

在黑白影像中用區塊圖樣編碼法作失真最小化之動態規劃式資料隱藏

A Dynamic-Programming Approach to Data Hiding in Binary Images Using Block Pattern Coding with Distortion Minimization*

I-Shi Lee (李義溪)[#] and Wen-Hsiang Tsai (蔡文祥)[✝]

Department of Computer and Information Science

National Chiao Tung University

Hsinchu, Taiwan 300

Republic of China

E-mails: gis87809@cis.nctu.edu.tw, whtsai@cis.nctu.edu.tw

摘要

本篇論文提出一種在黑白影像中用區塊圖樣編碼法作失真最小化之動態規劃式資料隱藏的新方法。在資料隱藏過程中，二進位的秘密資料被隱藏於原圖中特殊的 2×2 區塊中，而隱藏的方式是改變影像區塊中的像素值至一圖樣，而此圖樣係根據一種新的圖樣編碼法，用以代表要被隱藏的秘密資料。抽出隱藏的秘密資料則是靠比對圖中特殊區塊的圖樣而達成。為了在黑白影像中隱藏更多資料，並同時維持影像的品質，我們建議使用多個區塊圖樣編碼表並選擇一個最佳的編碼表來使用。我們也使用貪婪及動態規劃搜尋演算法來將秘密影像的失真最小化。實驗結果顯示不僅平均一個區塊可以隱藏更多的資料，而且也降低了失真率，此證明我們的方法確實有效。

關鍵詞：資料隱藏，黑白影像，區塊圖樣編碼法，動態規劃法，貪婪搜尋，失真最小化

Abstract

A new approach to data hiding in binary images based on block pattern coding and the dynamic programming technique is proposed. A binary secret value is embedded into a 2×2 block in a cover image by changing the block pixel values into a pattern which represents the secret value according to a coding scheme proposed in this study. And extraction of the hidden secret

data is accomplished by block pattern decoding. To embed more data in a binary image and maintain the image quality simultaneously, multiple block pattern encoding tables are designed, from which an optimal one is selected for each input image. A dynamic programming algorithm is also designed for data embedding to minimize the resulting distortion in the stego-image. Accordingly, not only more data can be embedded in a block on the average, but also the overall distortion is reduced in an optimal sense. Experimental results are also included to show the effectiveness of the proposed approach.

Key words: data hiding, binary images, block pattern encoding, dynamic programming, greedy search, distortion minimization.

1. Introduction

Many data hiding techniques have been proposed for a variety of applications of digital images in recent years [1, 3-5, 7, 11]. Most of the techniques were proposed for color and gray-scale images because pixels in such images take a wide range of values and so are more proper for data hiding. One simple approach is to use the LSB replacement technique [11] to hide secret data or authentication signals. However, data hiding in binary images is a more challenging work. Because binary image pixels have drastic contrast, it is easier for humans' eyes to find out pixel value changes in binary images. Therefore, it is more difficult to hide data into binary images than into color or gray-level images. Wu et al. [10] embedded secret data in specific image blocks that are selected with higher "flippability" scores by pattern matching. Manipulated flippable pixels on the image region boundary are then used to embed a significant amount of data without causing noticeable

*This work was supported partially by the MOE Program for Promoting Academic Excellency of Universities under the grant number 89-1-FA04-1-4.

[#]Also with Department of Management Information at Kuang Wu Institute of Technology.

[✝]To whom all correspondence should be sent.

artifacts. Pan et al. [6] changed pixel values in image blocks, mapped block contents into the secret data, and used a secret key and a weight matrix to protect the hidden data. Given an image block of size $m \times n$, the scheme can conceal up to $\lfloor \log_2(m \times n + 1) \rfloor$ bits of data in the image by changing, at most, two bits in an image block. Koch and Zhao [2] embedded a bit 0 or 1 in a block by changing the number of black pixels in the block to be larger or smaller than that of white ones, respectively. Tzeng and Tsai [8] encoded the edge features of binary images into 4×4 block patterns, and authenticated the images by pattern matching. Tzeng and Tsai [9] also proposed a new feature, called surrounding edge count, for measuring the structural randomness in a 3×3 image block, and defined “pixel embeddability” from the viewpoint of minimizing image distortion. Accordingly, embeddable image pixels suitable for hiding secret data can be selected.

