JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 26, 1199-1212 (2010)

Transforming LSB Substitution for Image-based Steganography in Matching Algorithms*

CHENG-HSING YANG AND SHIUH-JENG WANG+
Department of Computer Science
National Pingtung University of Education
Pingtung, 900 Taiwan
+*Department of Information Management*
Central Police University
Taoyuan, 333 Taiwan

Simple least-significant-bit (LSB) substitution is a method used to embed secret data in least significant bits of pixels in a host image. The LSB approaches typically achieve high capacity. A simple LSB substitution, which hides secret data directly into LSBs, is easily implemented but will result in bad quality of the stego-image. In order to reduce the degradation of the host image after embedding, an LSB substitution scheme was proposed by Wang *et al.* They proposed a genetic algorithm to search for approximate solutions. Also, Chang *et al.* proposed a dynamic programming strategy to efficiently obtain a solution. In this paper, we propose a more general model of Wang *et al*.'s LSB substitution scheme, called as transforming LSB substitution. In order to overcome the problem associated with the two previous approaches of a long running time, a more efficient approach, referred to as the matching approach, is proposed to find a better solution. Some experiments, demonstrations and analyses are shown in this paper to demonstrate our new scheme and approach.

Keywords: information hiding, least-significant-bit (LSB), matching algorithm, image intellectual rights, security

1. INTRODUCTION

Information hiding is a method of hiding secret data into a host medium so that the hidden data are imperceptible but known to the intended recipient [1]. The host medium may be a digital image, audio, video, or another type of media. Among the different kinds of media, the digital image is most popularly used as the host media to convey secret information. In the image hiding system, the image used to embed secret data is called the *host image* (*cover image*). The resultant image, which is embedded with secret data, is called the *stego-image*.

Different approaches to data hiding have been proposed for different goals, such as, invisibility, robustness and capacity [2-6]. One of the common approaches is based on manipulating the least-significant-bit (LSB) planes, which replaces the least significant bits of the host image with secret data [7-13]. LSB approaches typically achieve high capacity.

In a data hiding procedure, the host image must not be degraded too much, otherwise the quality of the stego-image will not be acceptable, and the embedded data easily

Received July 29, 2008; revised November 3, 2008; accepted January 22, 2009.
Communicated by H. Y. Mark Liao.
* This paper was supported in part by the National Science Council of Taiwan, R.O.C. under Grants No. NSC 95-2221-E-015-002-MY2, NSC 97-2221-E-153-001, and NSC 97-2221-E-015-001.
+ Corresponding author.

detected. A simple LSB substitution, which hides secret data into LSBs directly, is easy implemented but will result in a low quality stego-image. In order to achieve a good quality stego-image, Wang *et al.* used a substitution matrix to transform the secret data values prior to embedding into the host image [8]. For a *k*-bit LSB substitution, the exhaustive search method would take a long period of time to find an optimal substitution matrix. In order to overcome a long running time of the exhaustive search, Wang *et al*. [8] proposed a genetic algorithm to search for an approximate solution, and Chang *et al*. [11] also proposed a dynamic programming strategy to find an optimal substitution matrix in an efficient manner.

In this paper, we provide a better LSB substitution scheme, referred to as the transforming LSB substitution scheme. It should be noted that the scheme proposed by Wang *et al.* [8] is a special case of ours. Also, some properties of our new scheme are shown in this paper. Since both the genetic algorithm [8] and the dynamic programming strategy [11] would need a lot of time to find an optimal solution in our new scheme, we propose a new approach, called the matching approach [14], to find a better solution of more efficient manner.

The remainder of this paper is organized as follows. Section 2 provides some brief descriptions of some prior related works. Our transforming LSB substitution scheme is given in section 3, and our matching approach to solve the new scheme is discussed here as well. Some properties of the new scheme are shown in section 4, and the results of the experiment and analyses are given in section 5. Finally, we draw our conclusions in section 6.

2. RELATED WORKS

In this section, we describe some previous works about LSB substitutions. First we show the LSB substitution scheme proposed by Wang *et al.* Then, we introduce two prior approaches, the genetic algorithm and the dynamic programming strategy, used to find a solution in the scheme.

2.1 Wang *et al.*'s LSB Substitution Scheme

As shown in Fig. 1, Wang *et al.* [8] proposed a model to describe their LSB substitution scheme. Suppose that the embedded data is secret image *C*, while the host multimedia is host image *H*. Both *C* and *H* are 8-bit gray images. Using simple LSB substitution, the rightmost *k* least significant bits of *H* will be replaced by *C*. *k* is denoted as the length of LSB, *i.e.*, |LSB| = *k*. *C*′ is defined by decomposing the bit streams of *C* into several *k*-bit units and treating each unit as a single pixel. Also, let *R* be the *k*-bit residual image, which is derived by extracting the rightmost *k* least significant bits from each pixel in the host image *H*. To increase security, the pixel location of *C*′ is randomized by a bijection (*i.e.*, one-to-one and onto) mapping function into a meaningless image *C*″. In order to achieve a good embedding result, an $N \times N$ substitution matrix $S = \{s_{ij}\}$ is defined by

$$s_{ij} = \begin{cases} 1 & \text{gray value } i \text{ is replaced by gray value } j, \\ 0 & \text{do nothing,} \end{cases}$$

where $N = 2^k$. The substitution matrix S is used to replace each pixel with gray value i in C'' by a pixel with gray value j. There are $(2^k)!$ different substitution matrices, denoted as S_1, S_2, …, $S_{(2^k)!}$. A matrix S_i transforms C'' into C_i''', and a stego-image Z can be obtained by replacing R with C_i'''. In order to estimate the quality of the stego-image, the mean square error (*MSE*) is proposed and is defined as follows:

$$MSE = \frac{1}{m}\sum_{l=0}^{m-1}(z_l - h_l)^2 = \frac{1}{m}\sum_{l=0}^{m-1}(c_l''' - r_l)^2.$$

