$

and

$$\text{major}(x) = \lfloor (x+1)/2 \rfloor \times 4 - 1 - x \tag{8}$$

Now using these new terms, we look at LSBMR with another point of view. If $\boldsymbol{r} = [0, 0]$, we have no change. In the case of $\boldsymbol{r} = [0, 1]$, either minor or major changes in x_{i+1} satisfy the f function based on its definition. If m_i is not matched but m_{i+1} is, then x_i should change while f should remain unchanged. This is done by making a minor change to x_i and if $\boldsymbol{r} = [1, 1]$, a major change in x_i can satisfy both matching equations simultaneously. Our algorithm takes three consecutive pixels at a time. The message bits are embedded in the functions of these pixels in such a way that after embedding, we have

$$\begin{aligned} m_i &= f(y_i, y_{i+1}) = \text{LSB}\left(y_i + \lfloor \frac{y_{i+1}}{2} \rfloor\right) \\ m_{i+1} &= f(y_{i+1}, y_{i+2}) = \text{LSB}\left(y_{i+1} + \lfloor \frac{y_{i+2}}{2} \rfloor\right) \\ m_{i+2} &= f(y_{i+2}, y_i) = \text{LSB}\left(y_{i+2} + \lfloor \frac{y_i}{2} \rfloor\right) \end{aligned} \tag{9}$$

where y_i, y_{i+1} and y_{i+2} are pixel values after embedding. Compared to LSBMR, slight change to the definition of the function f is observed here. To analyse the performance, we set a triple result vector $\boldsymbol{r} = [r_i, r_{i+1}, r_{i+2}]$ as an extension of (6)

$$\begin{aligned} \boldsymbol{r} = xor\big(&[m_i, m_{i+1}, m_{i+2}], \\ &[f(x_i, x_{i+1}),\ f(x_{i+1}, x_{i+2}),\ f(x_{i+2}, x_i)]\big) \end{aligned} \tag{10}$$

where x_i, x_{i+1} and x_{i+2} are pixel values before embedding. As a result, there will be eight possible states for the value of $\boldsymbol{r}$. If there is only one 1 in the $\boldsymbol{r}$ vector, one minor change in the corresponding pixel value will satisfy the function. If there are two 1s, a major change in the variable which is common in the two functions will satisfy both functions at the same time. Finally, in case where $\boldsymbol{r}$ has triple 1s, a major change in x_i and a minor change in x_{i+1} will be sufficient to convince all the three functions. Table 2 shows all these possibilities. The value of the first column is the decimal description of the vector $\boldsymbol{r}$. The last three columns indicate which change needs to be made to that pixel. It is observed from Table 2 that only eight output values differ from their corresponding inputs values through 24 states. Thus, the probability of change per pixel is $8/24 = 1/3$.

It is interesting to note that there exists still one random decision case among these eight states, which happens in the situation corresponding to the last row of Table 2. In other words, in case when $\boldsymbol{r} = [1, 1, 1]$, that is, none of three matching equations are satisfied prior to embedding, all these three decisions can be alternatively taken to satisfy the three matching equations simultaneously: (major(x_i), minor(x_{i+1})), (major(x_{i+1}), minor(x_{i+2})), and (major(x_{i+2}), minor(x_i)). Now, we explain how this degree of freedom can be used in the one-third probability embedding method to compensate for modifications in the histogram of the image in the best possible form.

Table 2 Different states and their corresponding changes in one-third probability embedding

R	r_i	r_{i+1}	r_{i+2}	x_i	x_{i+1}	x_{i+2}
0	0	0	0	none	none	none
1	1	0	0	minor	none	none
2	0	1	0	none	minor	none
3	1	1	0	none	major	none
4	0	0	1	none	none	minor
5	1	0	1	major	none	none
6	0	1	1	none	none	major
7	1	1	1	major	minor	none

doi: 10.1049/iet-ipr.2013.0109

Despite the case 8, in all other cases when a change must happen – that is the cases corresponding to rows 2–7 of Table 2 – this change is mandatory and there exists no randomness left. We define $N \times N$ change matrix $\boldsymbol{C}$, where $N-1$ equals maximum allowed value for pixels and entries of $\boldsymbol{C}$ are all set to zero at the beginning. Rows and columns of $\boldsymbol{C}$ are numbered from 0 to $N-1$. Embedding is performed in two distinct phases: mandatory decisions and random decisions. In mandatory phase, cover image is scanned and $\boldsymbol{r}$ vectors are calculated. Then decision over all $\boldsymbol{r}$ vectors with at least one zero – where the changes are mandatory – are taken according to rows 2–7 of Table 2. Suppose that we have to change a value from m to $m+1$ in this phase. We increment and decrement entries $(m+1, m)$ and $(m, m+1)$ of the $\boldsymbol{C}$ matrix by one respectively, which means that later we will need an $m+1$ to m change to compensate this modification and keep the histogram of the stego image the same as that of the cover one. At the end of this phase, all mandatory changes in the cover image are done and the $\boldsymbol{C}$ matrix has non-zero entries only in the superdiagonal and the subdiagonal. Almost 7/8 of the secret message is embedded during this mandatory phase. In the following random phase, we just consider the case $\boldsymbol{r} = [1, 1, 1]$, where three different sets of changes can satisfy all three matching equations. For each of such cases, we calculate the 'score' for each of these three changing sets. The score is calculated by adding up the sign of the entries of the $\boldsymbol{C}$ matrix corresponding to the two possible alterations. For better illustration, the score calculation is explained through an example here. Assume pixel intensities take values from 0 to 3, and after the first phase of embedding, the entries of the change matrix are as below:

$$\boldsymbol{C} = \begin{pmatrix} 0 & 4 & 0 & 0 \\ -4 & 0 & 2 & 0 \\ 0 & -2 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{11}$$

Now suppose that at the beginning of the second embedding phase, three message bits equal $m_1 = 0$, $m_2 = 1$ and $m_3 = 1$ have to be embedded in three pixels with values equal $x_1 = 1$, $x_2 = 1$ and $x_3 = 2$, respectively. Therefore

$$\begin{aligned} r = xor\big([0, 1, 1], \mathrm{LSB}([(1 + \mathrm{LSB}(\lfloor 1/2 \rfloor)), \\ (1 + \mathrm{LSB}(\lfloor 2/2 \rfloor)), \quad (2 + \mathrm{LSB}\big(\lfloor 1/2 \rfloor\big))])\big) = [1, 1, 1] \end{aligned} \tag{12}$$

All one r-vector implies that matching equation can be satisfied through three different set of decisions: (i) a major change in x_1 and a minor change in x_2, (ii) a major change in x_2 and a minor change in x_3 and (iii) a major change in x_3 and a minor change in x_1. Now we calculate the scores for these three sets and pick up the best.