In a binary image, there are only two pixel values, 0 and 1, and the corresponding pixels may be called *black* and *white* ones, respectively. When data are embedded in a binary image, the image pixels will be changed from black to white or from white to black. The distortion rate is 50% in general data hiding methods for binary images. We propose in this paper a new approach to data hiding based on a block pattern coding technique and a dynamic programming algorithm. The approach can be used to embed more data in a block of a binary image, and minimize the resulting stego-image distortion simultaneously.

In order to embed more data in a binary image, more pixels need be changed; however, the quality of the resulting stego-image will get worse. On the contrary, in order to maintain the quality of the resulting image, the amount of the embedded data should be limited. The proposed approach is designed to be a compromise between the embedded data volume and the resulting image distortion. Our method can extract embedded data without referencing the original image. It also has the merit of concealing up to *three* data bits in a 2×2 block by changing the smallest number of bits in a block. Contrastively, most existing methods for hiding data in binary images can embed only one or two data bits in a 2×2 image block [6, 8].

In the remainder of this paper, the proposed method for dealing with 2×2 image blocks is first described in Section II. Some experimental results are shown in Section III, followed by a conclusion in Section IV.

II. Proposed Data Embedding Method

The proposed method is designed to hide se-

cret data behind binary images in random fashions controlled by secret keys. The method consists of a data embedding process and a data extraction process. In this section, the ideas behind the proposed method are presented first, followed by the details of the proposed data embedding and extraction processes.

A. Encoding Block Patterns for Secret Data Representation

In order to embed secret data into a binary cover image, every 2×2 block of the cover image is regarded as a *pattern* with a corresponding 4-bit *binary value* in this study, with each black pixel representing a bit 0 and each white one representing 1. An illustration is shown in Figure 1. Therefore, in a 2×2 block, possible binary values of the block pattern are 0000_2 through 1111_2 , where “ 0000_2 ” means an entirely black block while “ 1111_2 ” means an entirely white one.

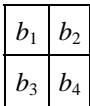

A 2×2 block pattern	Corresponding binary value
	$b_1 b_2 b_3 b_4$
	0110

Figure 1. Illustration of block patterns and corresponding binary values.

The main idea of the proposed data hiding method is based on the use of a *block pattern encoding table* which maps each block pattern into a certain code for use as hidden data with the code being up to three bits in length. And data embedding is accomplished by changing the block patterns so that the codes of the resulting blocks become just the input secret data to be embedded. A *block pattern encoding table* designed for use in this study is shown in Table 1. The idea behind the design of this table is described as follows. It is emphasized by the way that such a table is just one of the many possible ones all of which are usable for data hiding, and the proposed approach chooses an optimal table among them for each specific input image.

The number of possible patterns in a 2×2 block are 16. This number is much larger than the need to represent the two secret bits ‘0’ and ‘1’, so we may use multiple block patterns to represent a single secret value, allowing the possibility of choosing among the patterns an optimal one to replace the original image block in

the data embedding process, thus reducing the resulting image distortion in the replaced block. Furthermore, we wish to embed more data in a block, and for this goal we may use a block pattern to represent more than one bit of secret data.

For example, we may use both the block pattern $t_1 = 1101_2$ and the pattern $t_2 = 1110_2$ to represent the two-bit secret value $s = 00_2$. In this way, when we want to embed, for example, the secret value $s = 00_2$ into a block B with pattern $v = 0110_2$, we have the *two* choices of block patterns $t_1 = 1101_2$ and $t_2 = 1110_2$ instead of the conventional case of just *one*, from which we can choose $t_2 = 1110_2$ to replace the pattern $v = 0110_2$ of the block B , resulting in the smaller distortion of just a 1-bit error. Note that if only the choice of $t_1 = 1101_2$ is allowed, then the error will be 3 bits which mean a larger distortion in the replaced block. It is such allowance of multiple choices for block pattern replacement that achieves distortion reduction in the proposed approach.