Here z_l, h_l, c_l''', and r_l represent the pixel gray values of the stego-image Z, host image H, secret image C''', and residual image R in location l, respectively, and m is the image size. An optimal secret image C^* is the secret image C_i''', which leads to an optimal embedding result (*i.e.*, the result with the minimum *MSE*) in their scheme. Note that the $k \times 2^k$ bits of additional data, which records the transformation method, need to be kept or transformed for extracting. Wang *et al.* [8] proposed a genetic algorithm to find an approximate solution in their model.

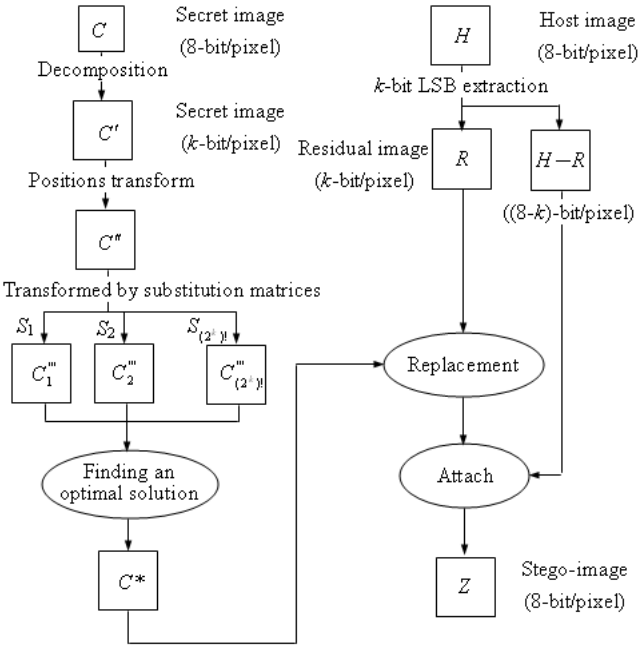

Fig. 1. The model of Wang *et al.*'s LSB substitution scheme.

2.2 Dynamic Programming Strategy

Chang *et al.* [11] proposed a dynamic programming strategy to find an optimal solution of Wang *et al.*'s model [8] in an efficient manner. Their main approach was to redefine the substitution matrix S as the matrix $M_{N\times N}$. If the pixels with gray value i in C''

were transformed into gray value j, the entry $M[i][j]$ of $M_{N\times N}$ represented the total square differences between the transformed pixels and the pixels of the corresponding locations in the residual image R. It is obvious that a list $g_0g_1\ldots g_{N-1}$ with a minimum of $M[0][g_0] + M[1][g_1] + \ldots + M[N-1][g_{N-1}]$ is an optimal solution of Wang *et al.*'s model. A dynamic approach was proposed to find an optimal list as follows: Suppose that *Set* is a subset of $\{0, 1, \ldots, N-1\}$. Let $Cost[r, Set]$ denote the minimum cost of the substitution list picked up from the sub-matrix M' of $M_{N\times N}$, where M' is constructed by rows from $N-1$ to r and columns listed in *Set*. The dynamic formula is expressed as follows:

$$Cost[r, Set] = \min_{j\in Set}\{M[r][j] + Cost[r+1,\ Set - \{j\}]\}.$$

The initial value of $Cost[N, \{\}]$ is zero. Then, $Cost[0, \{0, 1, \ldots, N-1\}]$ is the minimum cost of all $M[0][g_0] + M[1][g_1] + \ldots + M[N-1][g_{N-1}]$.

3. THE TRANSFORMING LSB SUBSTITUTION SCHEME

In this section, we propose a more general definition of Wang *et al.*'s LSB substitution scheme. In addition we demonstrate a matching approach to solve the new scheme.

3.1 The Transforming LSB Substitution Scheme

We redefine the LSB substitution scheme proposed by Wang *et al.* [8] in a more general situation. Let the secret image C form a bit string $S = s_0s_1\ldots s_{n-1}$, where n is the number of secret bits and s_i represents a bit, for $i = 0, 1, 2, \ldots, n-1$. For the k-bit simple LSB substitution approach, the bits in host image H, which will be replaced by the bit string S, form a bit string $R = r_0r_1\ldots r_{n-1}$. We will refer to the bit string S as the secret string and to the bit string R as the replaced string. As shown in Fig. 2, each bit s_i in S will replace the bit r_i in R. The value of k is usually equal to 1, 2, 3, or, 4 and each of the k adjacent bits of R in Fig. 2 come from the same pixel of the host image. The idea of Wang *et al.*'s LSB substitution scheme is that each k-bit string of S is transformed into another k-bit string before replacing R. Then we extend the k-bit string to the l-bit string, denoted as the matching string, where $l \geq k$. l is the length of a matching string. As shown in Fig. 2, each l-bit string of S will be transformed into another l-bit string before replacing R, therefore we call our new scheme as the l-bit transforming LSB substitution scheme. Note that the LSB substitution scheme proposed by Wang *et al.* is a special case of our new scheme with $l = k$.