$$\begin{aligned} &1{:}\,\mathrm{Major}(x_1){:}1 \rightarrow 2,\ \mathrm{Minor}(x_2){:}1 \rightarrow 0 \\ &\Rightarrow \mathrm{Score}(1) = \mathrm{sign}(C(1, 2)) + \mathrm{sign}(C(1, 0)) = 0 \\ &2{:}\mathrm{Major}(x_2){:}1 \rightarrow 2,\ \mathrm{Minor}\big(x_3\big){:}2 \rightarrow 3 \\ &\Rightarrow \mathrm{Score}(2) = \mathrm{sign}(C(1, 2)) + \mathrm{sign}(C(2, 3)) = 2 \\ &3{:}\mathrm{Major}(x_3){:}2 \rightarrow 1,\ \mathrm{Minor}\big(x_1\big){:}1 \rightarrow 0 \\ &\Rightarrow \mathrm{Score}(3) = \mathrm{sign}(C(2, 1)) + \mathrm{sign}(C(1; 0)) = -2 \end{aligned} \tag{13}$$

Consequently, the best decision is

$$y_1 = x_1 = 1,\ y_2 = \mathrm{Major}\big(x_2\big) = 2,\ y_3 = \mathrm{Minor}\big(x_3\big) = 3 \tag{14}$$

Note that score '2' for this set of variations indicates that these two changes can compensate for two other changes already happened in the mandatory phase in the cover image histogram. In fact, this is a better pair of actions than the other two sets which cannot compensate (set 1) or even deteriorate (set 3) the histogram variations caused by the mandatory phase. It is important to consider the effect of this compensation in the change matrix by refreshing the entries corresponding to a decision already taken and updating the matrix $\boldsymbol{C}$ as follows

$$\boldsymbol{C} = \begin{pmatrix} 0 & 4 & 0 & 0 \\ -4 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{15}$$

As is seen, every change influences two entries of $\boldsymbol{C}$, same as the mandatory phase. Now suppose there exists more than one change set with maximum score. In such situations, set of changes with maximum sum value of the corresponding entries in $\boldsymbol{C}$ has priority. For example, consider the $\boldsymbol{C}$ matrix as below

$$\boldsymbol{C} = \begin{pmatrix} 0 & -4 & 0 & 0 \\ 4 & 0 & 2 & 0 \\ 0 & -2 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{16}$$

Calculating scores of all three changing sets results in 2, 2 and 0 this time. But considering the second condition, we calculate the secondary scores for the first two changing sets as $2+4=6$, and $2+1=3$, respectively. Therefore in this case, we choose the first action set. Proper changes in $\boldsymbol{C}$ are applied accordingly. If the change sets were the same in the second condition as well, the decision is taken randomly.

3.2 Embedding algorithm

Now considering the above explanation, the procedure of our embedding algorithm can be stated as follows.

Step 1 (Initialisation): The cover image is copied into the stego one. Pixels at extremities of $N-1$ and 0 are decreased and increased by one, respectively. The $N \times N$ matrix $\boldsymbol{C}$ where $N-1$ is the maximum value allowed for pixel values is defined and all of its entries is set to zero.

Step 2 (First scan): The cover image is divided into groups of three pixels (x_i, x_{i+1}, x_{i+2}) which are called embedding units. $((x_i, x_{i+1}, x_{i+2})$ are the same as (y_i, y_{i+1}, y_{i+2}) at the beginning). The order of selecting pixels is determined by a secret key shared between the transmitter and receiver; therefore, no one else can extract the secret message without having the secret key and the security of the algorithm is guaranteed. According to the details of the method explained above, three secret bits can be embedded into each embedding unit. Thus, three secret bits (m_i, m_{i+1}, m_{i+2}) are fetched from the input for each embedding unit. (y_i, y_{i+1}, y_{i+2}) will contain values for corresponding pixels in the stego image after data embedding. For each

embedding unit (x_i, x_{i+1}, x_{i+2}) and its corresponding secret bits (m_i, m_{i+1}, m_{i+2}), the result of matching vector is calculated as (10). This calculation is repeated until the result vector is derived for all embedding units. Based on the corresponding result vector, the embedding units are divided into three groups. If result vector is all zero, the embedding unit is added to unchanged list. The pixels in this list will remain unchanged. Those embedding units leading to a result vector of one or two 1s along with corresponding secret bits will be added to mandatory change list. One certain pixel among three must endure a certain minor or major change. Finally, the all-one result vector leads its corresponding embedding unit to the random change list. Three possibilities are available for each member of this list to satisfy the matching conditions.

Step 3 (Mandatory embedding phase): For each embedding unit in this list, pixel values and corresponding secret bits are fetched. The pixel values are changed according to the following six cases:

$$\begin{aligned}&\text{Case\#1: } \boldsymbol{r} = [1, 0, 0]:\\ &y_i = \text{Minor}(x_i),\\ &C(y_i, x_i) = C(y_i, x_i) + 1,\\ &C(x_i, y_i) = C(x_i, y_i) - 1.\end{aligned}$$

$$\begin{aligned}&\text{Case\#2: } \boldsymbol{r} = [0, 1, 0]:\\ &y_{i+1} = \text{Minor}(x_{i+1}),\\ &C(y_{i+1}, x_{i+1}) = C(y_{i+1}, x_{i+1}) + 1,\\ &C(x_{i+1}, y_{i+1}) = C(x_{i+1}, y_{i+1}) - 1.\end{aligned}$$

$$\begin{aligned}&\text{Case\#3: } \boldsymbol{r} = [1, 1, 0]:\\ &y_{i+1} = \text{Major}(x_{i+1}),\\ &C(y_{i+1}, x_{i+1}) = C(y_{i+1}, x_{i+1}) + 1,\\ &C(x_{i+1}, y_{i+1}) = C(x_{i+1}, y_{i+1}) - 1.\end{aligned}$$

$$\begin{aligned}&\text{Case\#4: } \boldsymbol{r} = [0, 0, 1]:\\ &y_{i+2} = \text{Minor}(x_{i+2}),\\ &C(y_{i+2}, x_{i+2}) = C(y_{i+2}, x_{i+2}) + 1,\\ &C(x_{i+2}, y_{i+2}) = C(x_{i+2}, y_{i+2}) - 1.\end{aligned}$$

$$\begin{aligned}&\text{Case\#5: } \boldsymbol{r} = [1, 0, 1]:\\ &y_i = \text{Major}(x_i),\\ &C(y_i, x_i) = C(y_i, x_i) + 1,\\ &C(x_i, y_i) = C(x_i, y_i) - 1.\end{aligned}$$

$$\begin{aligned}&\text{Case\#6: } \boldsymbol{r} = [0, 1, 1]:\\ &y_{i+2} = \text{Major}(x_{i+2}),\\ &C(y_{i+2}, x_{i+2}) = C(y_{i+2}, x_{i+2}) + 1,\\ &C(x_{i+2}, y_{i+2}) = C(x_{i+2}, y_{i+2}) - 1.\end{aligned} \tag{17}$$

The score matrix ($\boldsymbol{C}$) is passed to the next phase.

