Reversible Binary Image Watermarking Method Using Overlapping Pattern Substitution

Keming Dong, Hyoung Joong Kim, Yong Soo Choi, Sang Hyun Joo, and Byung Ho Chung

This paper presents an overlapping pattern substitution (PS) method. The original overlapping PS method as a reversible data hiding scheme works well with only four pattern pairs among fifteen possible such pairs. This paper generalizes the original PS method so that it will work well with an optimal pair from among the fifteen possible pattern pairs. To implement such an overlapping PS method, changeable and embeddable patterns are first defined. A class map is virtually constructed to identify the changeable and embeddable pairs. The run-lengths between consecutive least probable patterns are recorded. Experiments show that an implementation of our overlapping PS method works well with any possible type of pairs. Comparison results show that the proposed method achieves more embedding capacity, a higher PSNR value, and less human visual distortion for a given embedding payload.

Keywords: Binary watermarking, reversible data hiding, pattern substitution.

Manuscript received Sept. 24, 2014; revised Apr. 21, 2015; accepted June 8, 2015.

This work was supported by the Technology Innovation Program of the Ministry of Trade, Industry & Energy, Rep. of Korea (No. 10050653, Research Standardization Project for Multimedia Integrity Verification via Reversible Data Hiding Technique), Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. 2013R1A1A1013410), and the grant from the National Natural Science Foundation of China (No. 61170207).

Keming Dong (kevindong333@hotmail.com) and Hyoung Joong Kim (corresponding author, khj-@korea.ac.kr) are with the Graduate School of Information Management and Security, Korea University, Seoul, Rep. of Korea.

Yong Soo Choi (ciechoi@sungkyul.ac.kr) is with the Division of Liberal Arts and Teaching (Multimedia), Sungkyul University, Anyang, Rep. of Korea.

Sang Hyun Joo (joos@etri.re.kr) and Byung Ho Chung (cbh@etri.re.kr) are with the SW & Contents Research Laboratory, ETRI, Daejeon, Rep. of Korea.

I. Introduction

Digital data hiding techniques hide data into digital files and have been used for copyright protection, covert communications, tamper detection, and authentication of digital property. There are many watermarking techniques for color and gray-scale images [1]–[6]. However, these techniques cannot be used with binary images. Binary images have only one signal bit in every pixel; therefore, this makes binary image watermarking very challenging.

There are two kinds of data hiding methods for binary images — lossy watermarking and lossless watermarking (also known as reversible watermarking). In the early development stages of binary image watermarking, most binary watermarking methods employed the method of lossy watermarking.

Chen and others [7] divided an image into disjoint blocks and then embeded a message by flipping one bit (that is, binary pixel value) in every block to compute a *weight score* using a randomly generated *weight matrix*, where the score is equal to the secret message.

Further improvements to the visual quality of an embedded image (stereo image) were made by introducing quality control when embedding data into blocks [8]. In every block, the number of black pixels can be either an odd or even number. The approach to embed one bit "1" or "0," in [8], is to flip pixels to control the number of black pixels in one block to be even or odd.

In [9]–[11], Wu and others proposed a *flippability score* considering the connectivity and smoothness of a square area comprising three-by-three pixels; Yang and Kot [12]–[13] proposed a *flippability criterion*.

An authentication scheme with tampering localization was

proposed by Cheng that uses a different flippability criterion to reduce image distortion [14]–[16].

Tzeng and Tsai [17] used an *optimal code holder* to embed binary data into every block.

Yang and others [18] used a morphological transform to achieve high embedding capacity. Unfortunately, the algorithms used in the aforementioned studies were all of the lossy type. As a result, there are not many reversible watermarking methods for binary images available today.

Tsai and others [19] have proposed a method called pair-wise logical computation (PWLC). It successfully exploits special patterns, such as "000000" and "111111," and hides secret data within these patterns. The PWLC method identifies such patterns on the boundaries (that is, edge area) of an image to reduce visual perceptibility. Thus, even though there are many cases of such special patterns within a single binary image, only a certain number of such special patterns on the boundaries of the image are used. The embedding rule is simple. If the secret bit is "0," then the third and fourth bit positions are flipped. Otherwise, if the secret bit is "1," then the third and fourth bit positions are flipped. Of course, embedding capacity is dependent upon the number of such patterns at the boundaries. Since this method utilizes lengthy special patterns of six bits and flips only the center bit positions, boundaries can look untidy after data hiding.

A pattern substitution (PS) method has been proposed by Ho and others [20] and is an improvement upon the PWLC method. The PS method hides data into the boundaries of a binary image. It locates the boundaries using an edge detector [21]. The authors' PS method uses four bits for each pattern; thus, there are sixteen possible patterns in total. Among them, two pattern types — more probable pattern (PM) and less probable pattern (PF) — are used to hide data. For each PM, its counterpart, PF, is decided by the PS method. After this, the PS method records all the locations of the PFs in advance. The PS method hides a secret bit "0" by rewriting a PM or PF pattern to PM, and it hides a secret bit "1" by rewriting a PM or PF pattern to PF. Since a four-bit pattern is used instead of a six-bit pattern, the PS method is less likely to have a significant effect on the appearance of the boundaries of an image.

