Jigsaw puzzle images for steganography

En-Jung Farn
Chaur-Chin Chen
National Tsing Hua University
Department of Computer Science
101 Section 2 Kwan-Fu Road
Hsinchu, 30013
Taiwan
E-mail: cchen@cs.nthu.edu.tw

Abstract. The jigsaw puzzle has been popular from past to present, and there are many jigsaw puzzle images on the Internet. This paper proposes a novel method to hide secret data in jigsaw puzzle images. First, a digital image is taken as input and divided into blocks. Then, a semicircle is drawn and attached to the right and bottom sides of each block. The secret data are embedded through the attached positions and orientations of the semicircles according to a stegokey. The resulting image looks like those jigsaw puzzle images appearing on many jigsaw puzzle Web sites. Experiments show that the proposed method is undetectable under the passive warden and robust to format conversion, lossy compression, and lossy recompression. [DOI: 10.1117/1.3159872]

Subject terms: jigsaw puzzle image; steganography; stegokey; undetectable.

Paper 080986RR received Dec. 19, 2008; revised manuscript received May 14, 2009; accepted for publication May 19, 2009; published online Jul. 28, 2009.

1 Introduction

Steganography[1,2] is the art of concealing the existence of a message within seemingly innocuous carriers. A steganographic message is often embedded in a carrier, called a cover-carrier, and results in a stego-carrier. The main requirement of steganography is undetectability,[3] the hidden message should not be detected by a passive warden under all possible visual and statistical attacks.[3–6] Even if the warden is not able to find the secret message with statistical attacks, he still may intentionally distort the content. Thus, a limited robustness is preferred; that is, the hidden message can still be extracted successfully, even if the stego-carrier is modified by some operators, such as format conversion, lossy compression, and lossy recompression. On the other hand, perceptibility is permitted because the cover-carrier has no intrinsic value.[3]

In steganography, the most popular carrier is the image. There are many steganographic methods[1–3,7–22] proposed for various kinds of image formats. Gray-scale image domain, palette based and JPEG are the three most commonly used formats. The methods based on gray-scale image domain[7,8] typically insert secret messages into the least significant bits (LSB) of pixels. The LSB insertion is vulnerable to lossy compression. For palette-based images, directly embedding messages in those indices will cause radical color change. Many efforts[9,10] try to reduce the distortion created in the embedding process. These palette-based methods are vulnerable to format conversion. The methods based on JPEG images[11–14] typically hide the secret message in discrete cosine transform (DCT) coefficients. Upham's JSteg[11] sequentially replaces the least significant bits of DCT coefficients with the message's data. The χ^2-test[4] successfully detects the steganographic system. F5 proposed, by Westfeld,[13] uses decrements of the DCT coefficients' absolute values against the χ^2-test. However, Fridrich et al.[5] and her group presented a steganalytic method that does detect images with F5 content. Sallee[14] presented the model-based steganography approach for JPEG images, which is a general framework for constructing steganographic systems that preserve a chosen model for the cover image. Note that all JPEG-based methods are vulnerable to lossy recompression and format conversion[3].

Reversible steganography[15,16] has recently drawn considerable attention from many scholars. However, this kind of method is detectable and vulnerable to lossy compression. The above-mentioned steganographic methods share the placement of embedding changes (the selection rule[3]) between the sender and the receiver. Recently, several approaches[3,17–19] with nonshared selection rules have been proposed; the receiver does not need to know where data are embedded. These approaches provide improved steganographic security and are less vulnerable to steganalytic attacks compared to existing method with shared selection channels. However, this kind of method is also vulnerable to lossy compression.

As we know, the more kinds of images a steganographic system can use to embed data, the more secure the system is. In this paper, we will propose a method to embed secret data in a new type of image called the jigsaw puzzle image. Before describing the proposed method, we will first give a brief description of jigsaw puzzles and jigsaw puzzle images.

A jigsaw puzzle is a puzzle that requires the assembly of numerous small and often oddly shaped pieces. Jigsaw puzzles are originally created by painting a picture on a flat, rectangular piece of wood, and then cutting that picture into small pieces with a jigsaw. Most modern jigsaw puzzles are made out of cardboard, and an enlarged photograph or printed reproduction of a painting or other two-dimensional artwork is glued onto the cardboard before cutting.

Recently, many Web sites[23–27] sell traditional jigsaw puzzles or online jigsaw puzzle games. They also allow users to upload their own pictures to create jigsaw puzzles. JigZone[23] is such a website. If a user wants to make his own jigsaw puzzle, he can select an image as input and decide the number and type of pieces to cut, then the cor-

0091-3286/2009/$25.00

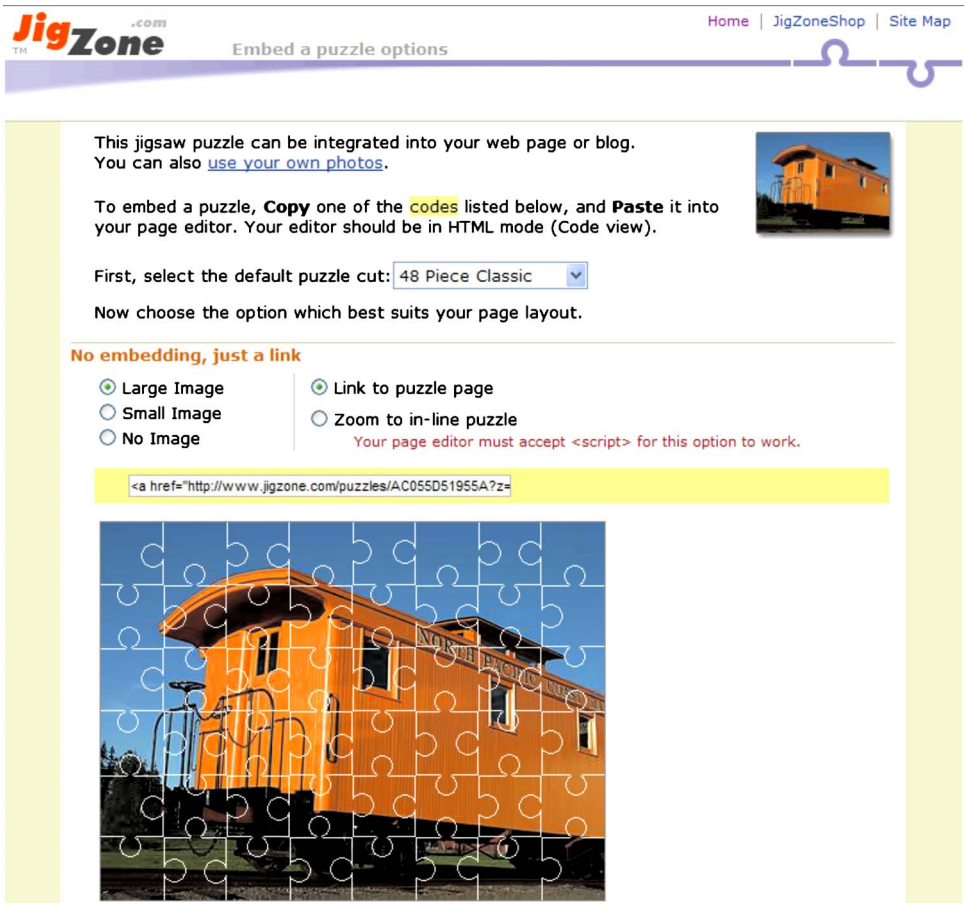

Fig. 1 One jigsaw puzzle image from JigZone.[23]