Step 4 (Random embedding phase): For all embedding units in this list, we have $\boldsymbol{r} = [1, 1, 1]$. Therefore, for each embedding unit in this list, the scores of these three possible changes are calculated; the action set with the highest score is taken and the matrix $\boldsymbol{C}$ is updated as explained before

$$\begin{aligned}&\text{Action set \#1: } y_i = \text{Major}(x_i),\ y_{i+1} = \text{Minor}(x_{i+1})\\ &\text{Action set \#2: } y_{i+1} = \text{Major}(x_{i+1}),\ y_{i+2} = \text{Minor}(x_{i+2})\\ &\text{Action set\#3: } y_{i+2} = \text{Major}(x_{i+2}),\ y_i = \text{Minor}(x_i)\end{aligned} \tag{18}$$

Step 5 (Stego image generation): The processed pixels from all three lists are integrated and located in their original coordinates to generate the stego image. A block diagram of the embedding phase is sketched in Fig. 1.

3.3 Extraction algorithm

The stego image is first divided into groups of three pixels (y_i, y_{i+1}, y_{i+2}) which we call embedded units. The order of selecting pixels is determined using the secret key. Three secret bits are extracted from each embedded unit according to (9). The process is continued until all embedded bits are detected. The simple block diagram of the extraction phase is shown in Fig. 2.

4 Extension of the proposed method to the groups of more than three

Assuming the one-third probability embedding algorithm as a level-three extension of a general LSB technique, its

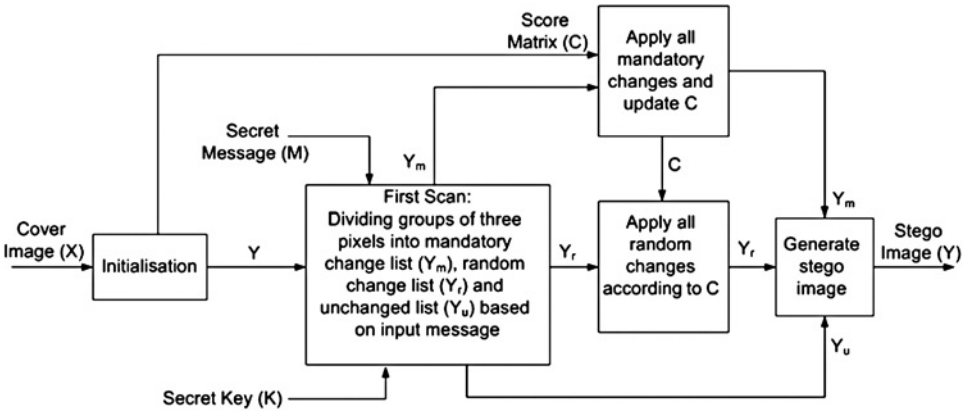

Fig. 1 *Block diagram of the embedding phase*

doi: 10.1049/iet-ipr.2013.0109

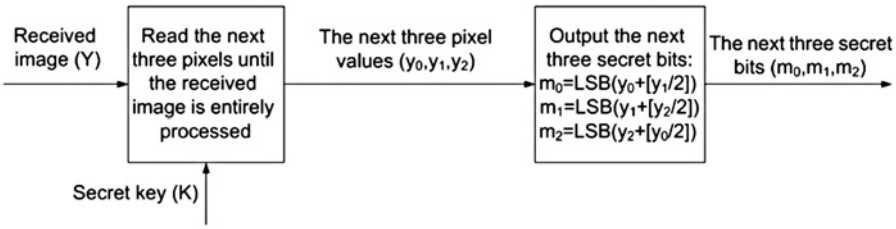

Fig. 2 *Block diagram of the extraction phase*

level-four extension can be shown as

$$m_i = f(y_i,\ y_{i+1}) = \mathrm{LSB}\left(y_i + \lfloor\frac{y_{i+1}}{2}\rfloor\right)$$
$$m_{i+1} = f(y_{i+1}, y_{i+2}) = \mathrm{LSB}\left(y_{i+1} + \lfloor\frac{y_{i+2}}{2}\rfloor\right)$$
$$m_{i+2} = f(y_{i+2}, y_{i+3}) = \mathrm{LSB}\left(y_{i+2} + \lfloor\frac{y_{i+3}}{2}\rfloor\right)$$
$$m_{i+3} = f(y_{i+3}, y_i) = \mathrm{LSB}\left(y_{i+3} + \lfloor\frac{y_i}{2}\rfloor\right) \quad (19)$$

In general form, the level-k extension of this algorithm is given as

$$m_i = f(y_i, y_{i+1}) = \mathrm{LSB}\left(y_i + \lfloor\frac{y_{i+1}}{2}\rfloor\right)$$
$$m_{i+1} = f(y_{i+1}, y_{i+2}) = \mathrm{LSB}\left(y_{i+1} + \lfloor\frac{y_{i+2}}{2}\rfloor\right)$$
$$\vdots$$
$$m_{i+k-2} = f(y_{i+k-2}, y_{i+k-1}) = \mathrm{LSB}\left(y_{i+k-2} + \lfloor\frac{y_{i+k-1}}{2}\rfloor\right)$$
$$m_{i+k-1} = f(y_{i+k-1}, y_i) = \mathrm{LSB}\left(y_{i+k-1} + \lfloor\frac{y_i}{2}\rfloor\right) \quad (20)$$

Here, we still have the $\boldsymbol{r}$ vector with its new definition similar to its level-three form. Assuming that there exist two consecutive '1's in this vector, only one major change in the common pixel can satisfy both corresponding matching equations due to the extended functions defined in (20). Therefore, we should find these consecutive '1's and remove them by a series of major changes in the most suitable form. Afterwards, every '1' which is left isolated ('1' with two '0' at both sides) can be handled via a minor change in the corresponding pixel. Note that in this analysis, the change matrix and histogram compensating ability are ignored for the sake of simplicity. The generalised level-k embedding procedure can be summarised as below:

Step 1: Copy the cover image into stego image. All of the mentioned changes in the next steps are done on this stego image. Construct the k function of k consecutive pixels according to (20). Continue with other steps until the last message bit is embedded.
Step 2: Get k input bits and k corresponding stego image pixels and calculate the appropriate result vector $\boldsymbol{r}$ with k binary elements describing the matching results.
Step 3: Find the first pair of consecutive '1's in $\boldsymbol{r}$. Consider $\boldsymbol{r}$ as a circular vector. It means $\boldsymbol{r}(k)$ and $\boldsymbol{r}(1)$ can be a pair too. Go to step 5 if there is no such a pair.
Step 4: Make a major change to the common pixel of the pair of equations corresponding to this pair of 1s in the $\boldsymbol{r}$ vector. Then flip both bits in the $\boldsymbol{r}$ vector to zero and go to step 3.
Step 5: Look for the first '1' left isolated in $\boldsymbol{r}$. Stego image is produced and embedding is over, if there exists no isolated '1'.
Step 6: Make a minor change to the first pixel in equation corresponding to this isolated '1' in the $\boldsymbol{r}$ vector. Then flip it to zero and go to step 5.

