Computer Standards & Interfaces 33 (2011) 367–371

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

A module-based LSB substitution method with lossless secret data compression

Shang-Kuan Chen [*]

Department of Computer Science and Information Engineering, Yuanpei University, Hsinchu, Taiwan, ROC

ARTICLE INFO

Article history:
Received 26 May 2009
Received in revised form 16 July 2010
Accepted 27 November 2010
Available online 19 January 2011

Keywords:
Module-based LSB substitution
Data hiding
Lossless compression

ABSTRACT

All the various data hiding methods can be simply divided into two types: (1) the extracted important data are lossy, (2) the extracted important data are lossless. The proposed method belongs to the second type. In this paper, a module-based substitution method with lossless secret data compression function is used for concealing smoother area of secret image by modifying fewer pixels in the generated stego-image. Compared with the previous data hiding methods that extract lossless data, the generated stego-image by the proposed method is always with better quality, unless the hidden image is with very strong randomness.

1. Introduction

Data hiding [1–10] is a commonly used technique that embeds important data into the host images by modifying the pixels to protect the data from illegal peeking or damaging. In general, a stego-image is generated by hiding the important data into a host image. The data are transmitted much safely by using the stego-image than by transmitting the data themselves directly. We can categorize data hiding schemes into two types. In the first type, the data extracted from the generated stego-image are allowed with some distortions. In the second type, the extracted data cannot be allowed with any distortion. In [1,2], the extracted data were distorted, but these two methods are with extremely-high hiding capacity, for they could conceal data of very large size. For example, Chung et al. [1] hid the important image in a host image of the same size. Wang and Tsai [2] even hid a larger important image in a smaller host image. In both methods, the quality of the extracted image is acceptable, although not lossless. The LSB (Least-Significant Bit) substitution approach is a lossless approach. The simplest one is the so-called simple LSB substitution method which directly replaces the least-significant bit planes of the host image with hidden data. The related works [3–5] of LSB substitution method are stated in the next section.

In this paper, the repetition of the secret data is considered for reducing possible replacement of pixels in the stego-image. Hiding the secret data of heavy repetition will replace fewer pixels than that of light repetition by using the proposed method. Therefore, if the secret data is an image, in almost all the cases, the quality of the stego-images generated by the proposed method will be better than the current LSB-substitution hiding methods [3–5].

This paper is organized as follows. The related works are introduced in Section 2. The proposed method is stated in Section 3. Experimental results and analysis are shown in Sections 4 and 5, respectively. Finally, the conclusion is made in Section 6.

2. The related works

The related works [3–5] of simple LSB substitution method enable the stego-image to obtain better quality. Wang et al. [3] defined an optimal re-naming problem for the hidden data, and then used a genetic algorithm for searching the problem's nearly optimal solution. Chang et al. [4] gracefully developed a dynamic programming strategy to replace Wang et al.'s [3] genetic algorithm to get a faster algorithm to obtain the aforementioned theoretical solution. Rather than finding the optimal re-naming solution, Thien and Lin [5] proposed a digit-by-digit data hiding method based on modulus function. The quality (PSNR values) of their stego-image outperformed that of Ref. [3,4], without damaging the hiding capacity or lossless recovery of the secret data. The method is also faster than the two methods [3,4].

Thien and Lin's method [5] first partitions the secret data into a sequence of non-overlapping segments, and each segment m_i is in the range $\{0, 1, 2,\dots, q-1\}$ where q is a given positive integer called the base for module function. A segment m_i can be hidden in the ith pixel (whose original gray value is p_i) of the host image. In the simple LSB substitution method, the ith pixel value of the generated stego-image is $p_i' = p_i - (p_i \bmod q) + m_i$, true for each i. In Thien and Lin's method, p_i' is adjusted further to a new number p_i'' that is closer to p_i, and yet still congruent to p_i' on the module base q, i.e. $m_i = p''_i \bmod q = p_i' \bmod q$. Therefore, the decoding (to get m_i from p_i'') is simple and fast, and the impact to the host image is smaller.

To improve Thien and Lin's method, the proposed method uses repetition property of secret data for decreasing the number of modified pixels without extra bit for recording the repetition. That is a

* Tel.: +886 939113807.
E-mail address: skchen@mail.ypu.edu.tw.