responding jigsaw puzzle image (see Fig. 1), which will be further explained in Sec. 2.1, will be shown on the Web site for preview. Spintop-Games[24] sells 17 different kinds of online jigsaw puzzle games. Each game provides a function to create different kinds of jigsaw puzzle images (see Fig. 2). From our observation, we find that puzzle lines and patterns on most jigsaw puzzle images are drawn in white or black. On the other hand, some Web sites teach users to make a jigsaw puzzle.[28–32] All these websites tell users how to create jigsaw puzzle images for cutting purposes. Bajaj[31] provides a tutorial to show how to create a full-slide puzzle effect in PowerPoint. The fact that jigsaw puzzle images are often seen on the Internet intrigues us to propose a method to hide data in a jigsaw puzzle image.

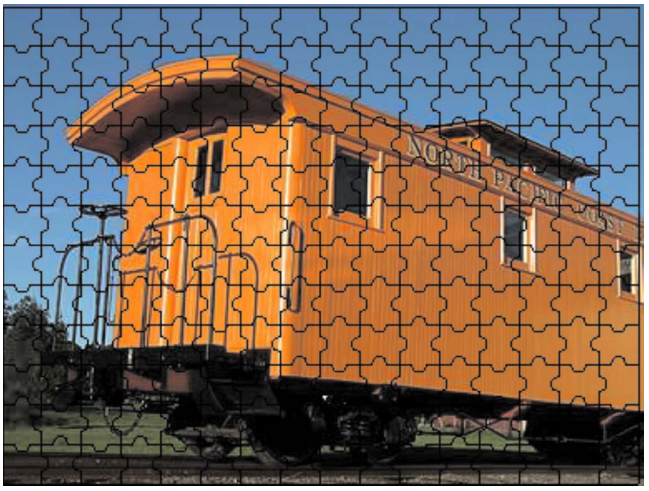

Fig. 2 One jigsaw puzzle image from Spintop-Games.[24]

In the proposed method, first, an image is divided into blocks. Second, the right and bottom sides of each block (except the boundary) will be attached a semicircle. The secret message is embedded through the attached positions and the orientations of the semicircles, which depend on the secret message and a random number generator with a stegokey as its seed. The resulted image looks like an ordinary jigsaw puzzle image. Experiments show that the proposed method is undetectable and robust to format conversion and lossy compression/recompression.

The rest of the paper is organized as follows. Section 2 describes the proposed method. In Sec. 3, we analyze the limited robustness and security/undetectability of the proposed method. The conclusion is made in Sec. 4.

2 Proposed Method

Before describing the proposed method, we first introduce what a jigsaw puzzle image is and how to embed a secret message in a jigsaw puzzle image.

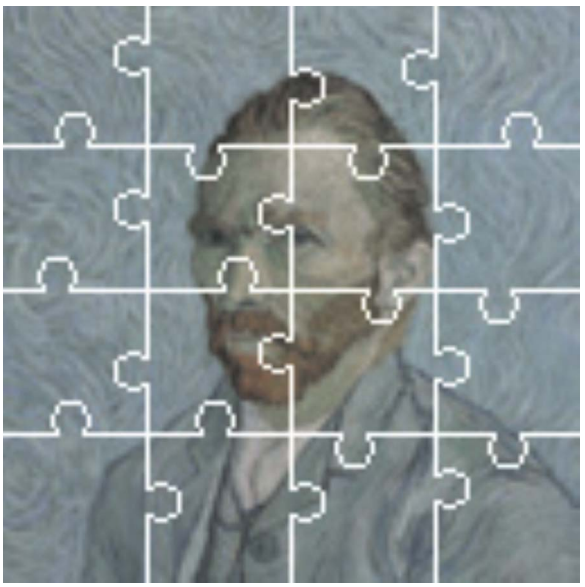

Fig. 3 Jigsaw puzzle image.

2.1 *Embedding a Secret Message in a Jigsaw Puzzle Image*

A jigsaw puzzle image (see Fig. 3) is an image formed by dividing a meaningful image into several blocks with special patterns. Block patterns[23] can be designed in different ways. This paper focuses on a special kind of design such that each side of a square block will have a semicircle attached at a certain point of the side. There are four types of sides corresponding to four different orientations of the attached semicircles, which are right-, left-, top- and bottom-connected sides. Figure 4 illustrates some examples.

The different types of sides and the attached point of a semicircle can represent different information. On the basis of this fact, we will use the right and bottom sides of each block to embed several bits. Note that the right (bottom) sides of blocks in the right (bottom) boundary of an image will not be used for data embedding. And the side type will be used to embed one bit. For the convenience of explanation, in this paper, we will make the rule that the right-connected (or the top-connected) side is assigned bit 0 and the left-connected (or the bottom-connected) side is assigned bit 1.

In the following, we will describe how to draw an attached semicircle according to the secret message. Suppose that a block has size $n \times n$. Let the radius of the semicircle be r. Here, we suggest that $r \geq 3$; otherwise, the semicircle will be too small to cut smoothly. Let the positions of pixels on a block side be indexed from 0 to $n-1$, and let the semicircle be attached at position P with $P \in [3r, n-3r]$, this restriction is used to avoid two neighboring attached semicircles overlapping. Figure 5 shows an overlapping example. Thus, we have $(n-6r)+1$ different positions, which can be used to embed data; the orientation of the semicircle also can be used to embed one bit. This means that the embedding capacity of each block side is at least $\lfloor \log_2(n-6r) \rfloor + 1$ bits. If the total embedding capacity of an image is larger than that needed by the secret message, then we do not need to embed data in each block side. We can utilize this point to raise the security level by classifying all sides into two classes. One is the dummy side (DS) without data embedded; the other is the information side (IS) with data embedded. Let $k = \lfloor \log_2(n-6r) \rfloor$, s be the secret data of k bits, and $t \in [0, n-6r)$ be obtained by a random number generator G, then we can calculate the attached position P as follows:

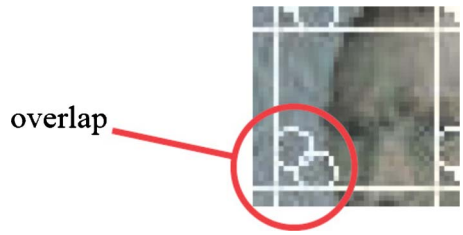

Fig. 5 Two overlapping attached semicircles.

$$s' = (t+s) \bmod (n-6r), \tag{1}$$

$$P = \begin{cases} s' + 3r & \text{if the block side is an IS and } s' < t \\ s' + 3r + 1 & \text{if the block side is an IS and } s' \geq t \\ t + 3r & \text{if the block side is a DS.} \end{cases} \tag{2}$$

Note that the random number t is used to prevent a passive warden from statistical attack; this point will be explained later. On the other hand, in Eq. (2), we design that for a dummy side, $P-3r=t$, and for an information side, $P-3r \neq t$. This situation will be used in the extraction process to judge if a block side is a DS or IS.

