Data Hiding in 2-Color Images

Yu-Chee Tseng, *Member*, *IEEE Computer Society*, and Hsiang-Kuang Pan

Abstract—In an earlier paper [4], we proposed a steganography scheme for hiding a piece of critical information in a host binary image. That scheme ensures that, in each $m \times n$ image block of the host image, as many as $\lfloor \log_2(mn+1) \rfloor$ bits can be hidden in the block by changing at most 2 bits in the block. In this paper, we propose a new scheme that improves [4] in its capability to maintain higher quality of the host image after data hiding by sacrificing some data hiding space. The new scheme can still offer a good data hiding ratio. It ensures that, for any bit that is modified in the host image, the bit is adjacent to another bit which has a value equal to the former's new value. Thus, the hiding effect is quite invisible.

Index Terms—Binary image processing, coding, cryptography, information hiding, prisoners' problem, security, steganography.

1 Introduction

As digital media are gaining wider popularity, their security-related issues are becoming a greater concern. One central issue is *confidentiality*, which is typically achieved by *encryption*. However, as an encrypted message usually flags the importance of the message, it also attracts cryptanalysts' interests. The sometimes confusing terminology, *steganography*, has a different flavor from encryption; its purpose is to embed a piece of critical information in a noncritical host message (e.g., webpages, advertisements, etc.) to distract opponents' attention [9], [16]. The existence of the critical information in the host message should be hardly detectable. One less confusing name for steganography would be *data hiding*. It should be understood that, steganography is orthogonal to encryption, and it may be combined with encryption to achieve a higher level of security.

The study of this subject may be traced back to [13], where the *Prisoners' Problem* was proposed. In this scenario, Alice and Bob are in jail and wish to hatch an escape plan. All their communications must go through the warden, Willie if Willie detects any encrypted messages, he will frustrate their plan by throwing them into solitary confinement. So, they must find some way to hide their plaintext (or ciphertext) in an innocuous-looking covertext. The history and bandwidth concerns of the *subliminal channel* are discussed in [14], [15]. Information hiding has applications in many commercial, military, and anticriminal-related issues. Classifications and surveys of information hiding can be found in [2], [11].

In the context of images, data hiding is usually achieved by alternating some nonessential information in the host image. Given a color image, one simple approach is to use the least-significant bit (LSB) of each pixel to hide information [19]. As this is not likely to degrade the quality of the image much, a number of software packages have adopted this approach [21]. For instance, the Kodak Photo CD has a maximum resolution of $2,048 \times 3,072$ pixels, each represented by 24 bits using the RGB format. Thus, such a picture can hide about 2.36 megabytes of data. To further reduce the hiding effect on the image quality, a genetic algorithm is proposed in [20]. A hiding scheme based on the conventional keystream generator is proposed in [5]. Information hiding for security documents (such as currency and bill) is discussed in [8]. References [1], [2] consider how to apply public-key cryptography to steganography. Steganography for images is discussed in [17]. A broad review of steganography is in [2], [3], [10].

One less frequently addressed, but very challenging, problem is data hiding in a two-color/black-and-white image (such as facsimiles, xeroxs, and bar codes). The reason is that changing a pixel in such an image can be easily detectable. References [23], [24] deal with this subject. The image quality after hiding is further taken into account in [22]. A data hiding scheme using the statistical properties of dithered imagery, images that represent each pixel value with a pattern of dots, is proposed by [18], which can accommodate 2 kilobytes of hidden information in a binary 256×256 image. Data hiding for halftone images is addressed in [6], [7]. Watermarking on binary images is discussed in [12]. The scheme proposed in [4] significantly improves the data hiding capacity, where it is shown that, for each $m \times n$ block of the host image, the block can accommodate as many as $\lfloor \log_2(mn+1) \rfloor$ bits of secret data by changing at most 2 bits in the block. This is comparatively much more efficient than [22], [23], [24], which can hide at most one secret bit in each block by changing at most 1 bit in the block.

In this paper, we propose a new scheme that improves [4] in terms of the visibility of the hiding effect. By taking into consideration the quality of the image after hiding, the new scheme can make the hiding effect quite invisible. It ensures that, for any bit that is modified in the host image, the bit must be adjacent to another bit that has the same value as the former's new value. Thus, the existence of secret information in the host image is difficult to detect. Our scheme is secure in this sense. This will be achieved by sacrificing some data hiding space, but the new scheme still offers a good data hiding ratio. Specifically, for each $m \times n$ block of the host image, we will hide as many as $\lfloor \log_2(mn+1) \rfloor - 1$ bits of secret data by changing at most 2 bits in the block. Implementation results are demonstrated to verify the hiding effect of the new scheme.

The rest of this paper is organized as follows: Section 2 reviews our earlier scheme [4]. The revised new scheme is presented in Section 3. Implementation results are in Section 4. Conclusions are drawn in Section 5.