As an example, consider the level-7 embedding algorithm and its embedding function. Assume that we are in step 3. Stego image has been initialised with corresponding values of the cover image after copying, and the $\boldsymbol{r}$ vector equals to [1011101]. We begin with $\boldsymbol{r}(1)$. As seen, there exists no '1' after. The next '1' is found in $\boldsymbol{r}(3)$ forming a pair of consecutive 1s with $\boldsymbol{r}(4)$. Hence, we have a major change in y_{i+3}, which is a common pixel to the third and fourth equations of (20). The new vector is $\boldsymbol{r} = [1000101]$ and we are again in step 3. The next '1' is isolated but there is a '1' in $\boldsymbol{r}(7)$ with immediately following '1' in $\boldsymbol{r}(1)$. Therefore, we have a major change in y_i which is a common pixel to the first and last equations in (20). The resulted $\boldsymbol{r}$ is [0000100]. There will be nothing left to do in step 3. There exists an isolated '1' in $\boldsymbol{r}(5)$ found in step 5 which imposes a minor change in y_{i+4}. Now the matching equations are satisfied.

To see the performance of the proposed technique, we have calculated the probability of change per pixel for up to 20 levels. The results are shown in Fig. 3. The level-2 algorithm is another interpretation of the LSB matching

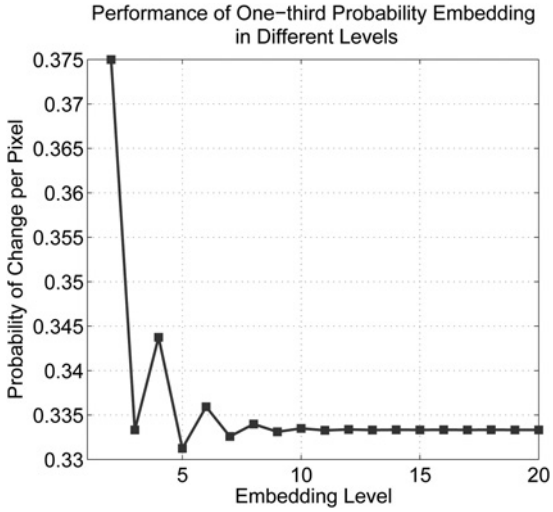

Fig. 3 *Performance of one-third probability embedding against embedding level*

revisited using these two functions

$$m_i = f(y_i, y_{i+1}) = \mathrm{LSB}\left(y_i + \lfloor \frac{y_{i+1}}{2} \rfloor\right)$$
$$m_{i+1} = f(y_{i+1}, y_i) = \mathrm{LSB}\left(y_{i+1} + \lfloor \frac{y_i}{2} \rfloor\right) \quad (21)$$

where level-1 extension results in the LSB matching.

As observed, we have a little improvement (about 0.2%) for level-5, compared to the level-3 algorithm described in Section 3. As illustrated in Fig. 3, the total trend of performance characteristic (average changes per pixel) against the level increasing is a damping oscillation around 1/3. To avoid inessential complexity, we neglect this 0.2% improvement and implement the original one-third probability embedding described in Section 3. A closed-form expression for the probability of change per pixel in terms of the number of embedding unit pixels is derived in Appendix 1.

5 Experimental result

5.1 Embedding distortion

Steganographic algorithms are required to be designed in a way to leave the least possible distortion on the cover image quality, that is, no trace of manipulation must be perceivable in the stego image compared to the original one. There are plenty of works measuring the similarity between the cover and stego images. In this experiment, we compare the distortion of different algorithms by mean square error (MSE) and peak signal-to-noise ratio (PSNR), defined as below

$$\mathrm{MSE} = \frac{1}{\mathrm{MN}} \sum_{i=1}^{M} \sum_{j=1}^{N} (C(i,j) - S(i,j))^2 \quad (22)$$

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{\mathrm{Max}^2}{\mathrm{MSE}} \right) \quad (23)$$

where M and N stands for the height and width of the images, $C(i, j)$ and $S(i, j)$ represent the corresponding cover and stego image pixels, and the Max is 255 for 8-bit images. Assuming p as the probability of change per pixel for any LSB algorithm, we have

$$\mathrm{MSE} = \frac{1}{\mathrm{MN}} (p \times \mathrm{MN}) = p \quad (24)$$
$$\Rightarrow \mathrm{PSNR} = 10 \log_{10} \left(\frac{255 \times 255}{p} \right)$$

It was shown that the probability of change per pixel equals to 0.5, 0.5, 0.375 and 0.333 for LSBM, LSCS, LSBMR and one-third probability embedding, which results in the average PSNR of 51.1411, 51.1411, 52.3905 and 52.9064, respectively. From the four proposed sample images listed in Table 3, we see that the resulting PSNR are well agreed with the expected value calculated theoretically. Since the PSNR of approximately 35 dB is assumed as the threshold where the HVS notices the distortion, we can be sure that all of the proposed methods are completely invisible as shown in Fig. 4. In this figure, the original sample images are shown in the left column, while a secret random message of the rate of one bit per pixel is embedded into them using the proposed method and the resulting stego images are shown in the right column.

Table 3 PSNR values resulting from different algorithms and images

	Lena	Peppers	Baboon	Cameraman
LSBM	51.1444	51.1439	51.1440	51.1357
LSCS	51.1444	51.1439	51.1440	51.1357
LSBMR	52.3887	52.3919	52.3858	52.3831
one-third	52.9024	52.8968	52.8963	52.9310

5.2 Reducing the probability of change per pixel

Here, the efficiency of our proposed method is compared with the LSBM and LSBMR [13] methods. Four thousand colour JPEG images of size 400×300 are used and converted into 8-bit grey-scale images. Two of the best known detectors for the LSBM technique are calibrated and adjacency HCF-COM introduced by Ker [6]. These features are improvements of Harmsen's HCF-COM detector [5], which does not work neither on grey-scale images nor coloured ones. Receiver operation characteristics (ROC) of these two detectors for three embedding methods are derived and compared in Figs. 5 and 6. These figures are generated using different threshold values for discrimination between the clean and stego images. Each threshold value results in a certain pair of probability of detection-false alarm on the ROC curve. From Figs. 5 and 6, it is perceived that the ROC curve of our proposed method stands lower than the other two methods, which results in less probability of detection by steganalysers of this kind. This result was expected since the HCF-COM detectors work on the histogram of pixel values which is left more intact in our algorithm with 1/3 average change per pixel as compared to 3/8 and 1/2 in the LSBMR and LSBM embedding, respectively. Table 4 supports the above facts. In our experiment, there exist four groups of images: The cover images and the stego images of LSBM, LSBMR, and one-third probability embedding algorithms. There are three features for each group. Each row of this table shows what ratio of each group is more similar to corresponding covers out of whole images and according to feature specified in that row. For example, suppose that we have 100 cover images and 100 stego images from each one of three embedding methods. Now assume that we calculate normal HCF-COM feature for all 100 cover images and 300 stego images. For each cover image, we find corresponding stego image with the most similar normal HCF-COM feature, and record the embedding method resulted in the stego image. The first row and first column entry of Table 4 illustrates that nearly 9% of stego images generated via LSBM has normal HCF-COM feature which is more similar to the corresponding cover image than stego images generated from the same cover image using LSBMR and one-third probability embedding methods. This number is called similarity percent. From this table, it is seen that the features of one-third embedded images are closer to that of the original ones. Using our proposed algorithm, the normal HCF-COM looks more natural than that of two other features. This is the reason why one-third probability method shows better resistance against the calibrated HCF detector which uses normal HCF-COM of the image than adjacency HCF-COM [6].