Before drawing an attached semicircle for the bottom side (BS) of a block B, we will first calculate the central point (C_x, C_y) of the attached semicircle according to the embedded data. As described previously, a side can embed $k+1$ bits. The right-most bit (RMB) is embedded through the side type; the remaining k bits with value s are embed-

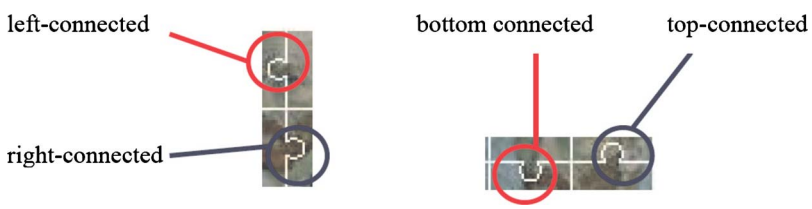

Fig. 4 Four types of sides.

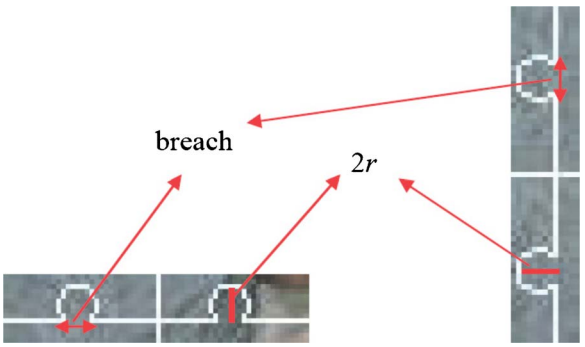

Fig. 6 Breach and the distance from breach to semicircle.

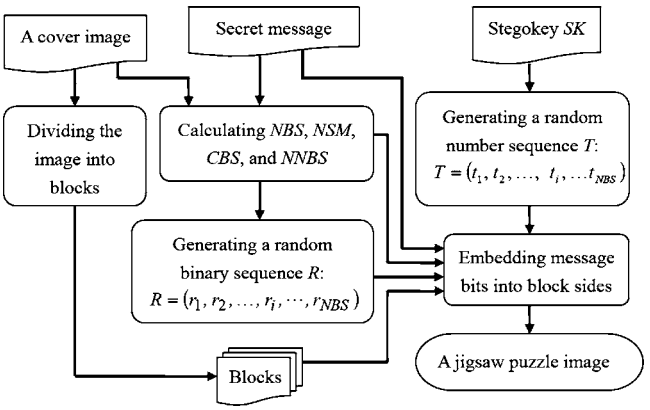

Fig. 7 Block diagram of the proposed embedding process.

ded through the attached position of the drawn semicircle. According to this rule, (C_x, C_y) can be evaluated as follows:

$$\mathrm{RMB}' = \mathrm{RMB} \oplus \mathrm{LSB}(t), \tag{3}$$

$$C_x = P, \tag{4}$$

$$C_y = \begin{cases} n-r & \text{if BS is an IS and RMB}' = 0 \\ n-2+r & \text{if BS is an IS and RMB}' = 1 \\ n-r & \text{if BS is a DS and LSB}(t) = 0 \\ n-2+r & \text{if BS is a DS and LSB}(t) = 1, \end{cases} \tag{5}$$

where P is evaluated by Eq. (2) and LSB (t) is the least significant bit of t. With (C_x, C_y), we can draw the attached semicircle. Note that $\mathrm{RMB}'=0(1)$, a top(bottom)-connected semicircle is drawn.

Similarly, for the right side (RS) of B, we can calculate the central point (C_x, C_y) of the attached semi-circle according to the embedded data as follows:

$$\mathrm{RMB}' = \mathrm{RMB} \oplus \mathrm{LSB}(t), \tag{6}$$

$$C_x = \begin{cases} n-2+r & \text{if RS is an IS and RMB}' = 0 \\ n-r & \text{if RS is an IS and RMB}' = 1 \\ n-2+r & \text{if RS is a DS and LSB}(t) = 0 \\ n-r & \text{if RS is a DS and LSB}(t) = 1, \end{cases} \tag{7}$$

$$C_y = P. \tag{8}$$

With (C_x, C_y), we can draw the attached semicircle. Note that $\mathrm{RMB}'=0(1)$, a right (left)-connected semicircle is drawn.

Figure 6 shows an example of drawing semicircles on bottom/right sides. Here, we define each line segment, which is on a bottom/right side and not a part of the drawn puzzle line, as breach (see Fig. 6). Note that the distance from the central point of the breach to the top of the semicircle is equal to $2r-1$; this fact will be used to extract the radius automatically in the extraction process.

2.2 *Proposed Embedding Process*

On the basis of aforementioned idea, we summarize a secret message embedding process, as shown in Fig. 7.

Note that a secret message is represented in a binary format. According to the message length, an $N \times N$ image is first chosen as a cover image with its size large enough to embed the message. Then the cover image is divided into blocks of $n \times n$. As mentioned previously, many Web sites[23–29] provide jigsaw puzzle images for preview, and most puzzle patterns[23,24] are drawn in white or black In this paper, we use white (255, 255, 255) as an example to draw the used block side lines and attached semicircles. Let the number of block sides (NBS), which can be used to attach a semicircle, be $NBS=2(N/n-1)(N/n-1)+2(N/n-1)$. Let NSM be the number of bits of the secret message (NSM) and let $\mathrm{CBS}=\lfloor \log_2(n-6r) \rfloor+1$ be the embedding capacity (bits) of each side (CBS), then the number of block sides (NNBS) needed to embed data, can be computed by $\mathrm{NNBS}=\mathrm{NSM}/\mathrm{CBS}$. As mentioned previously, if $\mathrm{NBS} > \mathrm{NNBS}$, then some block sides will not be used to embed data. To implement this point, we first create a binary sequence of NBS bits with the leftmost NNBS bits being 1 and the others being 0. Then the binary sequence is randomly permutated to form a random binary sequence $R=(r_1, r_2, \ldots, r_i, \ldots, r_{\mathrm{NBS}})$. The stegokey (SK) is only known by the sender and receiver and used as the seed of the random number generator G to generate a random number sequence $T=(t_1, t_2, \ldots, t_i, \ldots, t_{\mathrm{NBS}})$ with $t_i \in [0, n-6r)$.