There are two types of PS — block-wise disjoint PS (non-overlapping PS) and block-wise overlapping PS (overlapping PS). In the case of block-wise disjoint PS, all blocks are mutually exclusive to each other (that is, disjoint). Thus, embedding and extracting procedures are relatively simple. However, the embedding capacity is not so large. On the other hand, if we allow some blocks to overlap each other, as in the case of block-wise overlapping PS, then this allows us to hide more data. Ho and others [20] have presented both methods.

In Dong and Kim [22], a non-overlapping PS has been developed by reducing the PF location map. This method chooses the second least probable pattern, and it is defined as PFR in [21]. All the PF patterns are replaced with PFR patterns before the embedding process, and the sequence of PFs and PFRs is recorded using 0s and 1s in a new location map. The size of the new location map is equivalent to the sum of the PFs and PFRs. This approach can efficiently decrease the size of a PF location map; however, it can only be applied to a non-overlapping PS.

Xuan and others [23] have presented a different binary image reversible watermarking method. Their approach exploits a run-length histogram modification method. They construct a histogram of run-lengths from a binary image. A run-length pair consists of both a run-length of black pixels and a run-length of neighboring white pixels. By swapping one black pixel for one white pixel in the specific pairs, they can hide secret messages. The embedding capacity depends on the population counts of the most probable run-length and its neighboring run-length.

Zhang and others [24]–[25] have presented an optimal reversible data hiding method for *binary covers*. This method tries to embed binary data into binary covers. A cover is segmented into disjoint blocks, each block being of the same length. In each block, the pixels with value "0" are overwritten with a payload, and the positions of pixels with value "1" are recorded and embedded into the next block to achieve reversibility. The number of 1s in the payload determines the distortion. If the size of the payload is small, then the method losslessly decompresses the payload so that the number of 1s in the payload is reduced; in this way, the level of distortion is decreased.

In this paper, an improved version of the PS method in [20] is proposed. The overlapping PS version in [20] can be applied to only four pattern pairs among fifteen. Our method generalizes the algorithm so as to use all fifteen possible pattern pairs, which significantly increases the embedding capacity. We introduce two conditions for embedding based on two new concepts — changeable and embeddable patterns. If a pattern satisfies the two new conditions, then we can embed data into the pattern. These two new conditions reflect the interaction between the overlapping patterns that emerge after hiding data. If a pattern satisfies these two new conditions, even though this pattern may overlap with a neighboring pattern, then we can embed data without causing confusion in the extraction procedure. A new coding method is also proposed to decrease the size of the PF location map.

The remaining part of this paper is organized as follows. Section II reviews the original PS method and identifies the problem with this method. Then, the proposed improved PS method is explained in Section III. Important new concepts,

such as changeable and embeddable patterns and class map (CCM), are also introduced in this section. Section IV shows that the proposed method works well with any kind of pattern pairs. In addition, this section analyzes the performance of the proposed method. Finally, the conclusion is presented with future works in Section V.

II. PS Method

In the PS method, the *difference-value* matrix, denoted as **D**, is firstly derived from an original binary image according to

$$D(i,j)=\begin{cases} C(i,j) & i=1, j=1, \\ C(i,j)\oplus C(i-1,j) & i\neq 1, j=1, \\ C(i,j)\oplus C(i,j-1) & j>1, \end{cases} \quad (1)$$

where $\oplus$ stands for the exclusive-OR logic operation; i and j are the indices of the rows and columns in the binary image, respectively; and C is the matrix of the original binary image. Equation (1) shows that there are three different types of pixels in the binary image. The first case (when $i = 1, j = 1$) is for the pixel at the top-left corner. The difference value of this pixel is the value of the pixel in the original binary image. The second case (when $i \neq 1, j = 1$) is for the pixels in the first column, excluding the first pixel. The difference values of each of these pixels are defined as exclusive-OR with its upper pixel. In the third case (when $j > 1$), which is for the rest of the pixels, the difference values for these pixels is defined as exclusive-OR with its left pixel. One example of a binary image and its corresponding difference-value matrix is shown in Fig. 1.

Each set of four consecutive difference values within difference-value matrix **D** (corresponding to four consecutive pixels) is labelled as being a specific *difference-value pattern* (from hereon in simply referred to as a *pattern*). A single pattern contains four bits. Therefore, there are sixteen possible patterns in total; P_{0000}, … , P_{1110}, and P_{1111}. One bit of secret data is hidden through one PS in **D**. To reduce the distortion caused by substitution, the corresponding patterns are listed in Table 1.

Two patterns are chosen to embed data. These two patterns are known as a *pattern pair* and are denoted as PM and PF. To minimize image degradation, the following two requirements are necessary:

- The PM and PF elements that make up a pattern pair must come from the same pattern set.
- A PS is to achieve a Hamming distance of "1" between each set of four bits between the original and watermarked images.

All the difference-value patterns are classified as belonging to either an "odd pattern set" or an "even pattern set" based on the number of 1s occurring within each of the difference-value patterns.