doi: 10.1049/iet-ipr.2013.0109

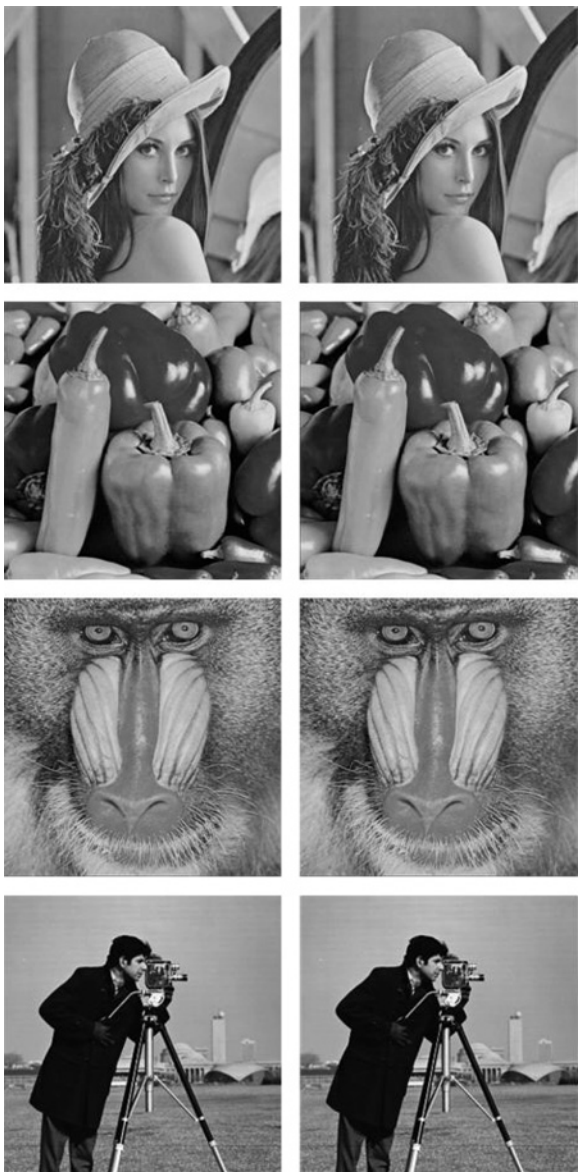

Fig. 4 *Cover images (left) and corresponding stego images (right) generated through one-third probability embedding algorithm with full capacity (1bpp)*

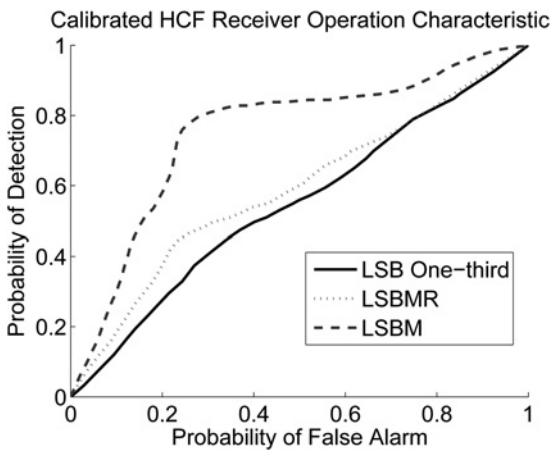

Fig. 5 *ROC curves for the calibrated HCF COM*

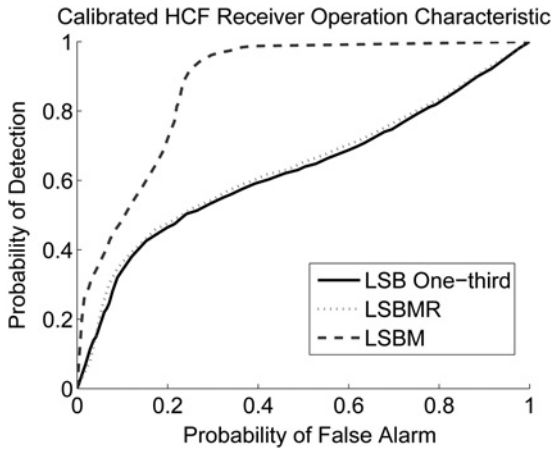

Fig. 6 *ROC curves for the adjacency HCF COM*

Table 4 Similarity percent of three stego-image groups and original image for each feature

	LSB matching, %	LSB revisited, %	LSB one-third, %
normal HCF-COM	8.74	24.55	66.71
calibrated HCF-COM	17.74	38.67	43.59
adjacency HCF-COM	8.17	40.95	50.88

5.3 Histogram compensating characteristic of one-third probability embedding

In this section, the histogram preserving characteristic of the one-third probability embedding is evaluated. Here, this characteristic is compared to LSCS algorithm [18] that was introduced in Section 2.2. Secret data are embedded in 50 grey-scale 512×512 Jpeg images via four different methods. The criteria has been the number of uncompensated changes in the histogram of image after embedding with full capacity, which equals to off-centre diameter sum value of $\boldsymbol{C}$ matrix, that can be stated as

$$\text{uncompensated\_changes} = \frac{\sum_{i=0}^{255} \left| h_c(i) - h_s(i) \right|}{2} \tag{25}$$

where h_c and h_s are histograms of the cover and stego images, respectively. Each uncompensated change makes two histograms to differ in two bins. This is why a factor of two is seen in the denominator.

The first technique is the one-third probability embedding. The second is LSBMR, the third is LSCS and the last one is LSBM, that is actually LSCS without histogram compensating. The improvement in LSCS method compared to its non-compensating version is clear in Fig. 7. This figure presents the number of uncompensated changes for 50 images in which a secret message is embedded at the rate of one bit per pixel using four different methods. Although LSCS has a higher probability of change per pixel compared to LSBMR, but it outperforms LSBMR in