Accordingly, we group in this study the 16 possible block patterns in a 2×2 block B into distinct sets according to the *entropy* values E of the block patterns, where an entropy value E of a block pattern P is defined as follows:

$$E = - \sum_k p_k \log_2 p_k = -p_0 \log_2 p_0 - p_1 \log_2 p_1$$

where p_0 and p_1 respectively are the occurrence probability values of black and white pixels appearing in P computed as

$$p_0 = (\text{number of black pixels in } B)/4;$$

$$p_1 = (\text{number of white pixels in } B)/4.$$

By the above definition, there are three possible entropy values 0, 0.811, and 1 in a 2×2 block. A pattern P in a set with a higher entropy value E is presumably more random in its black and white arrangement, and so is more suitable for hiding more secret data without causing a noticeable change. We summarize the above discussions about secret value encoding in Table 1.

B. Sketch of proposed data hiding idea

In the proposed data embedding process, the more data we embed in a 2×2 block, the worse the resulting image quality becomes. Therefore, we must control the number of destructed pixels in a block to reduce the image distortion. The idea of the data embedding process we propose in this study is sketched as four major steps in the follow, which includes two folds of distortion minimization.

- (I). **Checking the data embeddability of each block:** If a 2×2 image block is entirely black or white, discard it as unsuitable for data embedding.
- (II). **Randomizing the embedding position:** Use a secret key as well as a random number generator to select randomly a sequen-

tial list blocks for data embedding. This enhances the security of the hidden data from being accessed.

- (III). **Using multiple block pattern encoding tables for the first-fold distortion reduction:** Generate all possible block pattern encoding tables and select an optimal one in the data embedding process, in order to decrease the distortion.

- (IV). **Applying optimal search techniques for the second-fold distortion reduction:** We design cost functions for measuring degrees of block pattern changes for use in optimizing the block pattern replacement operations in the data embedding process. Apply the optimal search technique of dynamic programming to segment the input secret data stream optimally into codes and embed them in the input image according to a cost function designed in advance for measuring the degree of the pattern change in each image block. This reduces the resulting distortion further in a global sense.

C. Use of Multiple Block Pattern Encoding Tables

The first distortion-reduction technique using multiple block pattern encoding tables, as mentioned previously in the third major step of the proposed data; embedding process, is based on the idea that a single encoding table will not be suitable for every binary image in the embedding process. If a binary image is destroyed very seriously in the data embedding process using Table 1, it will be necessary to use another table with other combinations of block patterns and encoded hidden data. For example, assume that a binary secret value $v = 101_2$ is to be embedded into a sequence of three randomly selected image blocks with patterns 0000, 0100, and 1111 by Table 1. The data embedding process using Table 1, as illustrated in Fig. 2(a), will select optimally the block pattern type D2 = 1001, which encodes the three-bit secret value $v = 101_2$, to replace the first selected block with pattern 0000, resulting in reversing two bits. However, if we replace the encoded secret data of type A in Table 1 with those of type F, and replace those of all of types B1 through B4 with those of all of types E1 through E4, respectively, then we will get a new block pattern encoding table and the use of it to hide the secret value $v = 101_2$ will result in no bit reversing because here we can, as illustrated in Fig. 2(b), select in sequence optimally the new pattern type F = 0000 (encoding the secret data of 1_2) and the new pattern type E3 = 0100 (encoding the secret data of 01_2) to encode together the secret data $v = 101_2$. This means that adaptive table generations and selections for use in data embedding help distortion reduction indeed. More generally, by enumerat-

ing all possible ways for exchanging the encoded secret data of certain types in Table 1 with those of the other types, we can get 128 distinct block pattern encoding tables for selection in the data embedding process to minimize the distortion.

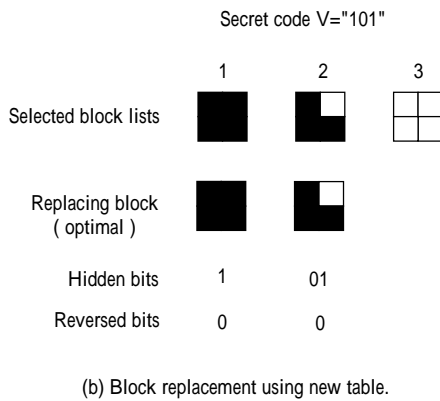
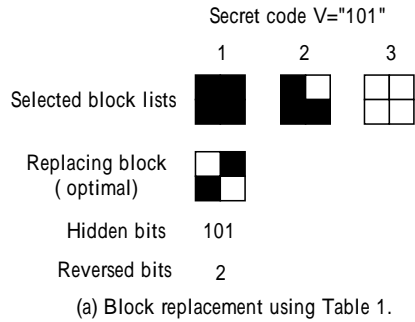


Figure 2. An example of proposed data embedding process.