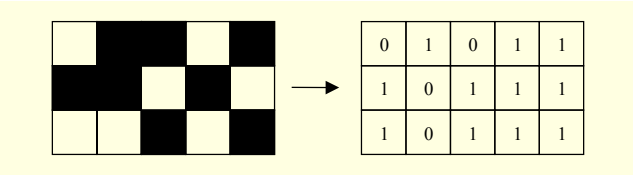


Fig. 1. Binary image and its corresponding difference-value matrix.

Table 1. Binary bit pattern and corresponding possible substitutable patterns.

Difference-value pattern	Substitutable patterns
P_{0001}	P_{0010}, P_{0111}, P_{1101}
P_{0010}	P_{0001}, P_{0100}, P_{1110}
P_{0011}	P_{0000}, P_{0101}, P_{1111}
P_{0100}	P_{0010}, P_{0111}, P_{1000}
P_{0101}	P_{0011}, P_{0110}, P_{1001}
P_{0110}	P_{0000}, P_{0101}, P_{1010}
P_{0111}	P_{0001}, P_{0100}, P_{1011}
P_{1000}	P_{0100}, P_{1011}, P_{1110}
P_{1001}	P_{1000}, P_{1010}, P_{1111}
P_{1010}	P_{0110}, P_{1001}, P_{1100}
P_{1011}	P_{1000}, P_{0111}, P_{1101}
P_{1100}	P_{0000}, P_{1010}, P_{1111}
P_{1101}	P_{0001}, P_{0111}, P_{1101}
P_{1110}	P_{0010}, P_{0111}, P_{1101}
P_{1111}	P_{0010}, P_{0111}, P_{1101}

Each pattern pair consists of a PM and a PF. The PM and PF of a pattern pair should be chosen from the same pattern set (either from the odd pattern set or even pattern set). This statement can be explained with a simple example. Consider a binary bit stream "0010000" from an original image, and assume the bit in front of this stream is "0," then this bit stream can produce the *difference-value* stream "0011000" by (1). For example, assume the binary stream in the image is "0010." If the bit in front of this stream is "0," then the binary stream is "00010;" the *difference value* is "0011." Otherwise, if it is "1," then the difference value is "1011." The first pattern, that is P_{0011}, can be replaced with P_{0010}. Note that P_{0011} belongs to the even group, but P_{0010} to the odd group. As a result, the substituted string becomes 0010000, which produces the watermarked string 0011111 after an inverse transform. The important thing to note here is that the bit string following the first four bits in the watermarked stream is inverted from the original stream. This fact implies that the image quality after data hiding will become extremely poor. However, if the PF and PM are chosen from the same group, then this kind of

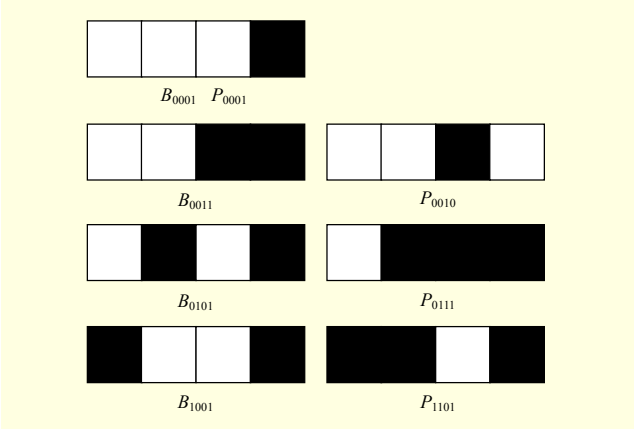


Fig. 2. Binary bit pattern B_{0001} (with a difference value of P_{0001}) can be substituted for either B_{0011}, B_{0101}, or B_{1001} (with a difference-value pattern of either P_{0010}, P_{0111}, or P_{1101}, respectively).

thing does not happen.

With regards to trying to achieve a Hamming distance of "1" between each set of four bits between the original and watermarked images, "a Hamming distance of 1" means that only one bit is flipped between the two bit streams in question. For a given bit stream, say B_{wxyz}, there are four different possible bit streams that produce a Hamming distance of "1" when paired with B_{wxyz}. For example, bit streams B_{1001}, B_{0101}, B_{0011}, and B_{0000} are four possible streams with a Hamming distance of "1" to B_{0001}. Note that there are PM and PF pairs ensuring a Hamming distance of "1" between the original and watermarked images.

Consider the binary bit pattern B_{0001} and assume that its corresponding difference-value pattern is P_{0001}. The candidates for substitution should then be of the form B_{xxx1}, where x stands for a "don't care" bit. Consider the "Hamming distance of 1" requirement. As a result, three possible output patterns can be B_{1001}, B_{0101}, and B_{0011}, for which the corresponding difference-value patterns are P_{0010}, P_{0111}, and P_{1101}, respectively (see Fig. 2). It can be verified that this is also the case for when the previous bit value is "1." Thus, P_{0001} can be a pair to P_{0010}, P_{0111}, or P_{1101}. In other words, P_{0001} can be substituted by any one of P_{0010}, P_{0111}, and P_{1101}. According to this, the substitutable difference-value patterns for any pattern can be found (see Table 1).