After dividing the cover image into blocks, we process each block from left to right and bottom to top. For each block, we draw a left- (right-) or top- (bottom)-connected semicircle attached to the right or bottom side at a certain position according to the message bits embedded, the random binary sequence R and the random number sequence T. The bits in R and the numbers in T are sequentially taken while undergoing the process; random bit r_i and random number t_i correspond to the i'th side. If bit r_i is 1, the i'th side is considered as an information side, then CBS embedding data bits are taken; a semicircle is drawn and attached to the side at the corresponding position, which is evaluated by Eq. (2) based on the embedding data bits and the random number t_i. If bit r_i is 0, the i'th side is considered as a DS; no data bits will be embedded, but a semicircle is still drawn based on the random number t_i. Note that an image

Fig. 8 Example for a jigsaw puzzle image with data embedded: (a) The original cover image, (b) jigsaw puzzle image with "jsawpuzzle" embedded, and (c) compressed version of (b).

is usually stored in compressed type; thus after the embedding process is completed, the created jigsaw puzzle image will be further compressed.

Figure 8(a) shows a 128×128 cover image. Figure 8(b) shows the result of embedding the secret message "jsawpuzzle" in Fig. 8(a) with $n=32$, $r=4$. Figure 8(c) shows the compressed version of Fig. 8(b) using Photoshop CS4 JPEG quality level 6.

2.3 *Extraction Process*

Here, we will provide a method to extract the secret message from the compressed jigsaw puzzle image. Figure 9 shows the block diagram of the proposed extraction process. The extraction process contains two parts. The first part is to locate the right (bottom) side of each block and evaluate the radius of the drawn semicircle. The second part is to determine the type of side and the attached position of the drawn semicircle.

In the first part, we first transfer the compressed jigsaw puzzle image into a gray-scale one. Second, for each row (column), count the percentage of white pixels appearing in each row (column). If the percentage is $>60\%$, then consider it as a jigsaw puzzle line. Note that due to the compression distortion, each pixel with a gray value of >225 is considered as a white pixel.

On the basis of these located jigsaw puzzle lines, we can obtain the block size and locate the right (bottom) side of each block in the image. Then, we will find the radius of the semicircle. First, for each bottom (right) side, we locate the breach on the side and calculate its length. After obtaining the lengths of breaches on all bottom (right) sides, we set the length, which appears most frequently among all lengths of breaches, as breach length (BRL). For each breach on a bottom (right) side with length equal to BRL, we calculate two distances, one from the central point of the breach to the first white pixel in the upward (rightward) direction, the other in the downward (leftward) direction. We choose the smaller one of the two distances and record it. After all breaches are processed, we take the distance, which occurs most frequently among all recorded distances, and set the radius of the semicircle, r, to be the half of the distance.

In the second part, based on the obtained r, we can determine the side type and the attached position of the drawn semicircle at each block side. First, for each bottom side and each point $(C_x,n-1)$ at the side with $C_x\in[3r,n-3r]$, we locate two central points UC $(C_x,n-r)$ and DC $(C_x,n-2+r)$. Next, we generate a pseudo top-connected semicircle with UC as its center and a pseudo bottom-connected semicircle with DC as its center, respectively. Then, for each generated pseudo semicircle, we calculate the sum of the gray values of all pixels on the pseudo semicircle. Note that if a semicircle actually exists, then each pixel on the semicircle should have gray value near 255 and the sum of the gray values of all pixels on the semicircle should be maximum. On the basis of this fact, we can find the actual semicircle and determine the side type and accurate attached position. According to the extracted attached posi-

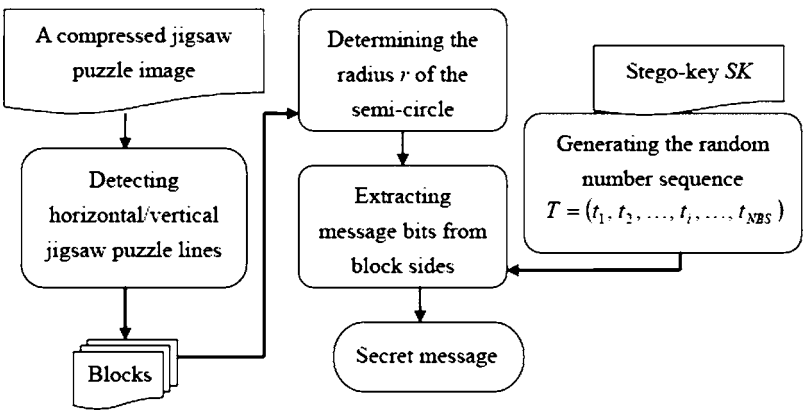

Fig. 9 Block diagram of the proposed extraction process.

tion, we can judge if the block side has data embedded. If yes, we can further extract the secret data. The details are depicted in Procedure 1.

Procedure 1. Data extraction for an embedded side.

Let P be the extracted attached position
* *Judge if any data is embedded*
$P=P-3r$
if $P=t$
the side is a DS, no data are embedded
else
the side is an IS
if $P>t$
$s=P-1-t$
else
$s=P-t+n-6r$
Extract the embedded secret data s
if the side type is top-connected or right-connected
$RMB'=0$
else
$RMB'=1$
$RMB=RMB' \oplus \mathrm{LSB}(t)$
$s^*=2s+RMB$
s is the embedded* $(k+1)$ *bit data.*

Note that t used in procedure 1 is the random number from the sequence T, which is the same as that used in the embedding process. Procedure 1 determines that the current side is DS or IS. If it is an IS, the value of the embedded $(k+1)$ bits, $s*$, will be extracted.

By the similar way, for each right side, we can first locate the attached position of the drawn semicircle. And procedure 1 is then applied to extract the embedded $(k+1)$ bits. Note that the possible attached position at the right side is $(n-1, C_y)$ with $C_y \in [3r, n-3r]$, and the corresponding two possible semicircle centers are $(n-r, C_y)$ and $(n-2+r, C_y)$.

Figure 10 shows the located semicircles by applying the proposed extraction method on Fig. 8(c). We can see that all semi-circles and their attached positions are correctly detected.

3 Analysis of the Proposed Method

Here, we will analyze the proposed method. The two requirements of steganography: limited robustness and security/undetectability will first be discussed, and then the embedding capacity will be addressed.

3.1 *Limited Robustness*

As mentioned previously, most existing steganographic methods cannot resist compression/recompression and format conversion.[3] However, the proposed method is robust under image compression/recompression and format conversion, including JPEG to GIF and GIF to JPEG. The main reason is that the secret messages are embedded through the attached positions and the shape of the semicircles, which will not be changed by compression/recompression and format conversion. To show this point,

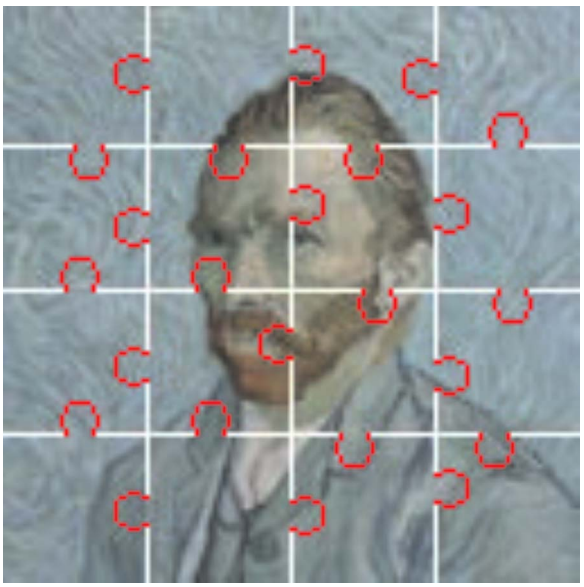

Fig. 10 Result of applying the extraction process on Fig. 8(c) with the extracted semicircles marked in red. (Color online only.)