2 Background and Motivation

The data hiding we consider in this paper is a secret-key steganography [10], which can be defined as a quintuple $\langle H, G, K, E_K, D_K \rangle$, where H is the set of possible covers (i.e., host messages), G is the set of secret messages (i.e., guest messages), K is the set of secret keys, $E_K : H \times G \times K \rightarrow H$, and $D_K : H \times K \rightarrow G$. Given $h \in H, g \in G$, and $k \in K$, E_K is the hiding function and D_K is the retrieving function such that

$$\begin{aligned} h' &= E_K(h, g, k) \\ D_K(h', k) &= D_K(E_K(h, g, k), k) = g. \end{aligned}$$

To distract the opponents, it should be hardly detectable that h' has been hidden with information. Under the context of using audio/video/image as the host message, this could mean that h' sounds/looks almost like the original h.

2.1 A Review

Below, we review the data hiding scheme proposed by Chen et al. [4], which is shown in Fig. 1 and is abbreviated as the *CPT* scheme. Through the review, we will identify some image quality problems associated with it. This will motivate the work in this paper.

- *Y.-C. Tseng is with the Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsin-Chu, 300, Taiwan. Email: yctseng@csie.ncu.edu.tw.*
- *H.-K. Pan is with the Department of Computer Science and Information Engineering, National Central University, Chung-Li, 320, Taiwan.*

Manuscript received 7 Nov. 2000; revised 28 June 2001; accepted 1 Oct. 2001. For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 113118.

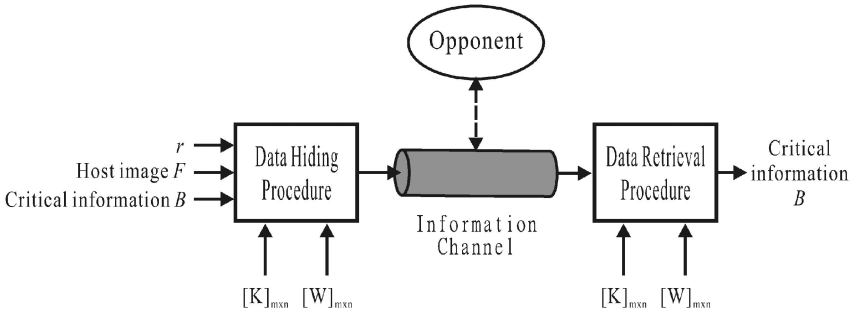

Fig. 1. Block diagram of the CPT data hiding scheme.

In the CPT scheme, we are given a host binary image F. F will be partitioned into blocks of fixed size $m \times n$ (for simplicity, we assume that F's size is a multiple of $m \times n$). The scheme is able to hide as many as $r \leq \lfloor \log_2(mn+1) \rfloor$ bits of the guest message in each host block by modifying at most 2 bits in the block. The secret key has two components:

- K: a randomly selected binary matrix of size $m \times n$.
- W: a *weight matrix* which is an integer matrix of size $m \times n$. W satisfies the condition that

$$\{[W]_{i,j} | i = 1..m, j = 1..n\} = \{1, 2, \ldots, 2^r - 1\}.$$

Note that, since $2^r \leq 2^{\lfloor \log_2(mn+1) \rfloor} \leq 2^{\log_2(mn+1)} \leq mn + 1$, it is trivial to find a matrix that can serve as a weight matrix. In fact, the number of choices for W can be calculated by first picking $2^r - 1$ entries from the $m \times n$ matrix and assigning $\{1, 2, \ldots, 2^r - 1\}$ to these entries, in any order, followed by assigning any value to each of the remaining $mn - (2^r - 1)$ entries:

$$C^{mn}_{2^r-1} * (2^r - 1)! * (2^r - 1)^{mn-(2^r-1)}.$$

For instance, if $m = n = 8$ and $r = 5$, there are $C^{64}_{31} * 31! * 31^{33}$ possible Ws. This number should be large enough to prevent a brute-force attack.

The data hiding is achieved by modifying some bits of F. Below, we show how to hide a bit stream $b_1 b_2 \ldots b_r$ into an $m \times n$ host block, say, F_i. More details can be found in [4].