Note that statistically P_{0000} is the most probable pattern. However, this pattern is not used due to the *visibility problem*. The most probable patterns among P_{0001}, … , P_{1110}, and P_{1111} are named as PM. For each PM, the least probable pattern among the three candidates from Table 1 is named as PF.

Data hiding will be carried out by PS. After determining a suitable PM-PF pair, the PS method will rewrite all PM-PF pairs to PM to hide a bit 0; likewise to PF to hide a bit 1. The location map of PF patterns has to be embedded to the image too. In this way, the secret data can be extracted and the original image can be recovered at the decoder.

There are two variants of the PS method in Ho and others [20]. The *non-overlapping PS method* processes the difference-value patterns one by one, and the different difference-value patterns contain no overlapping parts. However, different difference-value patterns intersect each other in the *overlapping PS method*; special care needs to be taken with this approach, since tricky situations to be discussed below can happen.

Assume that there is a binary bit pattern contained within a binary stream, such as in 0000011111, and let us assume that P_{0001} and P_{0111} make up the corresponding PM-PF substitution pair. In this case, an encoder, scanning the stream one bit at a time, encounters the first PM — the underlined pattern P_{0001}. The PM can be kept as it is when a bit 0 is hidden. A decoder can recover the hidden message and keep the original pattern as it is. On the other hand, the PM can be substituted by PF, in which case the resultant stream becomes 0001111111. However, the decoder assumes that the underlined pattern carries a hidden bit 0 — an incorrect interpretation.

To avoid such a misinterpretation, Ho and others [20] proposed the following two rules:

1. When the PM and PF overlap, take PF as the hidden pattern.
2. If the numeral "1" occurs five times consecutively, then "11111" is deemed an unacceptable hiding place.

Under the above rules, an encoder will skip an input string of "0000011111" in accordance with rule 2.

When P_{0001} is first encountered in the following input string "0000011110," the encoder searches for a P_{0111} pattern to see whether the PM and PF overlap each other. In this string, 0000011110, since the two patterns overlap, the encoder skips the underlined P_{0001} pattern and uses the pattern P_{0111} to hide a bit according to rule 1. If the bit to be hidden is "0," then the PF is changed into PM; that is, 0000000110. Otherwise, the PF remains as it is. Thus, there are two possibilities after data embedding — 0000000110 or 0000011110. As a result, there are no problems encountered in the decoding process.

III. New PS Method

The basic algorithm of the proposed method is quite similar to that featured in the work of Ho and others [20]. In this paper, two important concepts are considered — *embeddability* and changeability — to generalize the original PS method. These two concepts will be explained in the following examples.

Even though Ho and others [20] proposed the two aforementioned rules to avoid misinterpretation at the decoding stage, these rules only work well with the following four specific pairs: $<P_{0001}, P_{0111}>$, $<P_{1000}, P_{1110}>$, $<P_{0011}, P_{1111}>$, and

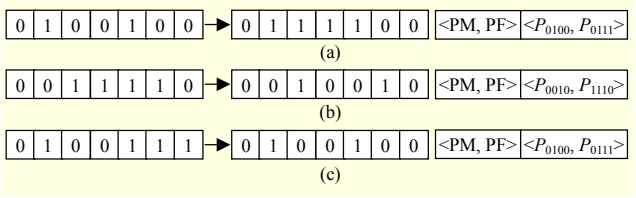


Fig. 3. Three counterexamples to Ho et al.'s rules [20].

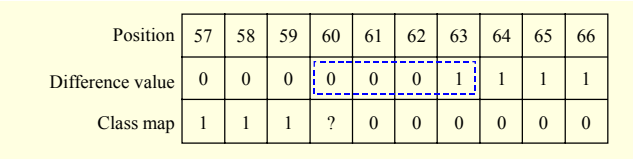


Fig. 4. Illustration of CM where PM is 0001 and PF is 0111.

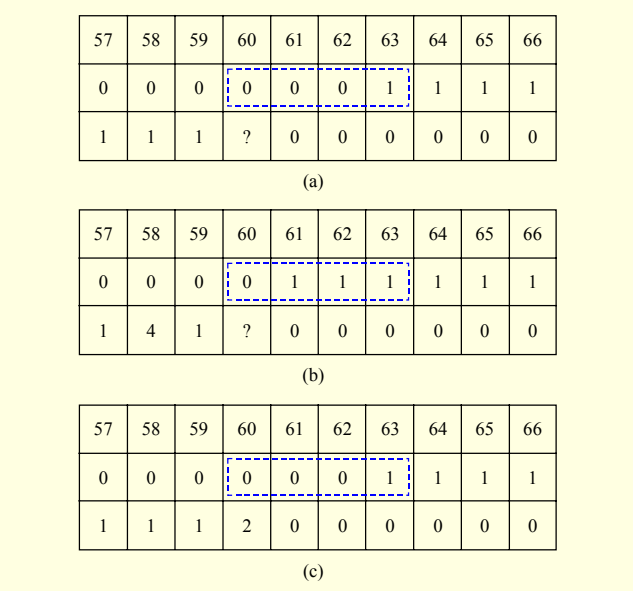


Fig. 5. Example of pattern that is not changeable when PM is 0001 and PF is 0111: (a) before substitution, (b) after substitution, and (c) final class of pattern 60.