we take nine 480×352 images shown in Fig. 11 to do the experiment; some images are from the JigZone Web site.[23] For each cover image, 10 different jigsaw puzzle images with block size 32×32 are created, half of the 10 jigsaw puzzle images use white to draw semicircles and block lines, and the other half use black. One of these 10 jigsaw puzzle images is shown in Fig. 12. For each created jigsaw puzzle image, we use Adobe Photoshop CS4 to store the image in JPEG format with a quality level of medium (6) and high (8) and in GIF format. Each JPEG image with a quality level (8) is recompressed into JPEG image with quality level (6); each JPEG image with a quality level (8) is converted into GIF format; and each GIF image is con-

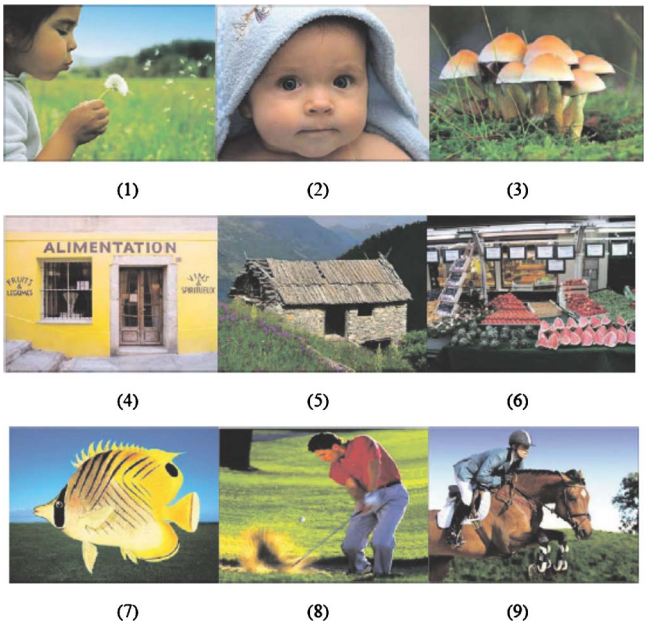

Fig. 11 Nine test images.

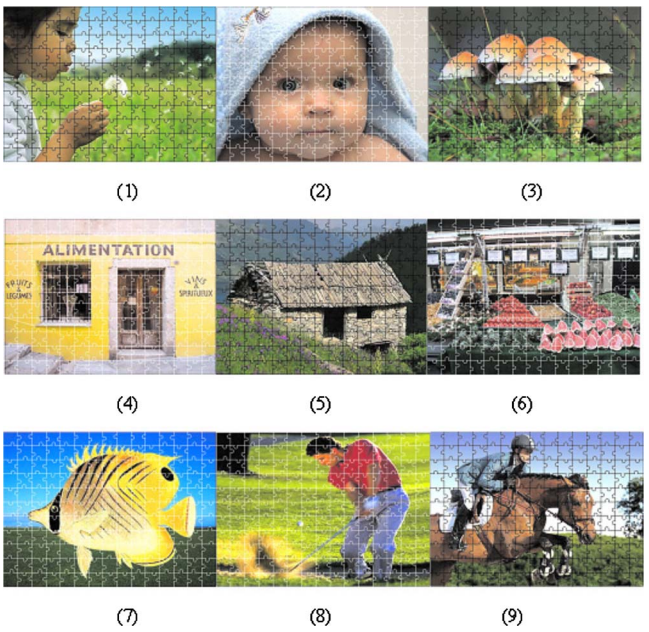

Fig. 12 Nine created jigsaw puzzle images.

verted into JPEG image with quality level (6). Thus, each jigsaw puzzle image is converted into six different compression versions. Hence, we totally have 540 compressed jigsaw puzzle images. Tables 1–6 show the average PSNR[3] and the correct detection rate of applying the proposed method to these jigsaw puzzle images. Note that the correct detection rate is evaluated to be the ratio of the number of sides with the attached points located correctly to the total number of sides used for data embedding. From Tables 1–6, we can see that the proposed method can correctly extract embedding secret data for all jigsaw puzzle images created from image 1 to image 6. For those jigsaw puzzle images created from images 7 and 8 using white to draw the semicircles and block lines, all embedded secret data can be correctly extracted. For those jigsaw puzzle images created from image 9 using black to draw the semicircles and block lines, all embedded secret data can be successfully extracted.

From Tables 1–6, we find that for few jigsaw puzzle images using white (black) to draw the semicircles and block lines, the secret data may not be correctly extracted, but using black (white), the secret data can be extracted successfully. The reason is that for an image with a white (black) area [see Fig. 13(a)], if an attached semicircle drawn by white (black) is in a darker (lighter) area near the white (black) area, then after compressing, the gray values of the pixels in the semicircle will be lowered down (raised up), but the gray values in the white (black) area still keep their values near 255 (0). This situation will result is geting a wrong attached position in the white (black) area (see Fig. 13). Fortunately, we have two ways to overcome this problem. The first is to avoid using this kind of image as the cover image; this is allowed because steganographer is free to choose a particular cover image.[3] The second is to use black (white) instead of the white (black) to draw semicircles and puzzle lines, Tables 1–6 show that this will always get correct results. Figure 13(a) shows four error detections using white to draw puzzle patterns, and Fig. 13(b) shows the successful extraction result using black to draw puzzle patterns. Here, we provide a mechanism to automatically implement these two ways. According to the idea proposed by Kharrazi et al.,[29] the sender can think ahead and attempt to guess what kind of operator the warden is going to use to do some modification and embed the same message into many different covers, run the worst-known modification on each stego-image, and then simply send the one that passes the modification. The proposed mechanism uses a similar idea to decide which color is suitable to be used to draw the semicircles and block lines. First, the

Table 1 Compression mode: JPEG quality level medium (6).

	Jigsaw puzzle pattern drawn by											
	White color						Black color					
	Correct detection rate (%)					PSNR	Correct detection rate (%)					PSNR
Image1	100	100	100	100	100	26.7	100	100	100	100	100	29.0
Image2	100	100	100	100	100	31.4	100	100	100	100	100	31.9
Image3	100	100	100	100	100	29.2	100	100	100	100	100	31.1
Image4	100	100	100	100	100	28.2	100	100	100	100	100	30.2
Image5	100	100	100	100	100	30.3	100	100	100	100	100	31.3
Image6	100	100	100	100	100	29.6	100	100	100	100	100	29.7
Image7	100	100	100	100	100	25.4	99.4	99.4	99.7	99	100	25.9
Image8	100	100	100	100	100	26.5	99.4	99.7	99.7	100	99.4	26.9
Image9	98.7	99.4	99.4	99.4	98.4	30.6	100	100	100	100	100	30.9

Table 2 Compression mode: JPEG quality level high (8).