0920-5489/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2010.11.002

different way with Ref. [3,4] that finds the optimal renaming for reducing the distortion. In detail, because there may be many repetitions in the secret data, the proposed method adopts an undefined number for flag, and records the repetition length, and repetition datum instead of the whole repetition data. The following experimental results show that the quality of the generated stego-image is better than that of the above three methods [3–5] unless the hidden image is with very strong randomness. The encoding and decoding algorithms are presented in the next section.

3. The proposed method

This section is divided into two subsections including encoding phase and decoding phase.

3.1. Encoding phase

The encoding phase is divided into two stages. In Stage 1, each datum is hidden in the pixel of host image to be a stego-pixel, while Stage 2 provides the pixel adjustment for obtaining nearest stego-pixel under the same datum extraction.

3.1.1. Stage 1: The hiding process

Let s_i denote the ith secret datum ranging from 0 to $m-1$, and each s_i be hidden in a pixel of the host image. The stego-image is generated in a pixel-by-pixel manner. Each time when hiding a not-yet-hidden datum s_i, we also take next coming data simultaneously if their values are all identical to s_i; i.e., if r contiguous data $s_i, s_{i+1},\dots,$ and s_{i+r-1} are all with the same value z for some positive integer r (Originally, these r data are supposed to be hidden in r pixels $p_i, p_{i+1},\dots$ and p_{i+r-1} of the host image, respectively, to generate r temporary stego-pixels). In the slight-repetition occurring, if r is 1, 2, or 3, then for each $l=i, i+1,\dots, i+r-1$, we assign the temporary stego-pixel t_l to $p_l-[p_l \bmod (m+1)]+s_l$. In a numerical system with base $b=m+1$, the available digits are $\{0, 1, 2,\dots, m\}$. Because each datum ranges from 0 to $m-1$, the only unused digit is m. This special digit m (the so-called flag-digit) is especially reserved to indicate the special case when the repetition length is over 3, i.e. $r>3$.

When a special heavy-repetition case occurs, besides using a flag digit (value m) for identifying the case, the repeated value z and the repetition length r ($r>3$) also need to be recorded. A three-digit vector $[m, z, r-4]$ is adopted for describing easily. The third component is $r-4$, rather than r, because r is at least 4 in this heavy-repetition case, and we can record $r-4$ rather than r in order to save coding-length. Notably, in the base $b=m+1$ system, the available digits are $\{0, 1, 2,\dots, m\}$, so the digit $(r-4)$ is forced to be in the range $0\le(r-4)\le m$. As a result, the actual value of the repetition length r must be in the range $4\le r\le m+4$. If there occurs an extremely-heavy-repetition case in which $r>m+4$, then the repetition interval should be cut into several intervals of smaller repetition length. Therefore, assume that $4\le r\le m+4$, then the temporary stego-pixels t_i, t_{i+1}, and t_{i+2} are $p_i-[p_i \bmod (m+1)]+m$, $p_{i+1}-[p_{i+1} \bmod (m+1)]+z$, and $p_{i+2}-[p_{i+2} \bmod (m+1)]+(r-4)$, respectively. (The remaining pixels $t_{i+3}, t_{i+4}, \dots$, and t_{i+r-1} equal to $p_{i+3}, p_{i+4},\dots,$ and p_{i+r-1}, respectively.) Notably, in this operation, the maximal ratio of the number of unchanged pixels to total number of pixels is $(m+1)/(m+4)$. Of course, with slight modification, these unchanged pixels can also be used to record some other data. This slightly-modified version can either increase *PSNR* or increase the amount-of-hidden-data (the so-called hiding-capacity).