<P_{1100}, P_{1111}>. Unfortunately, these rules do not work for other patterns.

Figure 3 shows three counterexamples that cannot be correctly decoded by Ho and others' work [20]. The input string in Fig. 3(a), 0100100, is encoded as 0111100 when the PM and PF pair is P_{0100} and P_{0111}, respectively. In this encoding, only one bit has been hidden. However, the decoder will try to extract two bits of message. The decoder first encounters the underlined pattern P_{0111} in 0111100, which is decoded as 0100100. Another underlined pattern, 0100100, is about to be decoded; the second decoding should not be carried out.

The input string in Fig. 3(b), 0011110, is encoded as 0010010, where the PM and PF pair is P_{0010} and P_{1110}, respectively. In this encoding, only one bit is hidden using the PF pattern P_{1110} (that is, 0011110). However, the decoder tries to extract two bits of message. It first encounters the underlined pattern as P_{0010} in 0010010, which is decoded as 0010010. Another underlined pattern, 0010010, is about to be decoded.

Consider the input string in Fig. 3(c), 0100111, where the PM and PF pair is P_{0100} and P_{0111}, respectively. According to rule 1, an encoder can hide one bit into the PF, whereby the encoded string would become 0100100. The decoder will extract two bits even though just one bit has been hidden. The first bit is recovered from the first PM pattern (P_{0111}) and the second bit from the second PM pattern (P_{0111}) again.

To generalize the original PS method, further refinement is needed. The refinement is made possible by categorizing patterns into five classes, as follows:

- Class 0: those that form a pattern not yet processed or that is waiting to be processed.
- Class 1: the proposed pattern containing neither PM nor PF.
- Class 2: the processed pattern of PM or PF, but not possible to embed data.
- Class 3: the processed pattern of PF and has been used to embed data.
- Class 4: the processed pattern of PM and has been used to embed data.

Every pixel belongs to one class only. An illustration is shown in Fig. 4. A table holding the class information is referred to as a CM. Pixels that lie within a distance of one to three pixels from a pixel are called *neighboring-range* pixels; those that are at a distance of one to six pixels are called *relative-range* pixels. A pattern's representative position is the position of its first pixel.

Let us assume that at position 60 (see Fig. 4) a pattern's difference-value substring is 0001, which is a PM pattern (that is, P_{0001}), and that this pattern is about to be processed by an encoder. Note that in Fig. 4 the PF pattern is 0111. Thus, the CM at positions 57 to 59 shows "1s"; the other patterns from position 61 onwards have not yet been processed; therefore, all are positions are showing a value of "0." The CM being described here is shown in Fig. 4.

At position 60, the PM pattern can be flipped to the PF pattern to embed a bit 1. In this case, after data hiding, the pattern 0001 is changed into the pattern 0111. In this case, after data hiding, the new pattern at position 58 becomes 0001, which is a PM pattern. Thus, the CM at position 58 is changed from "1" (before data hiding) to "4" (after data hiding: see Fig. 5). These kinds of changes to a CM can cause misinterpretation at a decoder. Such misinterpretation can be avoided. Thus, two kinds of patterns are introduced in the next paragraph.

Here, two new concepts are defined — a changeable pattern and an embeddable pattern. As is clear from the counterexamples, not all of the PM or PF patterns can be used

for data hiding. A changeable pattern should be either a PM or a PF before data hiding. In Fig. 5, let us assume that position 60 is the current position to be processed. In this case, the pixels at positions 57 to 59 and 61 to 63 are neighboring-range pixels. If the neighboring-range pixels of a CM do not change after PS, except in the cases of class 0 and class 2 pixels, then this PM or PF is known as a changeable pattern. Thus, changeability is dependent upon the values of neighboring-range pixels. In Fig. 5, the pattern at position 60 is not a changeable one, since after PS, at least one of the neighboring-range pixels is changed from a value of "1" to "4." Thus, in this case, an encoder should skip data-hiding because it is known that the pattern is not a changeable one. Now, the CM shows a value of "2" at position 60.

A changeable pattern should meet the following two conditions:

- The pattern should be PF or PM.
- After flipping the current pattern, any class 1 pattern in a neighboring range should remain as a class 1 pattern; any class 3 pattern should remain as a PF; and any class 4 pattern should remain as a PM.

When a PF or PM pattern is about to be substituted to embed data, the substitution has to meet at least two requirements. First, it should not destroy the patterns in which we have already embedded data. This means that PM or PF patterns of the class 3 or class 4 type in a neighboring range should not be destroyed. In other words, any PF pattern of the class 3 type should remain as a PF, and any PM pattern of the class 4 type should remain as a PM. Thus, any pattern in class 1 should be in the same class.

Assume a class 2 pattern is in the neighboring range. Due to the change of the current pattern (either PM or PF), the class 2 pattern can be changed into a class 1 pattern or remain as it is. In the first case, the class 2 pattern can be changed to neither PF nor PM (that is, into a class 1 pattern) after flipping. In the second case, the class 2 pattern can be flipped to either a PF or a PM and still be classed as a class 2 pattern. Thus, a class 2 pattern can be changed to any pattern without misinterpretation in a decoder.