1. Compute $F_i \oplus K$, where $\oplus$ means the bitwise exclusive-OR of two equal-size binary matrices.
2. Compute $SUM((F_i \oplus K) \otimes W)$, where $\otimes$ means the pairwise multiplication of two equal-size matrices and SUM means the sum of all elements in a matrix.
3. From the matrix $F_i \oplus K$, compute for each $w = 1..2^r - 1$ the following set:

$$S_w = \Big\{ (j,k) \,|\big(([W]_{j,k} = w) \wedge \big([F_i \oplus K]_{j,k} = 0\big)\big) \vee \big(([W]_{j,k} = 2^r - w) \wedge \big([F_i \oplus K]_{j,k} = 1\big)\big) \Big\}.$$

   Intuitively, S_w is the set containing every matrix index (j,k) such that, if we complement $[F_i]_{j,k}$, we can increase the sum in Step 2 by w. There are actually two possibilities to achieve this: i) if $[W]_{j,k} = w$ and $[F_i \oplus K]_{j,k} = 0$, then complementing $[F_i]_{j,k}$ will increase the weight by w and ii) if $[W]_{j,k} = 2^r - w$ and $[F_i \oplus K]_{j,k} = 1$, then complementing $[F_i]_{j,k}$ will decrease the weight by $2^r - w$ or, equivalently, increase the sum by w (under $\bmod 2^r$). Also, define $S_w = S_{w'}$ for any $w \equiv w' \pmod{2^r}$.
4. Define a weight difference

$$d \equiv (b_1 b_2 \ldots b_r) - SUM((F_i \oplus K) \otimes W) (\bmod 2^r).$$

   If $d = 0$, there is no need to change F_i. Otherwise, we run the following steps to transform F_i to F'_i:

   a. Randomly pick an $h \in \{0, 1, \ldots, 2^r - 1\}$ such that $S_{hd} \neq \emptyset$ and $S_{-(h-1)d} \neq \emptyset$.
   b. Randomly pick a $(j,k) \in S_{hd}$ and complement the bit $[F_i]_{j,k}$.
   c. Randomly pick a $(j,k) \in S_{-(h-1)d}$ and complement the bit $[F_i]_{j,k}$.

   To summarize, the above steps ensure the following invariant:

$$\mathbf{I1}: SUM((F'_i \oplus K) \otimes W) \equiv b_1 b_2 \ldots b_r \pmod{2^r}$$

Step 3 contains the most important definition of set S_w, which contains all matrix indices to increase the SUM by w. In Step 4, we pick two nonempty sets S_{hd} and $S_{-(h-1)d}$. Since these sets indicate the locations where we can complement F_i to increase the weight by hd and $-(h-1)d$, respectively, the overall effect is an increase of the weight by d. It is proven in [4] that Step 4 will always succeed. However, there are logical flaws in the above program, which we left out intentionally for ease of presentation. First, the set S_0 (and, similarly, S_{2^r}, $S_{2 \cdot 2^r}$, $S_{3 \cdot 2^r}$, etc.) is not yet defined. Like other S_ws, we can regard S_0 as the set of indices such that complementing these locations in F_i will result in an increase of weight by 0. Since this can be achieved by changing *nothing* on F_i, we can always regard S_0 as nonempty and, whenever the statement "complement the bit $[F_i]_{j,k}$" is encountered, we simply skip this step. This amendment will make the program logically complete.

To reflect the invariant **I1**, the receiver of F'_i simply recovers the hidden data as follows:

5. On receiving F'_i, the receiver computes $SUM((F'_i \oplus K) \otimes W) \bmod 2^r$ to find the hidden bit stream $b_1 b_2 \ldots b_r$.

For example, let the host image F, secret key K, and weight matrix W be as shown in Fig. 2a. First, F is partitioned into two 4×4 blocks, F_1 and F_2. We can hide $r \leq \lfloor \log_2(4 \times 4 + 1) \rfloor$ bits in each block. Let $r = 3$ and the secret data = 001001 (the first three bits will be embedded in F_1 and the last three bits will be embedded in F_2). The results of $F_1 \oplus K$ and $F_2 \oplus K$ are in Fig. 2b. To embed 001 in F_1, since

$$SUM((F_1 \oplus K) \otimes W) \bmod 2^3 = 24 \bmod 2^3 = 0,$$

the weight difference $d = 1 - 0 = 1$. Thus, swapping $[F_1]_{2,4}$ will increase the weight by $[W]_{2,4} = 1$. To embed 001 in F_2, since $SUM((F_2 \oplus K) \otimes W) \bmod 2^3 = 36 \bmod 2^3 = 4$, the weight difference $d = 1 - 4 = -3$. Thus, swapping $[F_2]_{2,2}$ and $[F_2]_{3,2}$ will

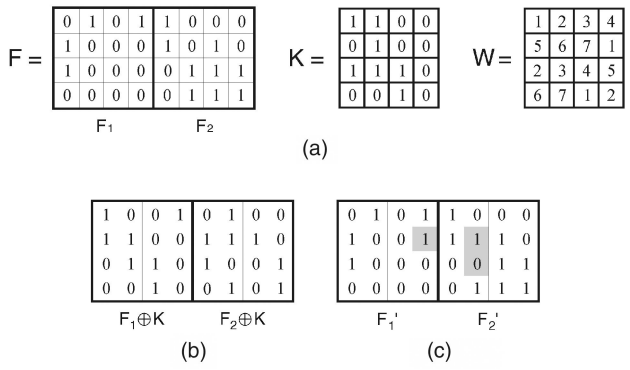

Fig. 2. A data hiding example: (a) the inputs, (b) $F_1 \oplus K$ and $F_2 \oplus K$, and (c) the modified image.

increase the weight by $-[W]_{2,2} = -6$ and $[W]_{3,2} = 3$, respectively. Fig. 2c shows the resulting F (with the modified bits in gray).