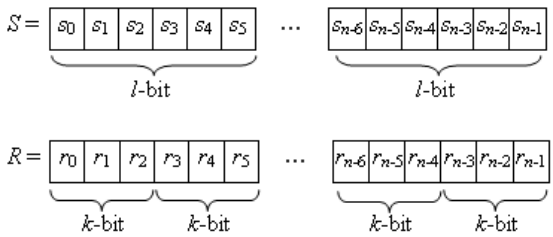

Fig. 2. An example of a bit string S and a bit string R, where $l = 6$ and $k = 3$.

3.2 Matching Approach of Weighted Bipartite Graph

In order to find an optimal solution in the transforming LSB substitution scheme, we propose a matching approach to solve this problem. A *bipartite graph* consists of two sets of vertices in which there are weighted edges to connect the two disjoint sets. The *matching* of a bipartite graph is a set of edges with no endpoints in common. A *maximal matching* is a matching to which no edge in the graph can be added. A *maximum matching* is a matching with maximum weight. Note that the maximum matching problem can be solved by a well-known approach, called as *Hungarian Method*, and has been applied to different regions [14-16].

Given an l-bit transforming LSB substitution scheme, we can construct a bipartite graph $G = (A \cup B, E)$, where vertex set $A = \{a_0, a_1, \ldots, a_{N-1}\}$, vertex set $B = \{b_0, b_1, \ldots, b_{N-1}\}$, and edge set $E = \{(a_i, b_j) \mid \text{for each } 0 \le i, j \le N-1\}$. N is equal to 2^l. Suppose S and R are the secret string and the replaced string, respectively. Then, a maximal matching of G has cardinality N and represents a transformation which transforms the secret string S into another secret string S'. We denote such maximal matching as (S, S'). Because we are only interested in the matching with cardinality N, any matching appearing in the following article is a maximal matching. When weights are assigned to all edges such that the weight of a matching (S, S') is equal to $MSE_{S'R}$, where $MSE_{S'R}$ denotes the mean square error between host image H and stego-image Z with R substituted by S', a matching with minimum weight is an optimal solution. For any value of l, not all such weighted bipartite graphs exist. But, if k divides l, the weighted bipartite graph can be created easily. Fig. 3 shows the case of $l = 3 \times k$, where $S_0, S_1, \ldots, S_{m-1}$ are all k-bit string of S and $R_0, R_1, \ldots, R_{m-1}$ are all k-bit string of R, and where m is the size of the host image. We scan strings S and R simultaneously, from left to right and process each l-bit string of S and R as follows. Consider each l-bit string $S_{3i+0}||S_{3i+1}||S_{3i+2}$ of S and l-bit string $R_{3i+0}||R_{3i+1}||R_{3i+2}$ of R, where $i = 0, 1, \ldots, (m/3) - 1$ and the symbol "||" means the string concatenation operator. Let x be the value of $S_{3i+0}||S_{3i+1}||S_{3i+2}$, denoted as $val(S_{3i+0}||S_{3i+1}||S_{3i+2})$. Also, let $y_0 = val(R_{3i+0})$, $y_1 = val(R_{3i+1})$, and $y_2 = val(R_{3i+2})$. Suppose j is an integer in 0, 1, …, $N - 1$. j can be represented as an l-bit string J, and let j_0, j_1, and j_2 be the value of left k-bit string, middle k-bit string, and right k-bit string of J, respectively. If the l-bit string $S_{3i+0}||S_{3i+1}||S_{3i+2}$ matches to the l-bit string J, the l-bit string J is used to substitute the l-bit string $R_{3i+0}||R_{3i+1}||R_{3i+2}$. Therefore, the square error in this case is $(j_0 - y_0)^2 + (j_1 - y_1)^2 + (j_2 - y_2)^2$. Assume that the weight of edge (a_x, b_j) is denoted as $cost\ (a_x, b_j)$. The $cost(a_x, b_j)$ in the current stage is calculated by adding $(j_0 - y_0)^2 + (j_1 - y_1)^2 + (j_2 - y_2)^2$ to the $cost(a_x, b_j)$ in the previous stage. Therefore we have the following equation,

$$\text{current } cost(a_x, b_j) = \{\text{previous } cost(a_x, b_j)\} + (j_0 - y_0)^2 + (j_1 - y_1)^2 + (j_2 - y_2)^2, \quad \text{for } j = 0, 1, \ldots, N-1. \tag{1}$$

Thus, as shown in Fig. 3, we add the value $(j_0 - y_0)^2 + (j_1 - y_1)^2 + (j_2 - y_2)^2$ to edge (a_x, b_j), for $j = 0, 1, \ldots, N - 1$. After all l-bit strings of S and R are processed, the weights of all edges in the bipartite graph have been assigned.