D. Proposed Distortion-Minimizing Cost Function and Search Techniques for Optimal Solutions

The cost function proposed in this study for use in the proposed data embedding process to minimize image distortion is *the total number of reversed bits* in the resulting stego-image. In Table 1, block patterns can be used to encode one, two, or three secret bits. Correspondingly, we hide a binary secret value v by embedding the first one, two, or three bits in the prefix of v into a block. To determine how many bits should be embedded, we may calculate first the cost function value for each of the three cases, and then replace the currently selected block with the block pattern which corresponds to the case with the minimum cost function value. This method provides a quick way for data embedding. However, it is actually a *greedy search* and *not* an optimal solution.

To see this, for example, for the previously-mentioned example in which the secret value v of 101_2 is embedded in three selected blocks with patterns 0000, 0100, and 1111 by Table 1, by the above-mentioned greedy algorithm we first replace the block with pattern 0000 by the block pattern E3 = 0100 to embed

two bits 10. The computed cost function value is 1 because a bit is reversed here. This cost is a local minimal one. Next, we replace the block with pattern 0100 by the block pattern A= 1111 to embed the last bit 1 of v , and get a local minimal cost value 3. The total cost value is 4. Now, if we do not use the greedy algorithm from the beginning, and replace instead the first block with pattern 0000 by the block pattern D2 = 1001 to embed three bits 101 directly, then the total cost value will be reduced to 2 which is smaller than the previous total cost 4. This shows that there indeed exist at least one solution better than that found by the greedy method. Figure 3 illustrates the data embedding process for this example. This is also true for many other examples, as found by this study. And so the search of an optimal solution is meaningful, for which the proposed method is dynamic programming.

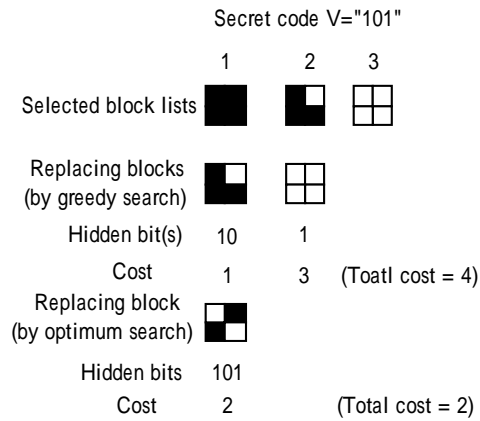


Figure 3. An example of proposed data embedding process.

In the proposed dynamic programming algorithm (abbreviated as DP in the sequel), certain edit distances are defined to minimize the cost function, as described in the following. Assume that the input secret data value to be hidden is in the form of an n -bit string S_1 with $S_1[i]$ denoting its $(i+1)$ th bit. Also, let the randomly selected blocks for embedding the secret value be expressed as a list in the form of another string S_2 with $S_2[i]$ denoting its $(i+1)$ th block. Only one type of edit operation, namely, *replacement*, is needed for use in the proposed algorithm to represent the image block replacement operations involving S_1 and S_2 in the proposed secret data embedding process. The edit distance of S_1 and S_2 is defined, according to the previous discussions, as the minimum cost to transform S_2 into S_1 by edit operations according to an optimal block pattern encoding table used in the data embedding process. Let C be a two-dimensional *cost matrix* with its element $C[i, j]$ denoting the minimum cost to transform a