2.2 Some Observations

In the above scheme, although at most 2 bits can be modified in each host block, there is no control on the quality of the image after modification. Specifically, the bits that are modified are selected from a random process. It does not take the locations and neighborhood of the modified bits into consideration. The development only focuses on the weight management (i.e., how to increase/decrease the SUM into a desired value). The quality of the image after hiding also needs to be taken care of. This motivates the work in this paper.

3 Data Hiding with Quality Control

In this section, we propose a revised version of the CPT scheme. Our goal is to improve the quality of the image after data hiding. This will be achieved by sacrificing some data hiding space.

3.1 Control of Image Quality after Data Hiding

Since we are working on a 2-color image, changing any bit in the image may be easily detectable. To maintain the quality of the image, certainly we should change as few bits as possible. A completely black or blank host block will not be used to hide data. Also, if a bit has to be changed, we expect that its location be very close to a bit who shares the same value as the former's new value. For instance, consider an image F represented by a matrix, which is modified into two images F' and F'', as follows:

$$F = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} F' = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$F'' = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Both F' and F'' differ from F in one bit. We would regard that F' looks similar to F than F'' does because F' differs from F in a location which is adjacent to an area of 1s. The modified 1 in F'' is more visible.

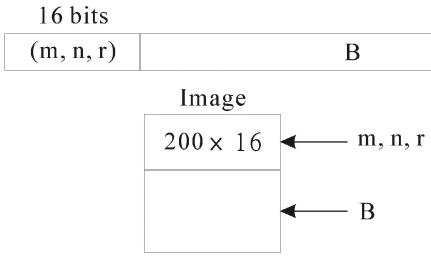

Fig. 3. The placement of parameters and critical data in our implementation.

To formulate the above observation, given an image F, we define a distance matrix $dist(F)$, which is an integer matrix of the same size as F such that

$$[dist(F)]_{i,j} = \min_{\forall x,y} \left\{ \sqrt{|i-x|^2 + |j-y|^2} \mid [F]_{i,j} \neq [F]_{x,y} \right\}.$$

That is, $[dist(F)]_{i,j}$ is the distance from $[F]_{i,j}$ to the closest element $[F]_{x,y}$ such that the complement of $[F]_{i,j}$ is equal to $[F]_{x,y}$. The matrix will later be used to reflect the priority in choosing a bit to be modified. For example, in the earlier example, we have

$$dist(F) = \begin{bmatrix} 2 & 1 & 1 & 2 & 3 \\ \sqrt{2} & 1 & 1 & 2 & 3 \\ 1 & 1 & \sqrt{2} & \sqrt{5} & \sqrt{10} \\ 1 & 1 & 2 & \sqrt{8} & \sqrt{13} \\ 1 & 1 & 2 & 3 & 4 \end{bmatrix}.$$

3.2 The Scheme

Now, we present our scheme. We are given a host image F. Still, F will be partitioned into blocks of size $m \times n$. For simplicity, we assume that the size of F is a multiple of $m \times n$. As mentioned earlier, our scheme will trade some data hiding space for better image quality. Specifically, for each host image block F_i, we will hide, if possible, r bits of data in F_i, where $r \leq \lfloor \log_2(mn+1) \rfloor - 1$. Note that this value of r is one less than that in the CPT scheme.

Let $b_1 b_2 \ldots b_r$ be the binary stream to be hidden in F_i, and F_i' be the modified image after the hiding. If F_i' is not completely black or blank, the scheme will ensure the following invariants:

I2 :$(SUM((F_i' \oplus K) \otimes W) \bmod 2 = 0) \Longrightarrow$
$SUM((F_i' \oplus K) \otimes W)/2 \equiv b_1 b_2 \ldots b_r (\bmod\ 2^{r+1})$

I3 :$(SUM((F_i' \oplus K) \otimes W) \bmod 2 = 1) \Longrightarrow$
(there is no data hidden in F_i')

Compared to **I1**, in the above two invariants, the value calculated by SUM is not completely used for hiding data. In fact, the last bit of SUM indicates whether there is hidden data or not. When the last bit is 1, it means that our scheme cannot find proper bits in the host image to be altered.