	Jigsaw puzzle pattern drawn by											
	White color						Black color					
	Correct detection rate (%)					PSNR	Correct detection rate (%)					PSNR
Image1	100	100	100	100	100	32.1	100	100	100	100	100	33.4
Image2	100	100	100	100	100	34.2	100	100	100	100	100	36.0
Image3	100	100	100	100	100	33.2	100	100	100	100	100	34.0
Image4	100	100	100	100	100	33.4	100	100	100	100	100	33.8
Image5	100	100	100	100	100	33.4	100	100	100	100	100	34.0
Image6	100	100	100	100	100	32.5	100	100	100	100	100	32.7
Image7	100	100	100	100	100	31.9	100	100	100	99	100	32.3
Image8	100	100	100	100	100	32.6	100	100	100	100	100	33.3
Image9	99.4	99.7	100	99.4	98.7	33.7	100	100	100	100	100	33.9

sender creates two jigsaw puzzle images with semicircles and block lines drawn in different colors. Then, the JPEG compression with a quality level of medium (6) and JPEG to GIF recompression are applied to each created jigsaw puzzle image. Finally, the sender applies the proposed extraction process to each modified jigsaw puzzle image and picks the one with all semicircles correctly detected in its modified version.

We have also done experiments to see the detection performance under different block sizes and radius of semicircles. We first take nine 480×352 images shown in Fig. 11. As mentioned previously, r should be >2, and to avoid two neighboring attached semicircles overlapping, each semi-circle should be attached at position P with $P\in[3r,\ n-3r]$. In order to embed at least $k+1$ bits in each block side, we should restrict that $n-6r\geqslant 2^k$. Here, we assume that $k=2$ and $r=3$, then $n\geqslant 22$. On the basis of this analysis, for each image, we first set $n=32$ and create 10 jigsaw puzzle images for $r=3$, 4, respectively. Next, we resize the nine images into 480×336 ones, and for each one, we set $n=24$ and create 10 jigsaw puzzle images for $r=3$, 10 jigsaw puzzle images with $n=48$, $r=3$, 4, 5, respectively. Then, the JPEG compression with a quality

Table 3 Compression mode: GIF formate.

	Jigsaw puzzle pattern drawn by											
	White color						Black color					
	Correct detection rate (%)					PSNR	Correct detection rate (%)					PSNR
Image1	100	100	100	100	100	33.6	100	100	100	100	100	33.6
Image2	100	100	100	100	100	37.8	100	100	100	100	100	37.7
Image3	100	100	100	100	100	32.7	100	100	100	100	100	32.7
Image4	100	100	100	100	100	36.5	100	100	100	100	100	36.5
Image5	100	100	100	100	100	33.7	100	100	100	100	100	33.7
Image6	100	100	100	100	100	32.6	100	100	100	100	100	32.6
Image7	100	100	100	100	100	33.5	100	100	100	99	100	33.5
Image8	100	100	100	100	100	32.6	100	100	100	100	100	32.6
Image9	100	100	100	100	100	34.0	100	100	100	100	100	34.0

Table 4 Format conversion: JPEG quality level high (8) to GIF.

	Jigsaw puzzle pattern drawn by											
	White color						Black color					
	Correct detection rate (%)					PSNR	Correct detection rate (%)					PSNR
Image1	100	100	100	100	100	30.2	100	100	100	100	100	30.7
Image2	100	100	100	100	100	32.5	100	100	100	100	100	32.8
Image3	100	100	100	100	100	30.0	100	100	100	100	100	30.4
Image4	100	100	100	100	100	31.6	100	100	100	100	100	32.0
Image5	100	100	100	100	100	30.6	100	100	100	100	100	30.9
Image6	100	100	100	100	100	29.7	100	100	100	100	100	29.8
Image7	100	100	100	100	100	29.7	99.4	99.4	100	99	100	30.0
Image8	100	100	100	100	100	29.7	100	100	100	100	100	30.1
Image9	98.7	99.7	99.4	98.7	98.4	30.8	100	100	100	100	100	31.0

level of medium (6) and JPEG to GIF recompression are applied to each created jigsaw puzzle image. By applying the proposed extraction method to each modified jigsaw puzzle image, the drawn semicircles can be correctly detected. From these experimental results, we suggest to choose $r \geqslant 3$ and n $\geqslant 2^k+6r$ to embed $k+1$ bits in each block side, these will result in the correct detection.

3.2 *Security and Undetectability*

Security refers to the inability of an eavesdropper to detect hidden information. In practice, a steganographic scheme is considered secure if no existing attack can be modified to build a detector that would be able to distinguish between cover and stego images with a success better than random guessing.[3]

Here, we will show that the proposed method is undetectable; that is, it is immune from visual and statistical attacks. For each cover image, consider each block side as a DS and generate a random number sequence T. Based on T, a corresponding jigsaw puzzle image without embedding any data is created. Figure 14 shows such a jigsaw puzzle image without data embedded. Comparing to Fig. 8(b), we cannot find any abnormal pattern in Fig. 14. This means that the proposed method is visually undetectable. As to

Table 5 Format conversion: GIF to JPEG quality level medium (6).

	Jigsaw puzzle pattern drawn by											
	White color						Black color					
	Correct detection rate (%)					PSNR	Correct detection rate (%)					PSNR
Image1	100	100	100	100	100	26.2	100	100	100	100	100	28.2
Image2	100	100	100	100	100	31.2	100	100	100	100	100	31.4
Image3	100	100	100	100	100	28.4	100	100	100	100	100	29.6
Image4	100	100	100	100	100	27.9	100	100	100	100	100	29.7
Image5	100	100	100	100	100	29.3	100	100	100	100	100	30.2
Image6	100	100	100	100	100	28.8	100	100	100	100	100	28.8
Image7	100	100	100	100	100	24.9	99.4	99.4	99.7	99	100	25.4
Image8	100	100	100	100	100	24.2	99.4	99.7	99.7	100	99.4	26.2
Image9	98.4	99.4	99.4	98.4	98.4	29.7	100	100	100	100	100	29.9

Table 6 Recompression: JPEG quality level high (8) to quality level medium (6).

	Jigsaw puzzle pattern drawn by											
	White color						Black color					
	Correct detection rate (%)					PSNR	Correct detection rate (%)					PSNR
Image1	100	100	100	100	100	26.2	100	100	100	100	100	28.2
Image2	100	100	100	100	100	30.8	100	100	100	100	100	30.9
Image3	100	100	100	100	100	28.3	100	100	100	100	100	29.5
Image4	100	100	100	100	100	27.8	100	100	100	100	100	29.4
Image5	100	100	100	100	100	29.2	100	100	100	100	100	30.1
Image6	100	100	100	100	100	28.6	100	100	100	100	100	28.2
Image7	100	100	100	100	100	24.8	99	99	99	99	97.7	25.2
Image8	100	100	100	100	100	24.0	99.4	99.4	99.4	99.7	99.7	26.1
Image9	98.4	99	99.4	98.4	98	29.5	100	100	100	100	100	29.7

statistical undetectability, until now, although there are many jigsaw puzzle images on the Internet, there are no rules restricting the orientation, size, and attached position of the drawn semicircle. Thus, no reference can be used to judge if a jigsaw puzzle image is abnormal.