substring of S_2 with bits $S_2[j]$ through $S_2[n-1]$ into a substring of S_1 with bits $S_1[i]$ through $S_1[n-1]$. Then $C[0, 0]$ is the value of the minimum cost to transform S_2 into S_1 . Also, let RC be a *replacement cost function* with its element $RC(L, i, j)$ denoting the cost for replacing the $(j+1)$ th block in S_2 , denoted by $S_2[j]$, with the block patterns encoding the initial L bits of a substring of S_1 with bits $S_1[i]$ through $S_1[n-1]$, where L may be 1, 2, or 3. By the above definitions, the value $C[i, j]$ is recursively just the value of the minimum of all possible values of $RC(L, i, j) + C[i+L, j+1]$, where $L = 0, 1, \text{ and } 3$. And because of this, the size of C must be expanded to $(n+2) \times n$. Furthermore, those elements of C with indices larger than $n-1$ should be given certain values (0 or ∞) to specify their correspondences to “*boundary conditions*”. Then, according to the dynamic programming technique, the minimum edit distance may be computed by the following recursive formulas:

Set initial values

$$\begin{aligned} C[n, j] &= 0, & j &= 0, 1, 2, \dots, n, \\ C[n+1, j] &= 0, & j &= 0, 1, 2, \dots, n, \\ C[n+2, j] &= 0, & j &= 0, 1, 2, \dots, n, \\ C[i, n] &= \infty, & i &= 0, 1, 2, \dots, n-1, \\ C[i, j] &= \infty, & i, j &= 0, 1, 2, \dots, n-1; \end{aligned}$$

and then for all $i = 0, 1, \dots, n-1, j = 0, 1, \dots, n-1$, compute

$$C[i, j] = \min\{RC(1, i, j) + C[i+1, j+1], RC(2, i, j) + C[i+2, j+1], RC(3, i, j) + C[i+3, j+1]\}.$$

Algorithm 1. *Computing minimum cost for minimizing distortion in data embedding process by DP.*

Input: an n -bit secret code string S_1 , a string of n randomly selected blocks S_2 , a block pattern encoding table, a two-dimensional cost matrix $C[i, j]$, for $i = 0, 1, \dots, n+2, j = 0, 1, \dots, n$ with the initial values specified in the above recursive formulas, a two-dimensional *index matrix* $I[i, j]$, for $i = 0, 1, \dots, n-1, j = 0, 1, \dots, n-1$, for recording the relative indices in the block pattern encoding table after calculating $C[i, j]$, and a two-dimensional matrix $N[i, j]$, for $i = 0, 1, \dots, n-1, j = 0, 1, \dots, n-1$, for recording the relative next step after calculating $C[i, j]$ with each element given an initial value of minus one.

Output: $C[i, j]$, the minimum cost to change the substring $S_2[j]$ through $S_2[n]$ into $S_1[i]$ through $S_1[n]$, $I[i, j]$, and $N[i, j]$.

Steps:

1. If $C[i, j]$ is equal to an infinitive value ∞ , continue the next step; else go to Step 4.
2. Calculate three temporary cost functions $T[1]$, $T[2]$, and $T[3]$, record every next step and the corresponding value as the indices index1 , index2 , and index3 of the block pattern encoding table which is used in calculating the minimal cost in $RC(1, i, j)$, $RC(2, i, j)$, and $RC(3, i, j)$, respectively, in the following way:
 - 2.1 $T[1] = RC(1, i, j) + C(i+1, j+1)$, $\text{next_step}[1] = i+1$, and acquire index1 .
 - 2.2 $T[2] = RC(2, i, j) + C(i+2, j+1)$, $\text{next_step}[2] = i+2$, and acquire index2 .
 - 2.3 $T[3] = RC(3, i, j) + C(i+3, j+1)$, $\text{next_step}[3] = i+3$, and acquire index3 .
3. Take $C(i, j)$ to be the minimum of the three temporary cost functions, record the corresponding relative next step in $N[i, j]$, and record the relative index in the block pattern encoding table in $I[i, j]$.
4. Return $C[i, j]$.

Because every next step and the used indices of the block pattern encoding table have been recorded, we can reconstruct the embedding process easily. The space complexity and time complexity are both $O(n^2)$ for the DP. Now, the proposed data embedding process is described in detail as an algorithm in the following.

Algorithm 2. *Data embedding process using block pattern encoding tables and DP.*

Input: a binary image I , a secret data string S_1 with n bits, a secret key K as well as a random number generator f , and 128 block pattern encoding tables.