Definition 1. *An $m \times n$ matrix W can serve as a weight matrix if $\{[W]_{i,j}, i = 1..m, j = 1..n\} = \{1, 2, \ldots, 2^{r+1} - 1\}$ and for each 2×2 sub-block of W, the sub-block contains at least one odd element.*

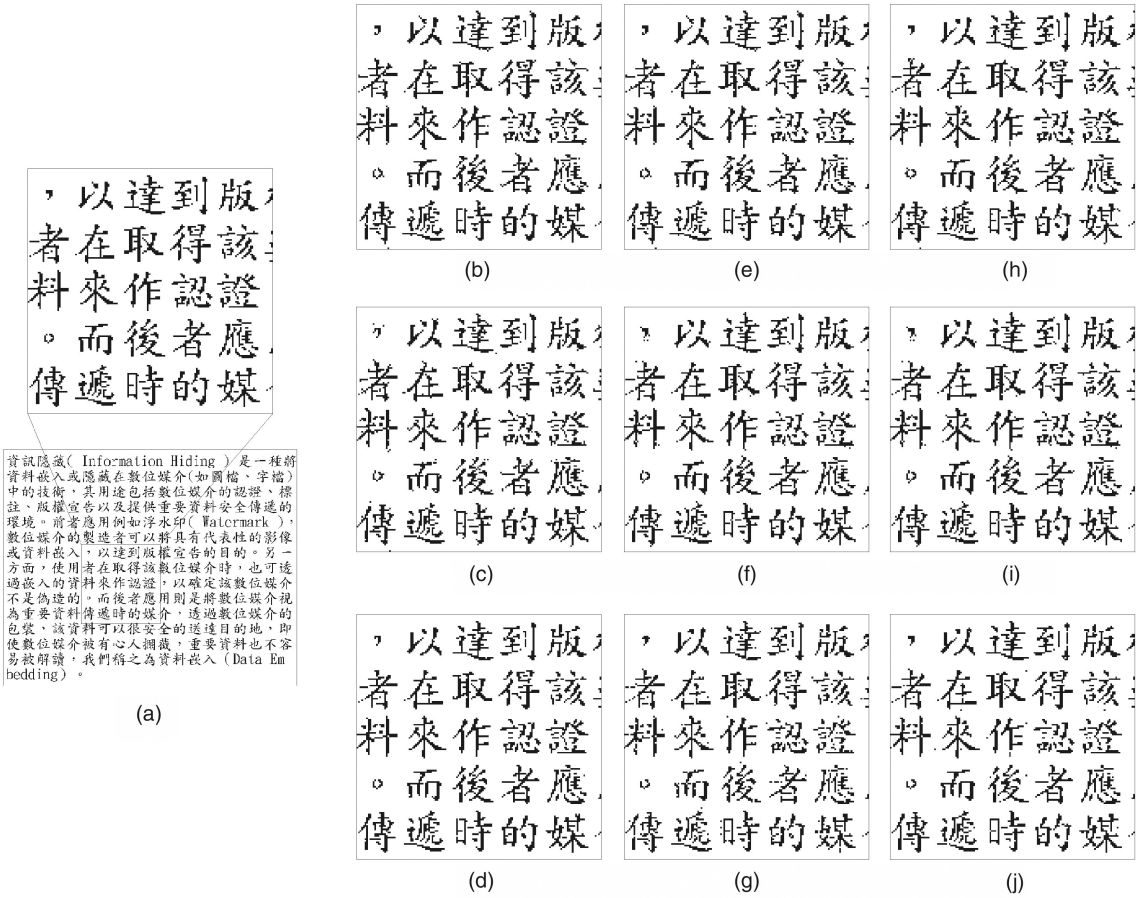

Fig. 4. Hiding effect on a Chinese document: (a) the original host image, (b) hiding 889 bytes by our new scheme with block size 12×12, (c) hiding 1,291 bytes by CPT scheme with block size 12×12, (d) hiding 190 bytes by WL scheme with block size 12×12, (e) hiding 933 bytes by our new scheme with block size 12×12, (f) hiding 933 bytes by CPT scheme with block size 16×16, (g) hiding 933 bytes by WL scheme with block size 4×4, (h) hiding 682 bytes by our new scheme with block size 16×16,

To be shown later, this definition will improve the quality of the image after data hiding. For example, the following shows three possible ways to define a 6×6 weight matrix:

$$\begin{bmatrix} o & e & o & e & o & e \\ e & o & e & o & e & o \\ o & e & o & e & o & e \\ e & o & e & o & e & o \\ o & e & o & e & o & e \\ e & o & e & o & e & o \end{bmatrix} \begin{bmatrix} o & o & o & o & o & o \\ e & e & e & e & e & e \\ o & o & o & o & o & o \\ e & e & e & e & e & e \\ o & o & o & o & o & o \\ e & e & e & e & e & e \end{bmatrix} \begin{bmatrix} e & o & e & o & e & o \\ o & e & o & e & o & e \\ e & o & e & o & e & o \\ o & e & o & e & o & e \\ e & o & e & o & e & o \\ o & e & o & e & o & e \end{bmatrix}, \quad (1)$$

where "o" means an odd number and "e" an even number. For instance, the number of legal weight matrices based on any of these patterns is

$$\left[C_{2^{r-1}}^{mn/2} * (2^{r-1})! * (2^{r-1})^{(mn/2)-(2^{r-1})}\right] * \left[C_{2^{r-1}-1}^{mn/2} * (2^{r-1}-1)! * (2^{r-1}-1)^{(mn/2)-(2^{r-1})+1}\right],$$

where the first part is for choosing those odd numbers and the second part for those even numbers.