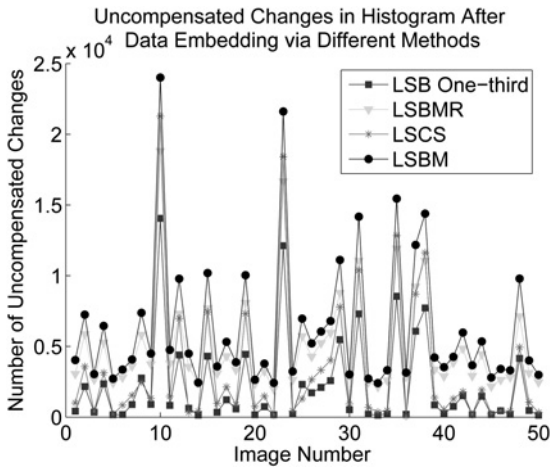

Fig. 7 *Uncompensated changes after embedding via four different methods*

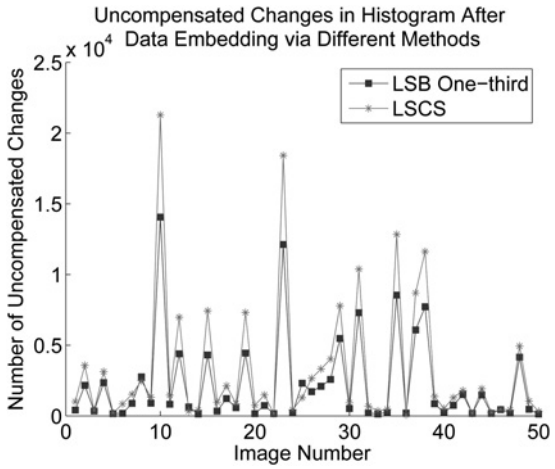

Fig. 8 *Uncompensated changes after embedding via LSCS and one-third probability embedding with compensation ability*

terms of histogram preserving. In the following, we just keep the curves of LSCS and our proposed methods for more accurate comparison.

From Fig. 8, it can be seen that the proposed method outperforms LSCS in terms of histogram preserving. Thus, one-third probability embedding outperforms LSCS and LSBMR, in both reducing probability of change per pixel and histogram compensating. Averaging over whole set of 50 images, the average uncompensated histogram changes are derived as 2210, 5062, 3273 and 6374, for one-third probability embedding, LSBMR, LSCS and LSBM, respectively. From these average values, it can be deduced that both one-third and LSCS methods act better in having less manipulation at the histogram of the cover image than other two methods not compensating the histogram. Besides, one-third probability embedding outperforms LSCS in terms of histogram preserving.

6 Conclusion

In this paper, we introduced a novel ± 1 image LSB steganography algorithm with capacity of one bit per pixel. As the result of reducing the probability of change per pixel, in the same capacity, the stego image pixels are more similar to the corresponding cover pixels. This feature makes the proposed method more imperceptible and also less detectable against well-known steganalysers such as HCF-COM-based methods compared to LSBM and LSBMR. This algorithm also exploits one degree of freedom, which is used to keep the histogram of the stego image unchanged more possibly. This advantage will help the proposed method to show a better resistance against histogram-based detectors. Since the extension of pixel groups from two pixels in LSBMR to three pixels in one-third probability embedding reduces the probability of change per pixel, and it is demanding to investigate if there is much more improvement to extend the one-third probability algorithm for more than three pixels. Therefore, we found a closed-form expression for the probability of change per pixel for any level-k extension of our generalised algorithm. It is proved that three is optimum value for the number of pixels in the groups.

7 References

1 Petitcolas, F., Anderson, R., Kuhn, M.: 'Information hiding-a survey', *Proc. IEEE*, 1999, **87**, (7), pp. 1062–1078
2 Westfeld, A., Pfitzmann, A.: 'Attacks on steganographic systems'. *Proc. Third Int. Workshop on Information Hiding*, 1999, **1768**, pp. 61–76
3 Fridrich, J., Goljan, M., Du, R.: 'Detecting LSB steganography in color and gray-scale images', *IEEE Multimedia*, 2001, **8**, (4), pp. 22–28
4 Ker, A.D.: 'Improved detection of LSB steganography in grayscale images'. Lecture Notes in Computer Science, 2005, vol. 3200, pp. 583–592
5 Harmsen, J.J., Pearlman, W.A.: 'Steganalysis of additive-noise modelable information hiding'. Society of Photo-Optical Instrumentation Engineers (SPIE) Conf., June 2003, vol. 5020, pp. 131–142
6 Ker, A.D.: 'Steganalysis of LSB matching in grayscale images', *IEEE Signal Process. Lett.*, 2005, **12**, (6), pp. 441–444
7 Li, X., Zeng, T., Yang, B.: 'Detecting LSB matching by applying calibration technique for difference image'. Proc. 10th ACM Workshop on Multimedia and Security, 2008, pp. 133–138
8 Pevny, T., Bas, P., Fridrich, J.: 'Steganalysis by subtractive pixel adjacency matrix', *IEEE Trans. Inf. Forensics Sec.*, 2010, **5**, (2), pp. 215–224
9 Huang, F., Li, B., Huang, J.: 'Attack LSB matching steganography by counting alteration rate of the number of neighbourhood gray levels'. IEEE Int. Conf. Image Processing, 2007. ICIP 2007, 2007, vol. 1, pp. 401–404
10 Farid, H.: 'Detecting hidden messages using higher-order statistical models'. 2002 Int. Conf. Image Processing, 2002, vol. 2, pp. 905–908
11 Wang, Y., Moulin, P.: 'Optimized feature extraction for learning-based image steganalysis', *IEEE Trans. Inf. Forensics Sec.*, 2007, **2**, (1), pp. 31–45
12 Li, B., Huang, J., Shi, Y.Q.: 'Textural features based universal steganalysis'. Society of Photo-Optical Instrumentation Engineers (SPIE) Conf., 2008, vol. 6819
13 Mielikainen, J.: 'LSB matching revisited', *IEEE Signal Process. Lett.*, 2006, **13**, (5), pp. 285–287
14 Fridrich, J., Filler, T.: 'Practical methods for minimizing embedding impact in steganography'. Society of Photo-Optical Instrumentation Engineers (SPIE) Conf., 2007, vol. 6505
15 Westfeld, A.: 'F5-A steganographic algorithm'. Lecture Notes in Computer Science, 2001, vol. 2137, pp. 289–302
16 Fridrich, J., Soukal, D.: 'Matrix embedding for large payloads', *IEEE Trans. Inf. Forensics Sec.*, 2006, **1**, (3), pp. 390–395
17 Li, X., Yang, B., Cheng, D., Zeng, T.: 'A generalization of LSB matching', *IEEE Signal Process. Lett.*, 2009, **16**, (2), pp. 69–72
18 Sun, H.M., Wang, K.H., Liang, C.C., Kao, Y.S.: 'A LSB substitution compatible steganography'. TENCON 2007 - IEEE Region 10 Conf., November 2007, pp. 1–3
19 Luo, W., Huang, F., Huang, J.: 'Edge adaptive image steganography based on LSB matching revisited', *IEEE Trans. Inf. Forensics Sec.*, 2010, **5**, (2), pp. 201–214
20 Chan, C.K., Cheng, L.: 'Hiding data in images by simple LSB substitution', *Patt. Recognit.*, 2004, **37**, (3), pp. 469–474