Fig. 4 shows an example to explain why k divides l required for creating a correct weighted bipartite graph. In Fig. 4, values k and l are 4 and 6, respectively. String S matches to string S', then string S' is used to substitute string R. The first 6-bit of S' sub-

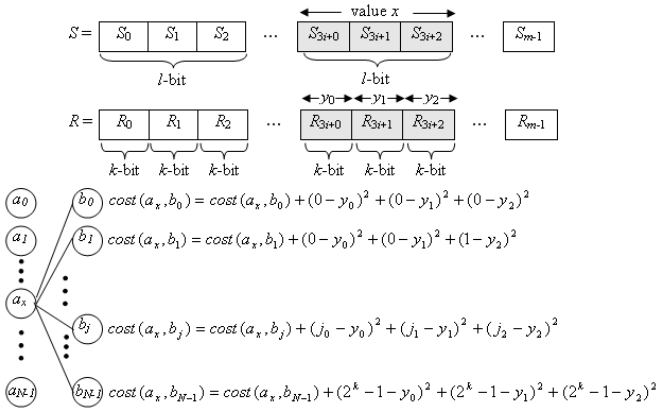

Fig. 3. An example of the weight assignment of a bipartite graph for bit strings S and R, where $l = 3 \times k$. x, y_0, y_1, and y_2 are the values of the corresponding bit strings.

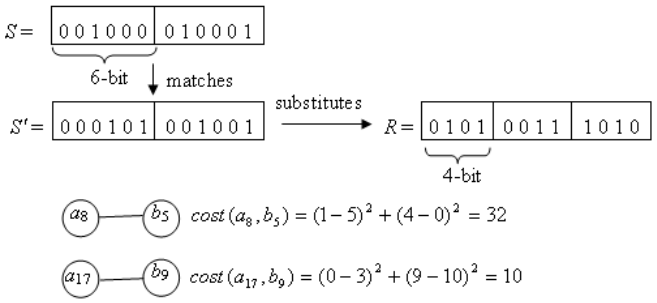

Fig. 4. An example of the weight assignment of a bipartite graph for bit strings S and R, where $k = 4$ and $l = 6$.

stitutes the first 6-bit of R. Note that the first 6-bit of R consists of 4 LSB bits of one pixel and 4th LSB bit and 3rd LSB bit of another pixel. Therefore, $cost(a_8, b_5) = (1 - 5)^2 + (4 - 0)^2$. The total cost of the matching is $cost(a_8, b_5) + cost(a_{17}, b_9) = 32 + 10 = 42$. However, the exact square error of this substitution is $(1 - 5)^2 + (4 - 3)^2 + (9 - 10)^2 = 18$.

3.3 Embedding Process and Extracting Process

Note that the optimal matching found by the matching approach must be kept in order to extract the embedded message. It needs $l \times 2^l$ bits, denoted as string O, to represent the optimal matching. We describe the embedding process and extracting process as follows.

Algorithm Embedding-Process (C, H, k, l)
Input: a secret image C and a host image H for l-bit transforming LSB substitution with k-bit simple LSB substitution;
Output: stego-image Z and optimal matching O;
1. Get secret string S from C and replaced string R from H

2. Create bipartite graph G and find an optimal matching O
3. Transform S into S' by the optimal matching O
4. Create stego-image Z by replacing R with S'

Algorithm Extracting-Process (Z, O, k, l)
Input: a stego-image Z and an optimal matching O for l-bit transforming LSB substitution with k-bit simple LSB substitution;
Output: secret image C;
1. Get embedded string S' from Z
2. Transform S' into S by the optimal matching O
3. Construct secret image C from S

4. PROPERTIES OF THE TRANSFORMING LSB SUBSTITUTION SCHEME

In this section, we show some properties of the l-bit transforming LSB substitution scheme.

Definition 1 Given a *secret string* S and a number l, the *matching space* $M(S, l)$ is defined as the set including all different bit strings S', where matching (S, S') is a matching of bipartite graph G_l. That is, $M(S, l) = \{S' \mid (S, S') \text{ is a matching of } G_l\}$.

Definition 2 Given a *secret string* S of a secret image C and a *replaced string* R of a host image H, then both MSE_{SR} and MSE_{HZ} are defined as the mean square error between host image H and stego-image Z, where Z is created by replacing R by S.

Definition 3 Given a number l, $BMSE_l$ is defined as the best mean square error between host image H and stego-image Z, where Z is a solution of the l-bit transforming LSB substitution problem.

Theorem 1 Given two l-bit transforming LSB substitution problems with $l = l_1$ and l_2, respectively. If $l_1 \mid l_2$ and $l_1 < l_2$, then for all secret strings S, *matching space* $M(S, l_1) \subseteq$ *matching space* $M(S, l_2)$.

Proof: Fig. 5 shows the case of $l_2 = 3 \times l_1$. This can be used to explain any case of $l_1 | l_2$ and $l_1 < l_2$. In Fig. 5, left side shows an l_1-bit matching (S, S'), which transforms l_1-bit string A, B, and C of secret string S into A', B', and C' of another secret string S', respectively. That is, $S' \in$ *matching space* $M(S, l_1)$. Then, an l_2-bit matching (S, S') can be obtained by transforming l_2-bit string $A||B||C$ of secret string S into $A'||B'||C'$ of secret string S'. Therefore, $S' \in$ *matching space* $M(S, l_2)$. We have shown that *matching space* $M(S, l_1) \subseteq$ *matching space* $M(S, l_2)$ for any secret string S. ❑

From Theorem 1, we can easily obtain the following theorem.

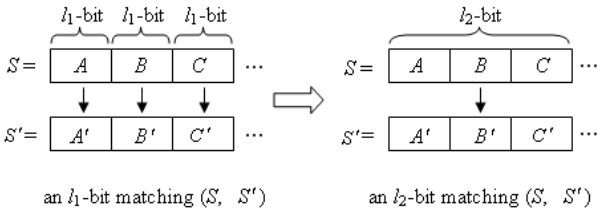

Fig. 5. An l_1-bit matching (S, S') can be used to form an l_2-bit matching (S, S').