The detailed data hiding steps are as follows:

1. If F_i is completely black or blank, simply keep F_i intact (not hidden with data) and skip the following steps. Otherwise, perform the following:
2. Compute $SUM((F_i \oplus K) \otimes W)$.
3. From the matrix $F_i \oplus K$, compute for each $w = 1..2^{r+1} - 1$ the following set:

$$S_w' = \Big\{(j,k) \mid \Big[([W]_{j,k} = w) \wedge ([F_i \oplus K]_{j,k} = 0) \wedge ([dist(F)]_{j',k'} \le \sqrt{2})\Big] \vee \Big[([W]_{j,k} = 2^{r+1} - w) \wedge ([F_i \oplus K]_{j,k} = 1) \wedge ([dist(F)]_{j',k'} \le \sqrt{2})\Big]\Big\},$$

where $[dist(F)]_{j',k'}$ is for the bit corresponding to $[F_i]_{j,k}$ in block F_i (note that F_i is only a block in the original image F). Also, define $S_w' = S_{w'}'$ for any $w \equiv w' \pmod{2^{r+1}}$.

Intuitively, in the above step, we try to control the quality of the image after modification. A bit can be swapped only if it has a neighbor which shares the same value as its new value. Here we

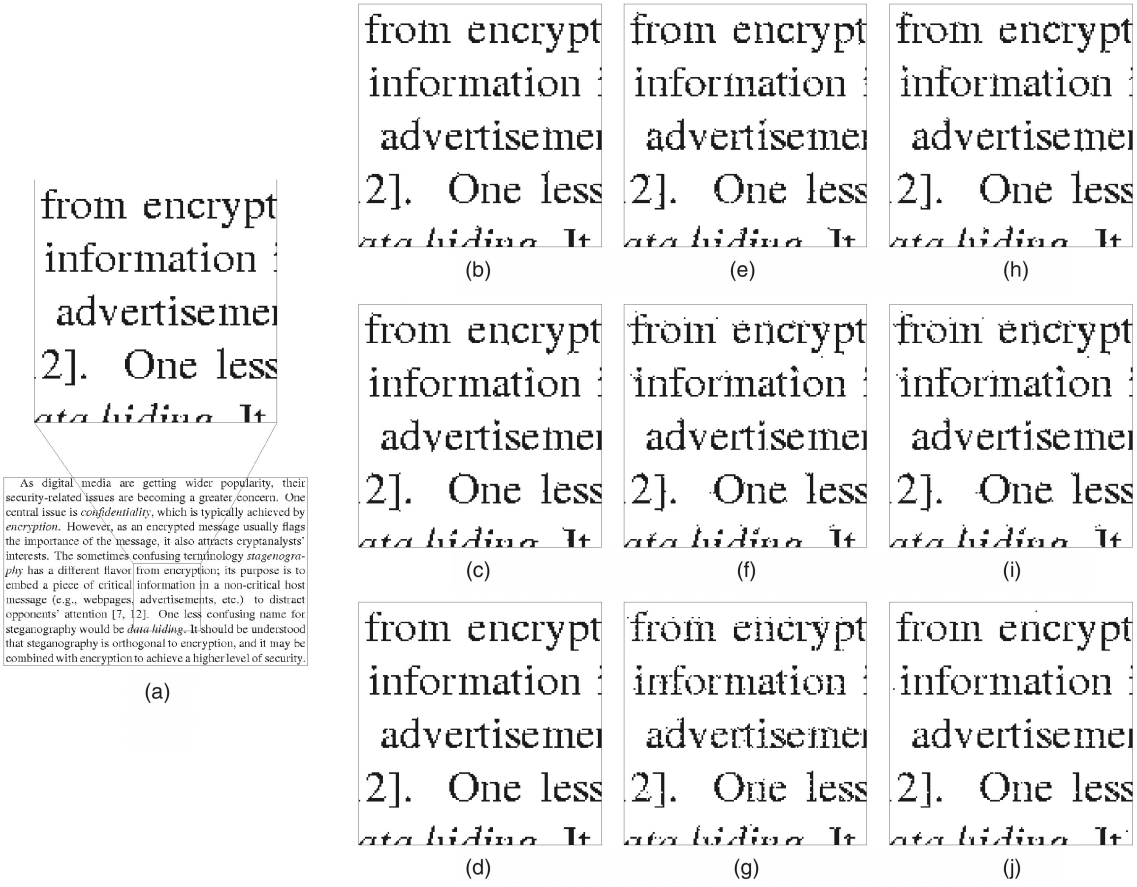

Fig. 5. Hiding effect on a English document: (a) the original host image, (b) hiding 203 bytes by our new scheme with block size 32×32, (c) hiding 340 bytes by CPT scheme with block size 32×32, (d) hiding 34 bytes by WL scheme with block size 32×32, (e) hiding 906 bytes by our new scheme with block size 12×12, (f) hiding 906 bytes by CPT scheme with block size 16×16, (g) hiding 906 bytes by WL scheme with block size 4×4, (h) hiding 568 bytes by our new scheme with block size 16×16, (i) hiding 937 bytes by CPT scheme with block size 16×16, and (j) hiding 348 bytes by WL scheme with block size 8×8.

define a neighbor to be one at a distance $\leq \sqrt{2}$, i.e., a bit has eight neighbors.

4. Define a weight difference

$$d' \equiv (b_1 b_2 \dots b_r 0) - SUM((F_i \oplus K) \otimes W) (\mathrm{mod}\ 2^{r+1}).$$

If $d' = 0$, there is no need to change F_i. Otherwise, we run the following program to transform F_i to F_i'.

if (there exists an h such that $S'_{hd'} \neq \emptyset$ and $S'_{-(h-1)d'} \neq \emptyset$) **then**