Can an encoder always embed data when a pattern is changeable? The answer is not always. Assume that PM and PF are 1001 and 1111, respectively (see Fig. 6). Note that the pixel processing order can be randomized for security purposes. For example, a hiding gap (HG) [20] has been proposed, and an HG value is shared by an encoder and decoder.

Assume that the processing order is randomly decided as follows: for example, first, at position 72, second, at position 70, and finally at position 73. The pattern at position 72 is 0100, which is neither PM nor PF. As a result, its CM is "1." At position 70, the pattern to be processed is 1001, which is a PM.

67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82
0	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0
1	1	1	4	1	1	?	0	0	0	0	0	0	0	0	0

(a)

67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82
0	1	0	1	0	0	1	1	1	1	0	0	1	0	0	0
1	1	1	2	1	1	?	0	0	0	0	0	0	0	0	0

(b)

67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82
0	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0
1	1	1	4	1	1	2	0	0	0	0	0	0	0	0	0

(c)

Fig. 6. Changeable but not embeddable pattern when PM is 1001 and PF is 1111, and processing order is at position 72, position 70, and position 73: (a) before substitution, (b) after substitution, and (c) final class of pattern 73.

Thus, its CM is "4." At position 73, the pattern to be processed is 1001, which is also a PM. This pattern is changeable according to the changeability rule. However, we want to see what happens if this pattern is substituted for 1111, which is a PF. If this pattern becomes 1111, then the previous pattern 1001 at position 70 is not changeable at the decoder, and its class changes from "4" to "2" as shown in Fig. 6(b). Thus, the second PM pattern (at position 73) should be skipped without hiding any bit. One bit has been hidden into the first PM pattern at position 70, since it was both changeable and embeddable. However, an encoder has to skip the second PM pattern since it is not embeddable. The encoder can also hide one bit into the third PM pattern at position 76 since it is both changeable and embeddable under the assumption that the second pattern has been skipped.

It is necessary to see whether a pattern is embeddable when a number of PM patterns repeat. An embeddable pattern should meet the following conditions:

- The pattern should be changeable.
- After flipping or substituting the current pattern, the changeability of any PF/PM pattern of relative-range pixels should not change, except class 0.

The conditions for embeddability are stricter than those for changeability. We have only to check the changeability of a pattern when there are no consecutive repeated PM or PF patterns. However, it is necessary to check if a pattern is embeddable when a number of PM or PF patterns repeat. For this purpose, the current PM pattern is intentionally changed into PF, or PF into PM, to see whether the changeability condition of the previous pattern is kept. One bit of data can be embedded only into an embeddable pattern.

Changeability ensures that after substitution of the current pattern, the change should not generate a new PF or PM, nor should it destroy either a PF or a PM from any class 3 or 4 patterns in the neighboring range. Turning changeable patterns into unchangeable ones, or vice versa, can also cause misinterpretation at a decoder. The embeddability condition ensures that the condition of changeability will not change after a substitution.

Ho and others [20] used a tuple, (x_i, y_i), to record the ith PF pattern, PF_i, where x_i and y_i are its coordinates. This technique needs 18 bits to encode one tuple for a 512 × 512 image, which is not efficient. In this paper, the positions of PF pixels are rearranged into a one-dimensional array, and its run-length between PF patterns is recorded to reduce the size of a location map.

In the embedding procedure, the difference-value matrix is first scanned to get the pattern pair to embed data. In this process, the PM-PF pair is identified based on the population of each pattern. Side information should be sent to a decoder for correct decoding. This information will be hidden in a specific area in the image using bit replacement. The payload consists of the original bit stream replaced with side information, a location map, and the actual secret bits. Side information includes the PM and PF patterns, number of PF patterns, number of secret bits, and the number of bits used for run-length coding for the PF patterns.

A decoder first extracts the side information from a specific region. Next, the decoder constructs a CM to see whether a pattern is changeable and embeddable. The first part of the decoded message is used to recover the replaced pixel values with the original ones. The rest of the decoded message contains the location map and secret message.

IV. Experimental Results

The main purpose of this paper is to generalize the original PS method. Experiments show that the generalization is perfect; that is, any pattern pair can be used. The performance comparison between the proposed scheme, the PS method (overlapping case), and other approaches [19], [22]–[23] is carried out in this section. The host images are shown in Fig. 7, where the "Lena" and "Boat" images are converted to binary images; the "Mickey" and "Cartoon" images, which are downloaded from the Internet for research purposes, are binary; and the rest of them are documents. These three image types represent most of the binary images that exist in the world today.

1. Comparison with PS

To implement cyclic embedding and solve the uneven embedding problem, a HG [20] is used. For example, if HG = 2, then the hiding places are the first bit, the third bit, the fifth bit, and so on. To obtain a good visual performance, HG should be bigger than a given threshold; since, in this way, any distortion can be spread over the entire image. With this in mind, we used HG = 23 throughout our experiments.

Table 2 shows the PMs and PFs used by the original PS method [20] and proposed PS method. It shows that the proposed method can always choose the best PM-PF pairs to embed data.