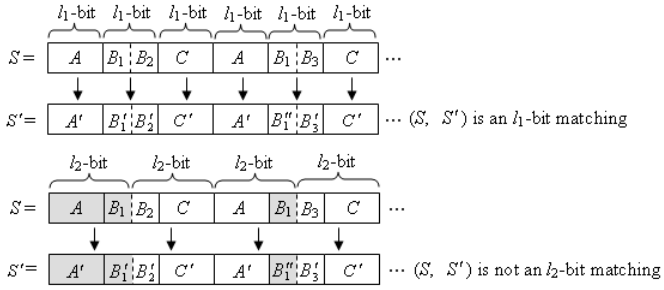

Fig. 6. (S, S') is an l_1-bit matching, but not an l_2-bit matching.

Theorem 2 Given two l-bit transforming LSB substitution problems with $l = l_1$ and l_2, respectively. If $l_1 \mid l_2$ and $l_1 < l_2$, then, for all secret string S, $BMSE_{l_1} \geq BMSE_{l_2}$.

Theorem 3 Given two l-bit transforming LSB substitution problems with $l = l_1$ and l_2, respectively. If l_1 does not divide l_2 and $l_1 < l_2$, then there exists a secret string S such that *matching space* $M(S, l_1) \not\subset$ *matching space* $M(S, l_2)$ and *matching space* $M(S, l_2) \not\subset$ *matching space* $M(S, l_1)$.

Proof: Figs. 6 and 7 show the case $\text{lcm}(l_1, l_2) = 3 \times l_1 = 2 \times l_2$, where $\text{lcm}(l_1, l_2)$ is the least common multiple of $l_1$and l_2. It can be used to explain any case where l_1 does not divide l_2 and $l_1 < l_2$. In Fig. 6, (S, S') is an l_1-bit matching, which transforms l_1-bit string A, $B_1 \,\|\, B_2$, $B_1 \,\|\, B_3$, and C into l_1-bit string A', $B_1' \,\|\, B_2'$, $B_1'' \,\|\, B_3'$, and C', respectively. But, (S, S') is not an l_2-bit matching, because l_2-bit string $A\|B_1$ will be transformed into both $A' \,\|\, B_1'$ and $A' \,\|\, B_1''$, which are pointed out by the gray background in Fig. 6. Therefore, there exists a secret string S such that $S' \in M(S, l_1)$ and $S' \notin M(S, l_2)$. Thus, $M(S, l_1) \not\subset M(S, l_2)$.

In Fig. 7, (S, S') is an l_2-bit matching, which transforms l_2-bit string $A\|B_1$, $B_2\|C$, and $A\|B_3$ into $A'\| B_1'$, $B_2' \| C'$, and $A''\| B_3''$, respectively. But, (S, S') is not an l_1-bit matching, because l_1-bit string A will be transformed into both A' and A'', which are pointed out by the gray background in Fig. 7. Therefore, there exists a secret string S such that $S' \in M(S, l_2)$ and $S' \notin M(S, l_1)$. Thus, $M(S, l_2) \not\subset M(S, l_1)$. ❑

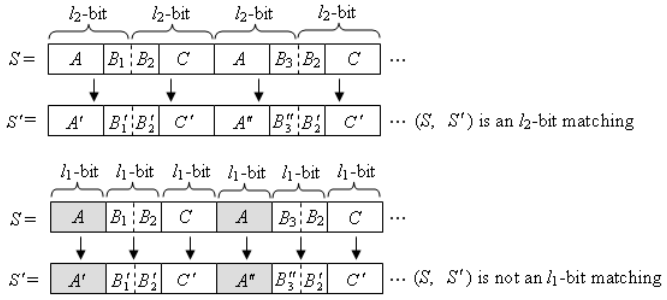

Fig. 7. (S, S') is an l_2-bit matching, but not an l_1-bit matching.

5. EXPERIMENTAL RESULTS AND ANALYSES

5.1 Experimental Results

In this subsection, we show the experimental results of our approach. We have written our approach in C language, and the running environment is a 1.3 GHz Pentium 4 (M) CPU and 512Mb RAM, with a Windows XP operating system. The images tested in our experiment are all 8-bit images with 256 gray levels. Two test images with size 512 × 512 shown in Figs. 8 (a) and (b) are treated as the host images. The other test images that are prepared to be embedded into the host images are classified into two classes: the first class contains four images with size 512 × 256 as shown in Figs. 8 (c)-(f), and the second class contains four images with size 256 × 256 as shown in Figs. 8 (g)-(j).