Output: a stego-image S , an optimal block pattern encoding table B , a length of block list L , and a minimal total cost C_{\min} .

Steps:

1. Get a list of embeddable 2×2 blocks from the input image I in a way as described previously.
2. Set the value of the desired minimal total cost C_{\min} to be infinitive.
3. For each block pattern encoding table B_i among the 128 possible ones, perform the following operations.
 - 3.1 Calculate a total cost C_i using B_i and the DP.
 - 3.2 If C_{\min} is larger than C_i , perform the following operations.
 - a. Take C_i as the minimal total cost

- c. C_{\min} .
 - b. Set the optimal block pattern encoding table B as B_i .
 - c. Sequentially, record every index obtained from Step 3.1 according to the next-step matrix N and index matrix I , until an element of N is equal to -1 . Meanwhile, calculate L , the length of the block list.
4. Replace the minimal cost block list with the selected block list of binary image I by the recorded index sequence of block pattern encoding table B and the length of the block list L .
 5. Take the final result as the desired stego-image S .

E. Data Recovery Process

The goal of the proposed data recovery process is to extract the embedded bit stream from a stego-image. It is easy to finish the extraction process as follows.

Algorithm 3. Secret data recovering process.

Input: a stego-image I' presumably including a secret bit stream S ; and the secret key K as well as the random number generator f used in the data embedding process; the index table B that points out which table is used in the embedding process, and the length of the block list L .

Output: the secret bit stream S .

Steps:

1. Extract a list of 2×2 embeddable blocks from the stego-image I' by the secret key K , the random number generator f , and the length L .
2. For each 2×2 embeddable block in I' , compute the corresponding block pattern P , and look P up in the table B to decode the data bits embedded in the block.
3. Take all the extracted data bits in sequence as the desired secret bit stream S .

III. Experimental Results

Some experimental results of applying the proposed method are shown here. Two streams of secret data were generated by a random fashion. One is a stream of 2432 bits, which was embedded into the image shown in Figure 4(c). The other is 992-bit long, which was embedded into the binary image shown in Figure 4(d). Figures 4(a) and 4(b) show two binary cover images of the sizes of 687×534 and 512×512 , respectively. And the stego-images after embedding the secret data using the DP algorithm are shown in Figures 4(c) and 4(d), respectively. Figures 4(e) and 4(f) show the respective images of differences between Figures 4(a), 4(b) and Figures

4(c), 4(d) in terms of white pixels. Note that the backgrounds of the images of Figures 4(e) and 4(f) are shown in gray values to emphasize the difference positions.

Tables 2 shows the statistical data of the stego-images of Figures 4(a) and (b) for the proposed algorithms, in which we list the numbers of the selected table index, the used blocks, the minimal cost values and the length of secret data. The minimum cost values show that the DP is the best, the greedy algorithm using an optimal encoding table among the 128 possible one is the next, and the greedy algorithm using just an encoding table is the worst. For other images, similar results can be observed. For the images shown here, the average number of secret data in a block, using the DP algorithm, is about 1.7 bits. And the distortion rate computed as the ratio of the number of reversed bits to the length of the secret data, using the DP algorithm, is in the range from 0.37 to 0.39, which is smaller than 0.5 yielded by most exist data hiding methods for binary images.

IV. Discussions and Summary

A novel optimal method for hiding secret data into binary images with a distortion minimization effect and a larger data embedding capability has been proposed. An optimal block pattern encoding table is chosen from 128 alternative ones for use in the proposed data embedding process to minimize distortion in the stego-image. The method can minimize further the distortion using the dynamic programming technique and can embed up to three bits in a 2×2 image block. Therefore, by our method, not only more data can be embedded in a binary image, but also the distortion rate of the stego-image can be effectively reduced. The proposed method is based on the use of 2×2 blocks in data embedding process. The future works may be directed to designing a better cost function for the human visual system, constraining certain conditions for the cost function to find a better image quality, and finding a better encoding table for replacing selected blocks to reduce stego-image distortion further.