Figure 8 shows a comparison of results in terms of embedding capacity based on the number of available flippable pixels. Note that the original PS method can hide 6,251 bits since there are 6,233 PMs and 18 PFs in the Mickey image. However, since not all patterns are changeable or embeddable,

Table 2. Comparison of original and proposed PS methods with HG value of 23.

	Original PS method				Proposed PS method				
	PF	PM	# of PFs	# of PMs	PF	PM	# of PFs	# of PMs	RL bits
Mickey	P_{1110}	P_{1000}	18	6,233	P_{0111}	P_{0100}	18	6,697	11
Body	P_{0111}	P_{0001}	4	5,959	P_{0111}	P_{0100}	4	6,644	11
English	P_{1110}	P_{1000}	82	6,188	P_{1111}	P_{1001}	5	4,998	11
					P_{0111}	P_{0100}	82	10,536	10
Chinese	P_{1110}	P_{1000}	19	6,134	P_{1111}	P_{1001}	0	3,885	N/A
					P_{0111}	P_{0100}	19	9,285	11
Handwriting	P_{0111}	P_{0001}	23	11,296	P_{1101}	P_{0001}	19	11,296	11
					P_{0111}	P_{0100}	23	12,124	11
Lena	P_{0111}	P_{0001}	586	7,673	P_{1101}	P_{0001}	496	7,673	7
Boat	P_{0111}	P_{0001}	697	8,685	P_{1101}	P_{0001}	428	8,685	8

ETRI Journal, Volume 37, Number 5, October 2015
http://dx.doi.org/10.4218/etrij.15.0114.1058

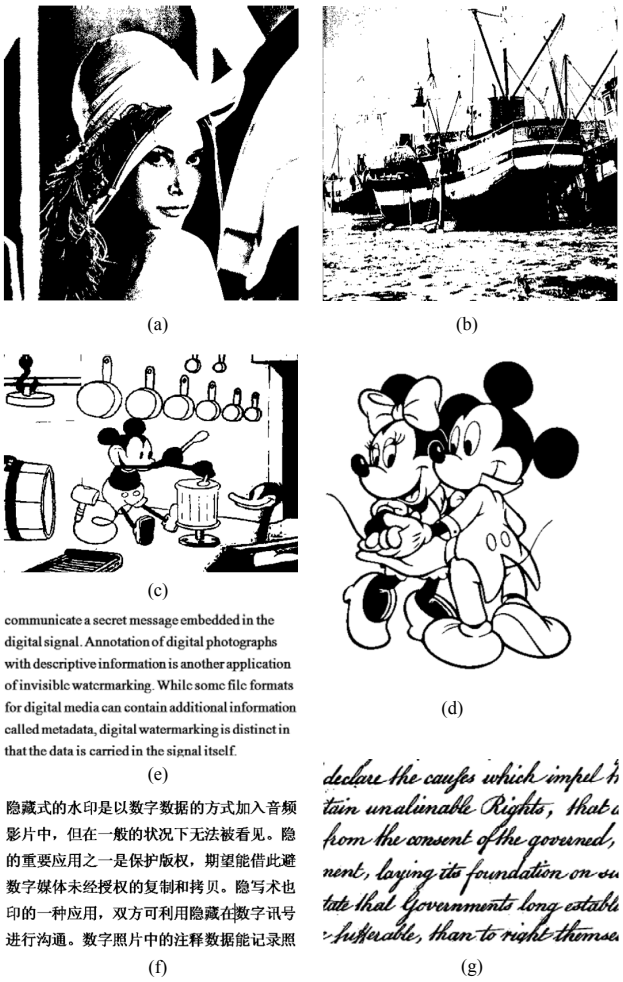


Fig. 7. Cover images used in experiments: (a) Lena (512×512), (b) Boat (512×512), (c) Mickey (500×370), (d) Cartoon (371×450), (e) English (560×273), (f) Chinese (560×273), and (g) Handwriting (605×378).

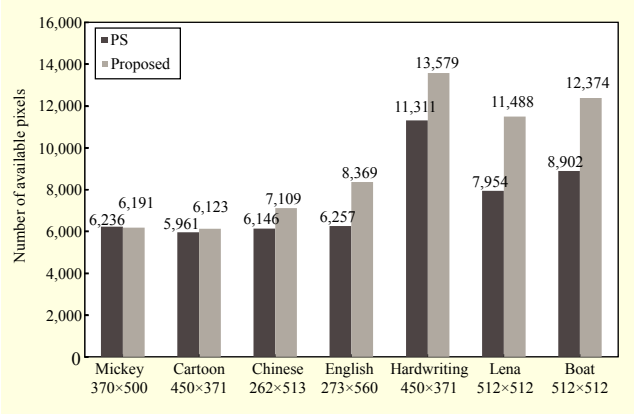


Fig. 8. Comparison results on number of flippable pixels.

the number of flippable pixels is a little smaller than the number of PM and PF patterns. Note that the proposed method can flip only 6,191 patterns due to the changeability and