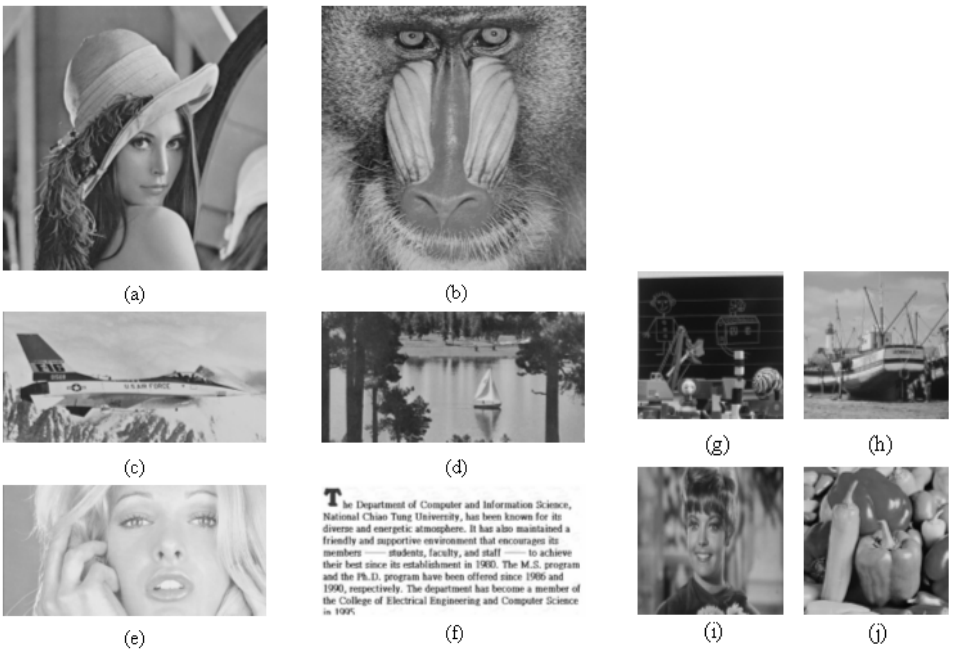

Fig. 8. The two host images with size 512 × 512 are (a) Lena and (b) Baboon. The first class of four secret images with size 512 × 256 are (c) Airplane, (d) Sailboat, (e) Tiff and (f) Text. The second class of four secret images with size 256 × 256 are (g) Toys, (h) Boat, (i) Girl and (j) Peppers.

Table 1. The results of embedding 512 × 256 secret images into host images Lena and Baboon by different approaches: simple LSB substitution, matching approach with $l = 4$ and 8 and $k = 4$.

		Simple LSB $k = 4$		Matching $l = 4$		Matching $l = 8$	
		Lena	Baboon	Lena	Baboon	Lena	Baboon
Airplane	MSE	38.73	38.82	33.29	32.84	26.64	26.28
	ratio	100%	100%	86%	85%	69%	68%
	time	0.35	0.21	0.41	0.32	1.73	1.68
Sailboat	MSE	37.87	37.05	34.18	33.66	28.56	28.15
	ratio	100%	100%	90%	91%	75%	76%
	time	0.26	0.21	0.25	0.25	1.62	1.66
Tiff	MSE	42.56	42.93	33.26	32.90	25.73	25.33
	ratio	100%	100%	78%	77%	60%	59%
	time	0.21	0.21	0.25	0.25	1.44	1.44
Text	MSE	57.71	58.77	29.16	28.69	26.51	26.11
	ratio	100%	100%	51%	49%	46%	44%
	time	0.25	0.23	0.23	0.25	1.81	1.81
Average	MSE	44.22	44.39	32.48	32.02	26.86	26.47
	ratio	100%	100%	76%	75%	63%	62%
	time*	0.27	0.22	0.35	0.33	1.75	1.82

* Each running time is all running time from first instruction to last instruction, including reading cover image, executing matching, calculating MSE, and writing stego-image. The average of the pure running times which are just for executing matching, is 0.037 seconds for $l = 4$ and is 0.490 seconds for $l = 8$.

Table 1 shows the results of those 512 × 256 images being embedded into host images, Lena and Baboon, by the following approaches: simple LSB substitution, matching approach with $l = 4$ and 8, which are all divisible by $k = 4$. We use the MSE created by simple LSB substitution as the basis. The ratios of the MSE of the matching approach to the simple LSB substitution are shown in the Table 1. The average ratios of the matching approach with $l = 8$ are 63% and 62% corresponding to the host images Lena and Baboon, respectively. It shows that the matching approach improves the quality of embedded images substantially. Also, the matching approach runs very fast. All of them finished within 2 seconds.

Table 2 shows the results of those 256 × 256 images being embedded into host images Lena and Baboon. Because these cases are 2-bit LSB substitution, we use $l = 2$, 4, 6, and 8, which are all divisible by $k = 2$, to show the results of the matching approach. The average MSE ratios of the matching approach with $l = 8$ are 83% for both images Lena and Baboon. The cells in Table 2, marked by gray background, are the cases that the results of $l = 6$ are worse than that of $l = 4$. It is for this reason that 4 does not divide 6 as shown in Theorem 3.

5.2 Analyses

In order to demonstrate that our approach is efficient in the running time, we compare the time complexity of our approach to that of Chang *et al.*'s [11]. The time analysis of Wang *et al.*'s genetic algorithm [8] is not given in our discussions, for the reason that their approach can just find an approximate solution and will spend more time than

Chang *et al.* as shown in [11]. Applying the *Hungarian Method* shown in [14] to find a minimum matching, the time complexity of our matching approach is $O(N^3 + Nm)$, where Nm is for creating the weighted bipartite graph G and N^3 is for finding an optimal solution. On the other hand, the dynamic programming approach takes $O(Nm)$ to construct a modified substitution matrix. Also, in order to calculate the value of $m\_Cost[0, \{0, 1, \ldots, N-1\}]$, all subsets of $Set = \{0, 1, \ldots, N-1\}$ must be created. Given a size i, there exists $\binom{N}{i}$ different subsets of *Set*. Therefore, there are $\binom{N}{0}+\binom{N}{1}+\cdots+\binom{N}{N}=2^N$ different subsets needed to be created. Hence, the time complexity of Chang *et al.*'s approach is $O(2^N + Nm)$.