doi: 10.1049/iet-ipr.2013.0109

21 Yang, C.H.: 'Inverted pattern approach to improve image quality of information hiding by LSB substitution', *Patt. Recognit.*, 2008, **41**, (8), pp. 2674–2683
22 Wu, D.C., Tsai, W.H.: 'A steganographic method for images by pixel-value differencing', *Patt. Recognit. Lett.*, 2003, **24**, (910), pp. 1613–1626
23 Zhang, X., Wang, S.: 'Vulnerability of pixel-value differencing steganography to histogram analysis and modification for enhanced security', *Patt. Recognit. Lett.*, 2004, **25**, (3), pp. 331–339
24 Yang, C.H., Weng, C.Y., Wang, S.J., Sun, H.M.: 'Adaptive data hiding in edge areas of images with spatial LSB domain systems', *IEEE Trans. Inf. Forensics Sec.*, 2008, **3**, (3), pp. 488–497
25 Wu, H.C., Wu, N.I., Tsai, C.S., Hwang, M.S.: 'Image steganographic scheme based on pixel-value differencing and LSB replacement methods', *IEE Proc. Vis. Image Signal Process.*, 2005, **152**, (5), pp. 611–615
26 Zhang, X., Wang, S.: 'Efficient steganographic embedding by exploiting modification direction', *IEEE Commun. Lett.*, 2006, **10**, (11), pp. 781–783
27 Chao, R.M., Wu, H.C., Lee, C.C., Chu, Y.P.: 'A novel image data hiding scheme with diamond encoding', *EURASIP J. Inf. Sec.*, 2009, **2009**, pp. 1–9
28 Hong, W., Chen, T.S.: 'A novel data embedding method using adaptive pixel pair matching', *IEEE Trans. Inf. Forensics Sec.*, 2012, **7**, (1), pp. 176–184
29 Sun, H.M., Weng, C.Y., Lee, C.F., Yang, C.H.: 'Anti-forensics with steganographic data embedding in digital images', *IEEE J. Sel. Areas Commun.*, 2011, **29**, (7), pp. 1392–1403
30 Lee, C.F., Chang, C.C., Wang, K.H.: 'An improvement of EMD embedding method for large payloads by pixel segmentation strategy', *Image Vis. Comput.*, 2008, **26**, (12), pp. 1670–1676
31 Hong, W., Chen, T.S., Shiu, C.W.: 'A minimal Euclidean distance searching technique for Sudoku steganography'. Int. Symp. Information on Science and Engineering, 2008. ISISE '08, December 2008, vol. 1, pp. 515–518
32 Wang, J., Sun, Y., Xu, H., Chen, K., Kim, H.J., Joo, S.H.: 'An improved section-wise exploiting modification direction method', *Signal Process.*, 2010, **90**, (11), pp. 2954–2964
33 Sarreshtedari, S., Ghotbi, M., Ghaemmaghami, S.: 'One-third probability embedding: Less detectable LSB steganography'. IEEE Int. Conf. Multimedia and Expo, ICME 2009, 2009, pp. 1002–1005

8 Appendix 1

8.1 Probability of change per pixel for different number of pixels in embedding units

A closed-form expression for the probability of change per pixel at the kth level of the proposed steganography algorithm is deduced in this section. In all definitions and proofs, the k-ary vector $\boldsymbol{v}$ is assumed to be circular. An r-length *run* in $\boldsymbol{v}$ consists of r consecutive one enclosed by two zeros. For any k-ary vector $\boldsymbol{v}$, there are 2^k combinations where each vector can carry *runs* with the lengths of 1, 2 and k. Now we consider a *run* with the length of r in $\boldsymbol{v}$. A major change happens if two consecutive 1s are converted to 0, and if there were a '1' left, it will be converted to 0 as a minor change. Therefore, for the sake of matching, the *runs* with the length of 1 and 2 will be changed in one position; the *runs* with the length of 3 and 4 will change in two positions and so forth. As a matter of fact, an r-length *run*, will cause $\lceil r/2 \rceil$ changes where $\lceil r \rceil$ is the smallest integer greater than or equal to r. To derive the probability of change per pixel at the kth level of the algorithm, the total number of changes in different combinations of $\boldsymbol{v}$ is divided by $k \times 2^k$. Assuming n_r as the number of r-length *runs*, the total number of changes at the level k equals to

$$N = \sum_{r=1}^{k} n_r \lceil r/2 \rceil \tag{26}$$

Lemma 1: For an integer number, $\lceil r/2 \rceil$ equals to

$$\lceil \frac{r}{2} \rceil = \frac{r}{2} + \frac{1 + (-1)^{r+1}}{4} \tag{27}$$

Proof: Case 1: Even numbers: In this case, $(1 + (-1)^{r+1})/4 = 0$, thus, (27) reduces to $r/2$ and is correct.
Case 2: Odd numbers: The proof is the same. □

Lemma 2: The number of r-length *runs* for a k-ary circular vector where $r \leq k$ equals to

$$n_r = \begin{cases} k \times 2^{k-r-2}, & r < k-1 \\ k, & r = k-1 \\ 1, & r = k \end{cases} \tag{28}$$

Proof: With the assumption of $r \leq k$, for $r = k$ and $r = k-1$, the proofs are straightforward. For $r < k-1$, there will be r consecutive 1s enclosed by two 0s. Consequently, there exists 2^{k-r-2} consequent 1s starting from anywhere in vector $\boldsymbol{v}$ which in total, we will have $k \times 2^{k-r-2}$ runs. □

Now by considering Lemma 1, (26) can be rewritten as

$$\begin{aligned} N &= \lceil \frac{k}{2} \rceil + k \lceil \frac{k-1}{2} \rceil \\ &\quad + \sum_{r=1}^{k-2} k \times 2^{k-r-2} \times \lceil \frac{r}{2} \rceil = A + B \end{aligned} \tag{29}$$

where

$$A = \lceil \frac{k}{2} \rceil + k \lceil \frac{k-1}{2} \rceil \tag{30}$$

and

$$B = \sum_{r=1}^{k-2} k \times 2^{k-r-2} \times \lceil \frac{r}{2} \rceil \tag{31}$$

The variables A and B are calculated separately and eventually will be substituted in (29). We use Lemma 1, in order to calculate the values of A and B

$$\begin{aligned} A &= \lceil \frac{k}{2} \rceil + k \lceil \frac{k-1}{2} \rceil \\ &= \frac{k}{2} + \frac{1 + (-1)^{k+1}}{4} + k\left(\frac{k-1}{2} + \frac{1 + (-1)^{k}}{4} \right) \\ &= \frac{k}{2} + \frac{1}{4} + \frac{(-1)^{k+1}}{4} + \frac{k^2}{2} - \frac{k}{2} + \frac{k}{4} + \frac{k(-1)^k}{4} \\ &= \begin{cases} \frac{1}{4} - \frac{1}{4} + \frac{k^2}{2} + \frac{k}{4} + \frac{k}{4} = \frac{k^2 + k}{2} & k \text{ even} \\ \frac{1}{4} + \frac{1}{4} + \frac{k^2}{2} + \frac{k}{4} - \frac{k}{4} = \frac{k^2 + 1}{2} & k \text{ odd} \end{cases} \end{aligned} \tag{32}$$

and

$$\begin{aligned} B &= \frac{1}{4}\times k\times 2^k \times \sum_{r=1}^{k-2}\left(\frac{1}{2}\right)^r\left(\frac{r}{2}+\frac{1}{4}+\frac{(-1)^{r+1}}{4}\right) \\ &= \frac{1}{4}\times k\times 2^k \times \left(\sum_{r=1}^{k-2} r\left(\frac{1}{2}\right)^{r+1}\right. \\ &\quad \left. +\sum_{r=1}^{k-2}\left(\frac{1}{2}\right)^{r+2}-\sum_{r=1}^{k-2}\left(\frac{-1}{2}\right)^{r+2}\right) \\ &= \frac{1}{4}\times k\times 2^k\times (C+D-E) \end{aligned} \tag{33}$$