Assume that an ordinary jigsaw puzzle image is created by randomly attaching a semicircle in a right (bottom) block side (see Fig. 15, from Ref. 33), the distribution of the attached positions of all semicircles in the ordinary jigsaw puzzle image will be near uniform. For a jigsaw puzzle image, let $U=\{u_i|u_i\in[3r,n-3r]\}$ be the set of all different attached positions of semicircles and let p_i be the probability of u_i. Define the entropy[34] of the distribution of U as

$$H(U)=-\sum_{i=0}^{n-6r} p_i \log_2 p_i. \tag{9}$$

The entropy can be used to judge if a distribution is uniform. If a distribution with v samples is uniform, its entropy will be maximum and equal to $\log_2 v$. In general, a good random number generator will generate a sequence of random numbers with near-uniform distribution. As described in Sec. 2.1, due to a random number sequence T used to adjust the attached positions [see Eqs. (1) and (2)],

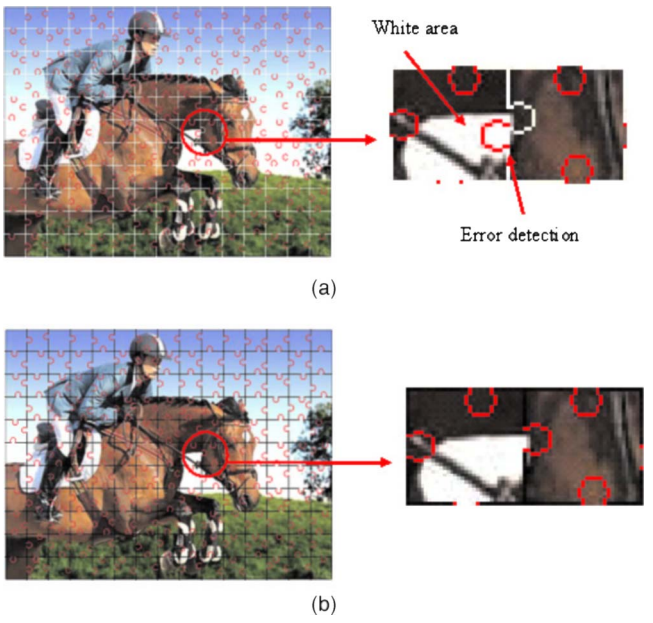

Fig. 13 A jigsaw puzzle image using different colors to draw puzzle patterns: (a) Using white with four error detections and (b) using black with no error detection.

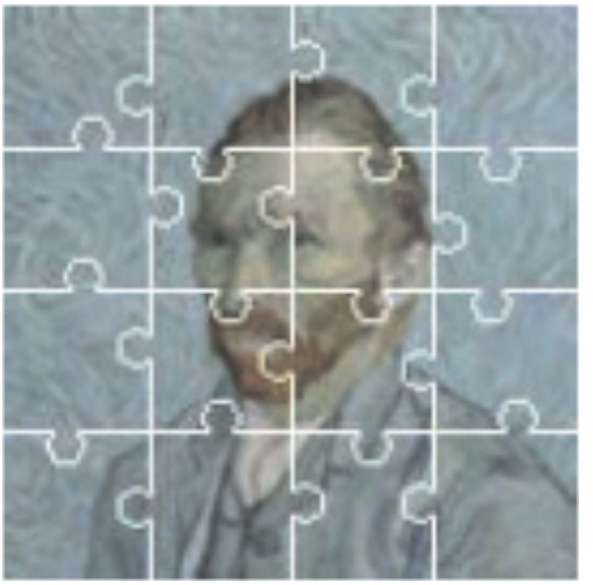

Fig. 14 Example for a jigsaw puzzle image without data embedded.

Fig. 15 Jigsaw puzzle image from Ref. 33 with attaching positions randomly selected.

for a jigsaw puzzle image with data embedded, the distribution of U will also be near uniform. To show this point, we have created 100 jigsaw puzzle images like Fig. 12 with $n=32$, $r=4$, and without data embedding. Thus, we have $n-6r+1$ $(=9)$ different attached positions, that is $\nu=9$, and $\log_2 9=3.17$. Each created jigsaw puzzle image corresponds to a sequence of randomly generated numbers T. We calculate the entropy of the distribution of attached positions in each jigsaw puzzle image to get 100 entropy values, which are in the range 3.107–3.166 with a mean of 3.15 and standard derivation of 0.000105. We also create another 100 jigsaw puzzle images with data embedded. The entropy range is 3.123–3.165 with a mean of 3.149 and standard derivation of 0.000094. These results demonstrate that the distribution of the jigsaw puzzle images with data embedded is very close to that of the jigsaw puzzle images without data embedded; thus, the proposed method is statistically undetectable.[3]

On the other hand, those existing attacks,[3] which try to find some statistical properties of pixel values in an image to determine if the image hides a secret message, do not work for our proposed method. The reason is that in the proposed method, for a created jigsaw puzzle image, except those pixels in the drawn semicircles and puzzle lines, the pixel values of other pixels are exactly the same as those in the original cover image. This makes the proposed method immune from those existing attacks. Furthermore, because the proposed method uses a stegokey as the seed of the random number generator G to adjust the attached positions of the drawn semicircles, even if the attacker has enough knowledge of the proposed method and can extract the attached positions, without the stegokey, the attacker still cannot get t_i, such that he cannot judge if a block side is a DS or IS and cannot understand what the attached position stands for. For these reasons, we can conclude that the proposed method is undetectable and secure.

3.3 *Capacity*

In our method, an $N1\times N2$ image divided into $n1\times n2$ blocks will have $N1/n1\times N2/n2$ blocks and $2[(N1/n1-1)\times(N2/n2-1)]+(N1/n1-1)+(N2/n2-1)$ embeddable block sides. Let the radius of the semicircle be r, and the attached positions of semicircles be in $[3r,n-3r]$, then each block side can allow $\lfloor\log_2(n-6r)\rfloor+1$ embedded bits. Thus, the embedding capacity is $(\lfloor\log_2(n-6r)\rfloor+1)\times\{2[(N1/n1-1)\times(N2/n2-1)]+(N1/n1-1)+(N2/n2-1)\}$ bits. If $n1=n2=32$, $r=4$, the embedding capacity of a 512×512 image is 240 bytes, a 1024×1024 image can allow 992 bytes. Although the capacity is not very large, it is useful to embed some critical messages, such as a secret key or an assignment with date and place. And most important is that our method is undetectable and robust to compression/recompression and format conversion. On the other hand, the embedding capacity of those existing methods[7–19] is about 6.5–13%. Although, these methods have higher capacity than ours, they are not robust to recompression and format conversion; some of them are even not robust to compression. Furthermore, they are only statistically undetectable for a specified model or statistics and do not preserve potentially hundreds of statistic quantities used in current blind steganalysis methods.[3] As mentioned previously, the main requirement of steganography is undetectability,[3] and, even if the warden is unable to find the secret message with statistical attacks, he still may intentionally distort the content, thus limited robustness is needed. On the basis of these two points, our method is valuable; even less capacity is provided.