Table 2. The results of embedding 256 × 256 secret images into host images Lena and Baboon by different approaches: simple LSB substitution, matching approach with $l = 2$, 4, 6 and 8 and $k = 2$.

		Simple LSB $k=2$		Matching $l=2$		Matching $l=4$		Matching $l=6$		Matching $l=8$	
		Lena	Baboon	Lena	Baboon	Lena	Baboon	Lena	Baboon	Lena	Baboon
Toys	MSE	2.45	2.45	2.18	2.19	2.09	2.09	2.09	2.10	1.79	1.79
	ratio	100%	100%	89%	89%	85%	85%	85%	85%	73%	73%
	time	0.23	0.11	0.22	0.14	0.19	0.18	0.37	0.36	1.56	1.55
Boat	MSE	2.30	2.29	2.28	2.29	2.16	2.16	2.15	2.15	1.91	1.91
	ratio	100%	100%	99%	100%	94%	95%	93%	94%	83%	84%
	time	0.11	0.11	0.14	0.14	0.17	0.19	0.37	0.42	1.46	1.45
Girl	MSE	2.32	2.31	2.31	2.31	2.21	2.21	2.24	2.24	2.00	2.00
	ratio	100%	100%	100%	100%	95%	96%	97%	97%	86%	87%
	time	0.11	0.11	0.14	0.14	0.17	0.18	0.37	0.36	1.35	1.36
Peppers	MSE	2.33	2.34	2.33	2.33	2.26	2.26	2.28	2.29	2.06	2.06
	ratio	100%	100%	100%	100%	97%	97%	98%	98%	88%	88%
	time	0.11	0.11	0.14	0.14	0.2	0.18	0.37	0.36	1.48	1.50
Average	MSE	2.35	2.35	2.28	2.28	2.18	2.18	2.19	2.20	1.94	1.94
	ratio	100%	100%	97%	97%	93%	93%	93%	94%	83%	83%
	time	0.14	0.11	0.16	0.14	0.18	0.18	0.40	0.38	1.44	1.47

Table 3. Compare the running time $O(N^3 + Nm)$ of our approach with the running time $O(2^N + Nm)$ of Chang *et al.*, for different values of l.

Time Complexities	$l=2$	$l=4$	$l=6$	$l=8$
$N^3 + Nm = (2^l)^3 + 2^l \times 2^{18}$	$2^6 + 2^{20}$	$2^{12} + 2^{22}$	$2^{18} + 2^{24}$	$2^{24} + 2^{26}$
$2^N + Nm = 2^{2^l} + 2^l \times 2^{18}$	$2^4 + 2^{20}$	$2^{16} + 2^{22}$	$2^{64} + 2^{24}$	$2^{256} + 2^{26}$

Table 3 compares the time complexity $O(N^3 + Nm)$ of our approach with the one of $O(2^N + Nm)$ in the scheme of Chang *et al.* It lists the values between $N^3 + Nm$ and $2^N + Nm$ for different values of l. When l is larger or equal to 6, the dynamic programming strategy will slow down considerably. It seems that it is hard to finish in a feasible time

Table 4. Compare the running time (unit: second) and MSE values of our matching approach and Chang *et al.*'s dynamic programming approach with $l = k = 2$ and $l = k = 4$.

	Dynamic $k = 2$		Matching $l = 2$			Dynamic $k = 4$		Matching $l = 4$	
	Lena	Baboon	Lena	Baboon		Lena	Baboon	Lena	Baboon
Toys	0.57	0.64	0.22	0.14	Airplane	0.70	0.76	0.41	0.32
Boat	0.60	0.69	0.14	0.14	Sailboat	0.68	0.73	0.31	0.34
Girl	0.60	0.58	0.14	0.14	Tiff	0.84	0.63	0.33	0.32
Peppers	0.59	0.59	0.14	0.14	Text	0.85	0.78	0.34	0.32
Average Time*	0.59	0.63	0.16	0.14	Average Time*	0.77	0.73	0.35	0.33
Average MSE	2.28	2.28	2.28	2.28	Average MSE	32.48	32.02	32.48	32.02

* For $l = k = 2$ and $l = k = 4$, the running times of ours are much better than that of Chang *et al.*'s. The running times of simulation results don't seem to fit the analyses of time complexity shown in Table 3. (Hint: Because the main cost of running time is Nm, which is caused by creating a weighted bipartite graph or a substitution matrix, for $l = k = 2$ and $l = k = 4$. Therefore, the difference for finding a solution at the running time between the matching-approach and the dynamic-programming-approach is minor only. A possible empiricism is that our program is written by programming-C, and the source-code programmed by the compared scheme is written by programming-C^{++}.)

when $l = 6$ and $l = 8$. Therefore our approach is more efficient than Chang *et al.*'s. Table 4 shows the running times of simulation results of our matching approach and Chang *et al.*'s dynamic programming approach with $l = k = 2$ and $l = k = 4$. Their program was implemented by Visual C++ language. The results of MSE values are the same in both approaches, and the averages of MSE values are also listed in the table. As shown in Table 4, to obtain similar MSE, our approach spends less time in both cases with $l = k = 2$ and $l = k = 4$. Also, if we let $l = 8$, then we could use a processing time, which was about 2.3 times longer than Chang *et al.*'s, to reduce the MSE obtained by Chang *et al.*'s dynamic programming [11] from 32.25 to 26.66 if $k = 4$ in [11] (or from 2.28 to 1.94 if $k = 2$ in [11]); in other words, to get an increase of about 0.8 dB (0.7 dB) in PSNR.