To calculate C, we use the following lemma:

Lemma 3:

$$\sum_{r=0}^{k-3}(r+1)\left(\frac{1}{2}\right)^r = 4\left(1-k\left(\frac{1}{2}\right)^{k-1}\right) \tag{34}$$

Proof:

$$\sum_{r=0}^{k-3}x^{r+1} = x\sum_{r=0}^{k-3}x^r = \frac{x-x^{k-1}}{1-x} \tag{35}$$

Finding the derivative of (35) yields

$$\sum_{r=0}^{k-3}(r+1)x^r = \frac{\left(1-(k-1)x^{k-2}\right)(1-x)+x-x^{k-1}}{(1-x)^2} \tag{36}$$

Now, by substituting x with 1/2, we have

$$\begin{aligned} &\sum_{r=0}^{k-3}(r+1)\left(\frac{1}{2}\right)^r \\ &= \frac{(1/2)-(k-1)(1/2)^{k-1}+(1/2)-(1/2)^{k-1}}{(1/4)} \\ &= 4\left(1-k\left(\frac{1}{2}\right)^{k-1}\right) \end{aligned} \tag{37}$$

Proof is complete. □

Now using Lemma 3

$$\begin{aligned} C &= \sum_{r=1}^{k-2} r\left(\frac{1}{2}\right)^{r+1} = \frac{1}{4}\sum_{r=1}^{k-2} r\left(\frac{1}{2}\right)^{r-1} \\ &= \frac{1}{4}\sum_{r=0}^{k-3}(r+1)\left(\frac{1}{2}\right)^r = 1-k\left(\frac{1}{2}\right)^{k-1} \end{aligned} \tag{38}$$

The calculations of D and E are more straightforward

$$\begin{aligned} D &= \sum_{r=1}^{k-2}\left(\frac{1}{2}\right)^{r+2} = \frac{1}{8}\sum_{r=1}^{k-2}\left(\frac{1}{2}\right)^{r-1} = \frac{1}{8}\sum_{r=0}^{k-3}\left(\frac{1}{2}\right)^r \\ &= \frac{1}{8}\frac{1-(1/2)^{k-2}}{(1/2)} = \frac{1}{4}\left(1-\left(\frac{1}{2}\right)^{k-2}\right) \end{aligned} \tag{39}$$

$$\begin{aligned} E &= \sum_{r=1}^{k-2}\left(\frac{-1}{2}\right)^{r+2} = \frac{-1}{8}\sum_{r=1}^{k-2}\left(\frac{-1}{2}\right)^{r-1} = \frac{-1}{8}\sum_{r=0}^{k-3}\left(\frac{-1}{2}\right)^r \\ &= \frac{-1}{8}\frac{1-(-1/2)^{k-2}}{(3/2)} = \frac{-1}{12}\left(1-\left(\frac{-1}{2}\right)^{k-2}\right) \end{aligned} \tag{40}$$

Therefore, the value of B equals to

$$\begin{aligned} B &= \frac{1}{4}\times k\times 2^k\times\left(1-k\left(\frac{1}{2}\right)^{k-1}+\frac{1}{4}-\left(\frac{1}{2}\right)^k\right. \\ &\quad \left.+\frac{1}{12}-\frac{1}{3}\left(\frac{-1}{2}\right)^k\right) \\ &= \frac{1}{4}\times k\times 2^k\times\left(\frac{4}{3}-(2k+1)\left(\frac{1}{2}\right)^k-\frac{1}{3}\left(\frac{-1}{2}\right)^k\right) \\ &= \frac{k}{3}\times 2^k-\frac{k^2}{2}-\frac{k}{4}-\frac{k}{12}(-1)^k \\ &= \begin{cases} \frac{k}{3}\times 2^k-\frac{k^2}{2}-\frac{k}{3}, & k \text{ even} \\ \frac{k}{3}\times 2^k-\frac{k^2}{2}-\frac{k}{6}, & k \text{ odd} \end{cases} \end{aligned} \tag{41}$$

Finally

$$\begin{aligned} N &= \begin{cases} \frac{k}{3}\times 2^k-\frac{k^2}{2}-\frac{k}{3}+\frac{k^2}{2}+\frac{k}{2}=\frac{k}{3}\times 2^k+\frac{k}{6}, & k \text{ even} \\ \frac{k}{3}\times 2^k-\frac{k^2}{2}-\frac{k}{6}+\frac{k^2}{2}+\frac{1}{2}=\frac{k}{3}\times 2^k+\frac{3-k}{6}, & k \text{ odd} \end{cases} \end{aligned} \tag{42}$$

Considering all, the probability of change per pixel for the kth level of one-third probability steganography which is $N/(k\times 2^k)$ equals to

$$P(k) = \begin{cases} \frac{1}{3}+\frac{1}{6\times 2^k}, & k \text{ even} \\ \frac{1}{3}+\frac{3-k}{6\times k\times 2^k}, & k \text{ odd} \end{cases} \tag{43}$$

This probability asymptotically equals to 1/3 which was observed in Fig. 3 of Section 4 as well. The even function of $P(k)$ is a monotonic descending function in that it finds its minimum asymptotically. For the odd function of $P(k)$, by equating numerator of its derivative (Nom) to zero we

doi: 10.1049/iet-ipr.2013.0109

obtain

$$\begin{aligned}\text{Nom} &= -6\,k \times 2^k - \left(6 \times 2^k + 6\,k \ln(2) \times 2^k\right)(3-k) = 0 \\ &\Rightarrow -k - (1 + k \ln(2))(3-k) = 0 \\ &\Rightarrow \ln(2)k^2 - 3\ln(2)k - 3 = 0 \end{aligned} \tag{44}$$

The valid (positive) root of (44) is 4.06. Since for this case, we are interested only in odd integer numbers, the desired numbers that can minimise the odd part of $P(k)$ are 3 and 5. Therefore, the minimum will occur among these cases

$$\begin{aligned} P(3) = 0.33333, \quad P(5) = 0.33125, \\ P(\infty) = 0.33333 \end{aligned} \tag{45}$$

Equation (45) shows that the minimum probability of change per pixel is achieved working on groups of five pixels as embedding units. However, since the advantage of working on five pixels instead of three pixels is not significant, for the sake of simplicity, we implement the algorithm just on groups of three pixels, as explained in Section 3.