References

- [1] S. Katzenbeisser and F. A. P. Petitolas, *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House, Boston, U. S. A., 2000.
- [2] E. Koch and J. Zhao, "Embedding robust labels into images for copyright protection," *Proceedings of International Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Techniques*, pp. 242-251, Munich, Germany, 1995.

- [3] D. Kundur, "Energy allocation principles for high capacity data hiding," *Proceedings of IEEE International Conference on Image Processing*, Vancouver, Canada, vol. 1, pp. 423-426, September 2000.
- [4] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc., Englewood Cliffs, NJ, U. S. A., 1983.
- [5] L. M. Marvel, J. C. G. Boncelet, and C. T. Retter, "Spread spectrum image steganography," *IEEE Transactions on Image Processing*, vol. 8, no. 8, pp. 1075-1083, August 1999.
- [6] H. K. Pan, Y. Y. Chen, and Y. C. Tseng, "A secure data hiding scheme for two-color images," *IEEE ISCC 2000*, pp. 750-755, 2000.
- [7] M. Swanson, M. Kobayashi, and A. Tewfik, "Multimedia data-embedding and watermarking technologies," *Proceedings of the IEEE*, vol. 86, pp. 1064-1088, 1998.
- [8] C. H. Tzeng and W. H. Tsai, "A new technique for authentication of images/videos for multimedia applications," *Proceedings of ACM Multimedia 2001 Workshops – Multimedia and Security: New Challenges*, Ottawa, Ontario, Canada, pp. 23-26, Oct. 2001.
- [9] C. H. Tzeng and W. H. Tsai, "A new approach to authentication of binary images for multimedia communication with distortion reduction and security enhancement," accepted and to appear in *IEEE Communications Letters*.
- [10] M. Wu, E. Tang, and B. Liu, "Data hiding in digital binary image," *Proceedings of IEEE International Conference on Multimedia & Expo 2000 (ICME'00)*, New York, New York City, 2000.
- [11] D. C. Wu and W. H. Tsai, "Spatial-domain image hiding using an image differencing," *IEE Proceedings – Vision, Image, and Signal Processing*, vol. 147, no. 1, pp. 29-37, 2000.

Table 1. Proposed block pattern encoding table.

Type	Block pattern	Entropy value	Corresponding binary value	Encoded secret data	Type	Block pattern	Entropy value	Corresponding binary value	Encoded secret data
A		0	1111	1	F		0	0000	0
B1		0.811	1110	00	E1		0.811	0001	11
B2		0.811	1101	00	E2		0.811	0010	11
B3		0.811	1011	01	E3		0.811	0100	10
B4		0.811	0111	01	E4		0.811	1000	10
C1		1	0011	011	D1		1	0110	100
C2		1	0101	011	D2		1	1001	101
C3		1	1010	010					
C4		1	1100	010					

Table 2. Statistics of two stego-images for three proposed algorithms, with the cost function defined as the sum of the number of reversed bits in a block.

stego-image	Algorithm	Table index	Used blocks	Cost	Secret data length
NCTU	Greedy (using a fixed table)	0	1528	1153	2432
	Greedy (using optimal one among 128)	16	1526	1115	
	DPED	26	1418	954	
Lena	Greedy (using a fixed encoding table)	0	621	431	992
	Greedy (using optimal one among 128)	30	637	401	
	DPED	41	582	369	



(a)



(b)



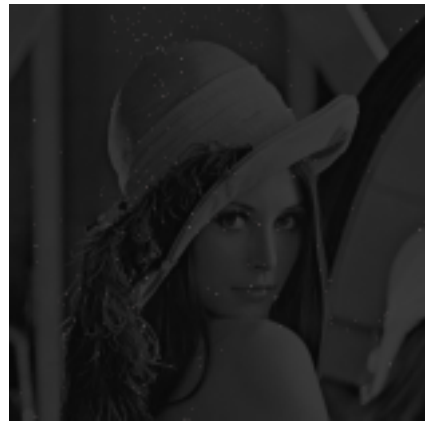
(c)



(d)



(e)



(f)

Figure 4. Input binary images, output stego-images with secret data, and the differences. (a) Binary image “NCTU”. (b) Binary image “Lena”. (c) and (d) Stego-images after embedding secret data, respectively. (e) and (f) The difference pixels after embedding secret data, respectively.