5. CONCLUSIONS

In this paper, we have proposed a more general definition for Wang *et al.*'s LSB substitution scheme, referred to as the transforming LSB substitution scheme. When compared to past similar studies such as [8, 11], a dramatic saving in time complexity is achieved using our scheme to find a better solution. In other words, the performance in terms of running time is substantially better than that of previous ones as is evident by the significant improvement in our experiments.

REFERENCES

1. F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding – A survey," *Proceedings of IEEE*, Vol. 87, 1999, pp. 1062-1078.
2. D. C. Wu and W. H. Tsai, "Spatial-domain image hiding using image differencing,"

IEE Proceedings of Vision, *Image and Signal Processing*, Vol. 147, 2000, pp. 29-37.
3. S. Katzenbeisser and F. A. P. Petitcolas, *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House, London, 2000.
4. S. D. Lin and C. F. Chen, "A robust DCT-based watermarking for copyright protection," *IEEE Transactions on Consumer Electronics*, Vol. 46, 2000, pp. 415-421.
5. C. C. Chang, T. S. Chen, and L. Z. Chung, "A steganographic method based upon JPEG and quantization table modification," *Information Sciences*, Vol. 141, 2000, pp. 123-138.
6. Y. H. Yu, C. C. Chang, and Y. C. Hu, "Hiding secret data in images via predictive coding," *Pattern Recognition*, Vol. 38, 2005, pp. 691-705.
7. Y. K. Lee and L. H. Chen, "A high capacity image steganographic model," *IEE Proceedings of Vision*, *Image and Signal Processing*, Vol. 147, 2000, pp. 288-294.
8. R. Z. Wang, C. F. Lin, and J. C. Lin, "Image hiding by optimal LSB substitution and genetic algorithm," *Pattern Recognition*, Vol. 34, 2001, pp. 671-683.
9. R. Z. Wang, C. F. Lin, and J. C. Lin, "Hiding data in images by optimal moderately significant-bit replacement," *Electronics Letters*, Vol. 36, 2000, pp. 2069-2070.
10. C. K. Chan and L. M. Cheng, "Improved hiding data in images by optimal moderately significant-bit replacement," *Electronics Letters*, Vol. 37, 2001, pp. 1017-1018.
11. C. C. Chang, J. Y. Hsiao, and C. S. Chan, "Finding optimal least-significant-bit substitution in image hiding by dynamic programming strategy," *Pattern Recognition*, Vol. 36, 2003, pp. 1583-1595.
12. C. C. Thien and J. C. Lin, "A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function," *Pattern Recognition*, Vol. 36, 2003, pp. 2875-2881.
13. C. K. Chan and L. M. Cheng, "Hiding data in images by simple LSB substitution," *Pattern Recognition*, Vol. 37, 2004, pp. 469-474.
14. C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., NJ, 1982, pp. 251-252.
15. M. S. Yu and C. H. Yang, "An $O(n)$ time algorithm for maximum matching on cographs," *Information Processing Letter*, Vol. 47, 1993, pp. 89-93.
16. C. H. Yang, S. J. Chen, J. M. Ho, and C. C. Tsai, "Efficient routability check algorithms for segmented channel routing," *ACM Transactions on Design Automation of Electronic Systems*, Vol. 5, 2000, pp. 735-747.

Cheng-Hsing Yang (楊政興) received the B.S. and M.S. degrees in Applied Mathematics from National Chung Hsing University, Taiwan, in 1990 and 1992, respectively, and the Ph.D. degree in Electrical Engineering from National Taiwan University, Taiwan, in 1997. Currently, he is a Full Professor in the Department of Computer Science, National Pingtung University of Education, Taiwan. His current research interests include information hiding and image watermarking.

Shiuh-Jeng Wang (王旭正) received his Ph.D. degree in Electrical Engineering at National Taiwan University, Taipei, Taiwan in 1996. He is a full professor with Department of Information Management at Central Police University, Taoyuan, Taiwan, where he directs the Information Cryptology and Construction Laboratory (ICCL, http://hera.im.cpu.edu.tw). He was a recipient of the 5th Acer Long-Tung Master Thesis Award and the 10th Acer Long-Tung Ph.D. Dissertation Award in 1991 and 1996, respectively. Dr. Wang was a visiting scholar of Computer Science Department at Florida State University (FSU), USA in 2002 and 2004. He also was a visiting scholar of Department of Computer and Information Science and Engineering at University of Florida (UF) from Aug. 2004 to Feb. 2005. Dr. Wang academically toured the CyLab with School of Computer Science in Carnegie Mellon University, U.S.A., in 2007 for international project collaboration inspection. He is also the author/co-author of seven books (in Chinese versions): *Information Security*, *Cryptography and Network Security*, *State of the Art on Internet Security and Digital Forensics*, *Eyes of Privacy – Information Security and Computer Forensics, Information Multimedia Security*, *Computer Forensics and Digital Evidence*, and *Computer Forensics and Security Systems*, published in 2003, 2004, 2006, 2007, and 2009 respectively. He is also a member of the IEEE and ACM. His current interests include information security, digital investigation and computer forensics, steganography, cryptography, data construction and engineering.