  Randomly pick an h which satisfies the above condition;
  Randomly pick a $(j,k) \in S'_{hd'}$ and complement the bit $[F_i]_{j,k}$;
  Randomly pick a $(j,k) \in S'_{-(h-1)d'}$ and complement the bit $[F_i]_{j,k}$;
else/ * no data will be hidden * /
  if $(SUM((F_i \oplus K) \otimes W) \bmod 2 = 1)$ **then**
    Keep F_i intact;
  else
    Select a (j,k) such that $[W]_{j,k}$ is odd and its corresponding $[dist(F)]_{j',k'}$ is the smallest;
    Complement the bit $[F_i]_{j,k}$;
  end if;
end if;

However, note that, if the resultant F_i' is completely black or blank, we will regard the data hiding as invalid. That is, F_i' will be regarded as not hidden with data and we will try to hide the same bit stream $b_1 b_2 \dots b_r$ again in the next host block. (See the discussion below.)

5. On receiving the block F_i', the receiver computes the hidden data to be $\frac{SUM((F_i' \oplus K) \otimes W)}{2}$, if F_i' is not completely black or blank and $SUM((F_i' \oplus K) \otimes W)$ is even. Otherwise, F_i' contains no hidden information.

Note that, in the above steps, we still regard S'_0 as nonempty. However, this set still represents those elements by changing which we can increase the total weight by 0. Since we can do this by modifying nothing on the host image block, we purposely use this convention to simplify our presentation.

There are two important properties maintained by this scheme. First, we guarantee that each bit being modified must be neighboring to a bit that is equal to its new value. Thus, such a modification will be quite invisible. This can be proven by observing the **if** statement in Step 4. If the condition holds true, then the definition of S'_w in Step 3 already ensures this property. Otherwise, in the **else** case, either F_i will be kept intact or a randomly picked bit whose corresponding element in W is odd and whose corresponding value in $dist[F]$ is the smallest will be swapped. Since F_i is not completely black or blank, there must exist two bits in F_i which are next to each other and whose values