Let the ordered n-tuples $<h_1, h_2,\dots, h_n>$ be the hidden data, the ordered n-tuples $\{img_1, img_2,\dots, img_n\}$ be the pixels of an n-pixels image, and the ordered n-tuples $[d_1, d_2,\dots, d_n]$ be the differences between the pixels of the host image and the stego-image. Assume that the data $<2, 4, 3, 3, 3, 3, 3, 5, 1, 1>$ are to be hidden in the ten pixels $\{70, 67, 66, 61, 68, 64, 63, 60, 61, 58\}$ of the host image. Since the data range from 0 to 5, m is assigned to 6 to reduce the distortion. Thus, the base system is $m+1=7$. The first two data are with no repetition; thus, the first two stego pixels are $63/7\ 7+2=65$ and $67/7\ 7+4=67$, where the operator / is integer division. From the third datum to the seventh datum, the values are the same and the number of these data is larger than 3; the lossless compression of these data can be used. First, the number 6 should be hidden in the third pixel of the host image; then the datum 3 should be hidden in the fourth pixel; finally, the repetition length 5 is hidden in the fifth pixel by hiding $(5-4=1)$. Therefore, using the proposed method to encode, the generated temporary stego-pixels will be $\{65, 67, 69, 59, 64, 64, 63, 61, 57, 57\}$. The differences between the host image and the temporary stego-image are thus $[5, 0, 3, 2, 4, 0, 0, 1, 4, 1]$. Notably, both the sixth and the seventh pixels of the host image and the temporary stego-image are the same, because the third, the fourth, and the fifth pixels of the temporary stego-image have already recorded the information of $<3, 3, 3, 3, 3>$ of the hidden data, by hiding a three-tuple $(6, z=3, r-4=1)$ in the third, fourth, and fifth pixels of the host image. Here, 6 is the heavy-repetition-flag, $z=3$ is the heavy-repeated-value, and $r+4=1+4=5$ is the repetition length that z is repeated.

3.1.2. Stage 2: The adjusting process

After the proposed hiding process, the difference between the pixel value t_i of the temporary stego-image and the corresponding p_i of the host image ranges from $-m$ to m. To reduce the distortion, this stage adjusts the stego-pixel value if there is another value that has the same extraction datum also closest to the pixel of the host image. The proposed method divides the possible situations into the following five cases:

Case 1: $(\lfloor(m+1)/2\rfloor<t_i-p_i\le m$ and $t_i\ge m+1)$.

$g_i\leftarrow t_i-m-1.$

Case 2: $(-m\le t_i-p_i<-\lfloor(m+1)/2\rfloor$ and $t_i\le 254-m)$.

$g_i\leftarrow t_i+m+1$

Case 3: $(-\lfloor(m+1)/2\rfloor\le t_i-p_i\le\lfloor(m+1)/2\rfloor)$.

$g_i\leftarrow t_i$

Case 4: $(\lfloor(m+1)/2\rfloor<t_i-p_i\le m$ and $t_i\le m)$. (Boundary)

$g_i\leftarrow t_i$

Case 5: $(-m\le t_i-p_i<-\lfloor(m+1)/2\rfloor$ and $t_i\ge 255-m)$. (Boundary)

$g_i\leftarrow t_i$

In the following, it is shown that the $|g_i-t_i|\le\lfloor m/2\rfloor$ in Cases 1 and 2.

Proposition 1. In Cases 1 and 2 of the proposed adjusting process, the difference of g_i and p_i is $|g_i-p_i|\le\lfloor m/2\rfloor$.

Proof. Recall that in Case 1, $g_i=t_i-m-1$, and in Case 2, $g_i=t_i+m+1$.

In Case 1:

(1) $g_i-p_i=t_i-m-1-p_i=(t_i-p_i)-m-1\le m-m-1=-1$.
(2) $g_i-p_i=t_i-m-1-p_i=(t_i-p_i)-m-1>\lfloor(m+1)/2\rfloor-m-1$.
When m is even, $g_i-p_i>\lfloor(m+1)/2\rfloor-m-1=m/2-m-1=-m/2-1$. Because g_i-p_i is integer, $g_i-p_i\ge -m/2=-\lfloor m/2\rfloor$.
When m is odd, $g_i-p_i>\lfloor(m+1)/2\rfloor-m-1=(m+1)/2-m-1=-(m+1)/2$. Because g_i-p_i is integer, $g_i-p_i\ge-(m-1)/2\ge-\lfloor m/2\rfloor$.
Combined (1) and (2), $|g_i-p_i|\le\lfloor m/2\rfloor$.

In Case 2:

(1) $g_i-p_i=t_i+m+1-p_i=(t_i-p_i)+m+1\ge -m+m+1=1$.