4 Conclusion

The more kinds of images a steganographic system can use to embed data, the more secure the system is. We have proposed a novel steganographic method that uses a new kind of image, called jigsaw puzzle images, to embed a secret message. A secret message is hidden by creating a jigsaw puzzle image. The proposed method is robust against format conversion, lossy compression, and change in compression quality. We have also shown that the proposed method is undetectable and secure. This paper focuses on classic jigsaw puzzle images. In the future, we will extend to other kinds of jigsaw puzzle images.

Acknowledgments

This work is supported by Grant No. NSC 97-2221-E-007-122-MY3. The authors thank the anonymous reviewers for their many valuable suggestions, which have greatly improved the presentation of the paper.

References

1. N. Johnson and S. Jajodia, "Exploring steganography: seeing the unseen," *Computer* **31**(2), 26–34 (1998).
2. N. Provos and P. Honeyman, "Hide and seek: an introduction to steganography," IEEE Security Privacy Mag. **1**, 32–44 (2003).
3. I. J. Cox, M. L. Miller, J. A. Bloom, J. Fridrich, and T. Kalker, *Digital Watermarking and Steganography*, pp. 429–495, Morgan Kaufmann Pub., Burlington (2008).
4. A. Westfeld and A. Pfitzmann, "Attacks on steganographic systems," in *Proc. 3rd Int. Workshop in Information Hiding*, pp. 61–76, Springer-Verlag, Berlin (1999).
5. J. Fridrich, M. Goljan, and D. Hogea, "Steganalysis of jpeg images: breaking the F5 algorithm," in *Proc. 5th Int. Workshop in Information Hiding*, pp. 310–323, Springer-Verlag, Berlin (2002).
6. S. Dumitrescu, X. Wu, and Z. Wang, "Detection of LSB steganography via sample pair analysis," in *Proc. 5th Int. Workshop on Information Hiding*, pp. 355–372, Springer-Verlag, Berlin (2003).
7. Y. K. Lee and L. H. Chen, "High capacity image steganographic model," *IEE Proc. Vision Image Signal Process.* **147**(3), 288–294 (2000).

8. C. K. Chan and L. M. Cheng, "Hiding data in images by simple lsb substitution," *Pattern Recogn.* **37**, 469–474 (2004).
9. A. Brown, S-Tool V4, ⟨ftp://idea.sec.dsi.unimi.it/pub/security/crypt/code/s-tools4.zip⟩.
10. J. Fridrich and R. Du, "Secure steganographic methods for palette images," in *Proc. 3rd Information Hiding Workshop*, *Lect. Notes Comput. Sci.* **1768**, 47–60 (1999).
11. D. Upham, "JPEG-JSteg," ⟨http://www.funet.fi/pub/crypt/stefanography/jpeg-jsteg-v4.diff.gzee⟩ (1997).
12. N. Provos, "Defending against statistical steganalysis," in *Proc. of 10th Usenix Security Symp.*, pp. 323–335, Usenix Assoc., Berkeley (2001).
13. A. Westfeld, "F5—a steganographic algorithm: high capacity despite better steganalysis," in *Proc. of 4th Int. Workshop in Information Hiding*, pp. 289–302, Springer-Verlag, Berlin (2001).
14. P. Sallee, *Model-Based Steganography*, *Lect. Notes Comput. Sci.* **2939**, 254–260 (2004).
15. Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Trans. Circuits Syst. Video Technol.* **16**(3), 354–362 (2006).
16. J. H. Lee and M. Y. Wu, "Reversible data-hiding method for palette-based images," *Opt. Eng.* **47**(4), 047008 (2008).
17. J. Fridrich, M. Goljan, and D. Soukal, "Perturbed quantization steganography," ACM Multimed. Security J. **11**(2), 98–107 (2005).
18. J. Fridrich, M. Goljan, and D. Soukal, "Wet paper codes with improved embedding efficiency," IEEE Trans. Inf. Security Forensics **1**(1), 102–110 (2006).
19. J. Fridrich and D. Soukal, "Matrix embedding for large payloads," IEEE Trans. Inf. Security Forensics **1**(3), 390–394 (2006).
20. K. L. Chung, C. H. Shen, and L. C. Chang, "A novel SVD- and VQ-based image hiding scheme," *Pattern Recogn. Lett.* **22**(9), 1051–1058 (2001).
21. C. S. Lu and H. Y. M. Liao, "Oblivious cocktail watermarking by sparse code shrinkage: a regional- and global-based scheme," in *Proc. Int. Conf. on Image Processing*, Vancouver, Canada, Vol. 3, pp. 13–16, IEEE, Piscataway, NJ (2000).
22. T. S. Chen, C. C. Chang, and M. S. Hwang, "A virtual image cryptosystem based upon vector quantization," *IEEE Trans. Image Process.* **7**(10), 1485–1488 (1998).
23. ⟨http://www.jigzone.com, last accessed June 4, 2009⟩.
24. ⟨http://www.spintop-games.com, last accessed June 4, 2009⟩.
25. ⟨http://www.kraisoft.com/puzzle-games/real-jigsaw-puzzle, last accessed June 4, 2009⟩.
26. ⟨http://www.jigsaw2order.com, last accessed June 4, 2009⟩.
27. ⟨http://www.alljigsawpuzzles.co.uk/personalised_jigsaw_puzzles.htm, last accessed June 4, 2009⟩.
28. ⟨http://howto.nicubunu.ro/gimp_jigsaw_puzzle, last accessed June 4, 2009⟩.
29. M. Kharrazi et al., ⟨http://www.photoshopzilla.com/How-to-make-jigsaw-puzzle-pieces_43.html, last accessed June 4, 2009⟩.
30. ⟨http://www.essortment.com/all/howtomakeyour_rgzx.htm, last accessed June 4, 2009⟩.
31. Bajaj, http://www.tutorials.com/07/0738/0738.asp, last accessed June 4, 2009⟩.
32. M. Kharrazi, H. T. Sencar, and N. Memon, "Cover selection for steganographic embedding," in *IEEE Int. Conf. on Image Processing*, pp. 117–120, IEEE, Piscataway, NJ (2006).
33. ⟨http://www.indezine.com/products/powerpoint/cool/puzzlepictures.html, last accessed June 4, 2009⟩.
34. C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.* **27**, 379–423, 623–656 (1948).

En-Jung Farn received his BS from the Department of Computer Science and Information Engineering, Chiao Tung University, Hsinchu, Taiwan, in 2004. He is currently a PhD student of the Department of Computer Science, National Tsing Hua University. His current research interests include image processing and information security.

Chaur-Chin Chen received his BS in mathematics from National Taiwan University, Taipei, in 1977, and his MS in both mathematics and computer science, and PhD in computer science, all from Michigan State University, East Lansing, Michigan, in 1982, 1984, and 1988, respectively. He worked as a research associate at MSU in 1989 and a visiting scholar at MSU from September to December in 1997. Presently, he is a professor in the Department of Computer Science at National Tsing Hua University. His current research interests are information hiding, biometrics, and microarray image data analysis.