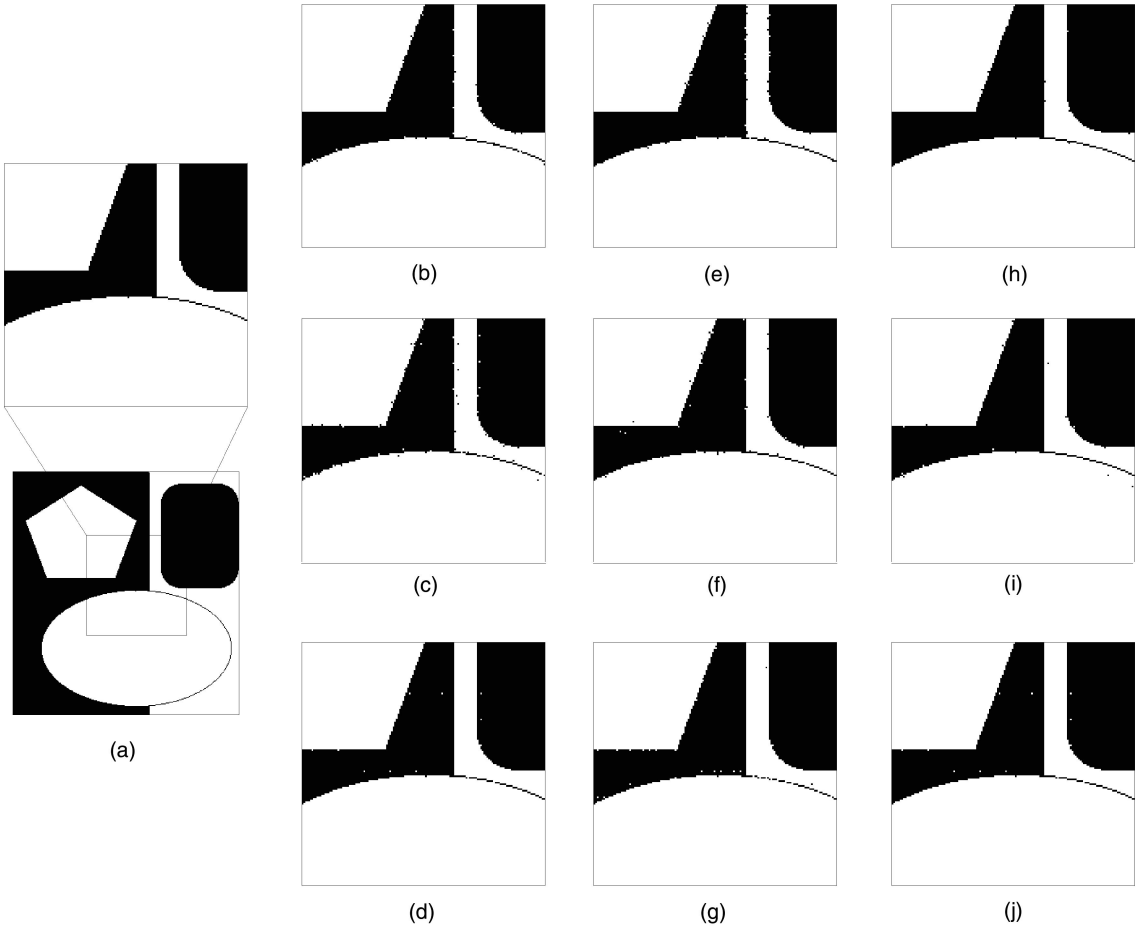

Fig. 6. Hiding effect on geometric shapes: (a) the original host image, (b) hiding 35 bytes by our new scheme with block size 16×16, (c) hiding 97 bytes by CPT scheme with block size 16×16, (d) hiding 16 bytes by WL scheme with block size 16×16, (e) hiding 62 bytes by our new scheme with block size 12×12, (f) hiding 62 bytes by CPT scheme with block size 24×24, (g) hiding 62 bytes by WL scheme with block size 4×4, (h) hiding 22 bytes by our new scheme with block size 32×32, (i) hiding 46 bytes by CPT scheme with block size 32×32, and (j) hiding 16 bytes by WL scheme with block size 16×16.

are complement to each other. It is not hard to prove that, for any 2×2 sub-block that covers the above two bits, its corresponding 2×2 sub-block in $dist(F)$ must have distances no larger than $\sqrt{2}$. According to Definition 1, the 2×2 sub-block in W corresponding to this 2×2 block must contain an odd entry. Thus, the bit that is complemented must be adjacent to a bit that is equal to its new value.

Second, to take the image quality after modification into consideration, a completely black or blank block is not used to hide data. However, in Step 4, a block, after modification, may become completely black or blank. If so, to avoid confusion, we will simply leave it completely black or blank. However, we will regard this block as containing no data because, otherwise, this will be indistinguishable from the case that the given host block is completely black or blank.

Finally, we analyze the data hiding ratio in terms of the number of bits being modified in the original host image. Specifically, we define the following metric:

$$\frac{\text{number of bits being hidden in the host image}}{\text{number of bits being modified in the host image.}}$$

For the CPT scheme, at most 2 bits in an $m \times n$ block may be modified to hide $\lfloor \log_2(mn+1) \rfloor$ bits of secret data, so the ratio is

$$\frac{\lfloor \log_2(mn+1) \rfloor}{2}.$$

For our new scheme, at most 2 bits in an $m \times n$ block may be modified to hide $\lfloor \log_2(mn+1) \rfloor - 1$ bits of data, or at most 1 bit may be modified to represent that there is no hidden data, so the ratio is

$$\frac{p(\lfloor \log_2(mn+1) \rfloor - 1)}{p * 2 + (1-p) * 1},$$

where p is the probability that a block is embedded with secret data. Note that p is image-dependent, which will be evaluated through experiments in the next section.

4 Experimental Results

We have implemented our scheme and conducted some tests. In the implementation, the parameters $m, n,$ and r are not sent to the receiver as a separate message, but, instead, transmitted along with the image as image-dependent parameters. Specifically, we hide $m, n,$ and r in the image together with the critical information B using our hiding scheme. We allocated 6, 6, and 4 bits for m, n, and r, respectively. These 16 bits were each hidden in a host image block of size 1×200. Thus, the first 16×200 bits of the image were used for hiding m, n, and r, and the rest for hiding B. This is illustrated in Fig. 3.