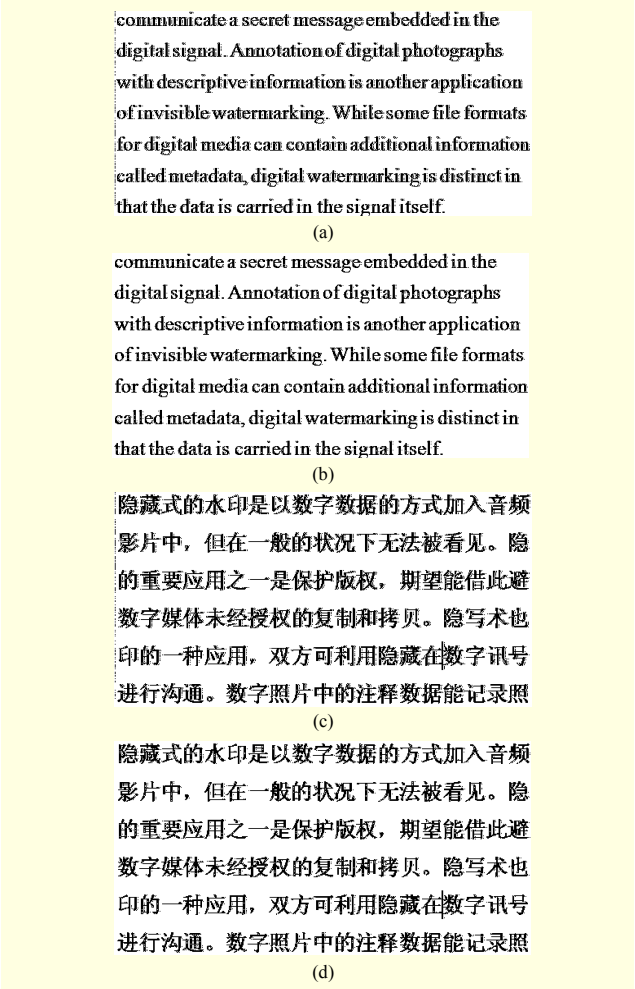


Fig. 9. Stego images by PS, (a) and (c), and proposed method, (b) and (d).

embeddability conditions when the pair P_{0100} and P_{0111} is used, even though there are 6,715 patterns. Thus, in this case, the proposed method can also use the pair P_{1000} and P_{1110}, as in the original PS method. Thus, it is shown that the proposed method can always hide more bits than the original PS method.

Figure 8 shows the proposed method obtains higher capacity than the original PS method. There are mainly two reasons for this. The first is that the original PS method can use only four (PF, PM) pairs including $<P_{0001}, P_{0111}>$, $<P_{1000}, P_{1110}>$, $<P_{0011}, P_{1111}>$, and $<P_{1100}, P_{1111}>$. Thus, these pairs may exclude the best pair for some images. In the proposed method, any pair can be used. The second reason is that the run-length-based approach uses fewer bits than the coordinate-based method; moreover, the location map can be contracted. The run-length coding for the location map can increase the capacity and also decrease the distortion. Table 2 shows the longest run-length requiring at most 11 bits.

Figure 9 compares the original and proposed PS methods in

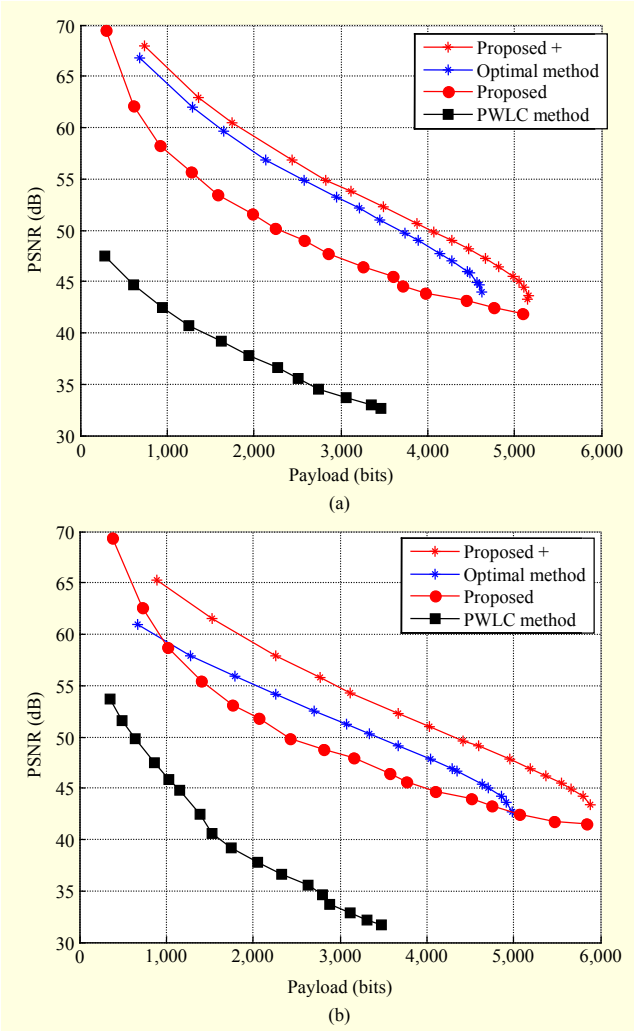


Fig. 10. Comparison of PSNR for (a) Lena (512 × 512) and (b) Boat (512 × 512).

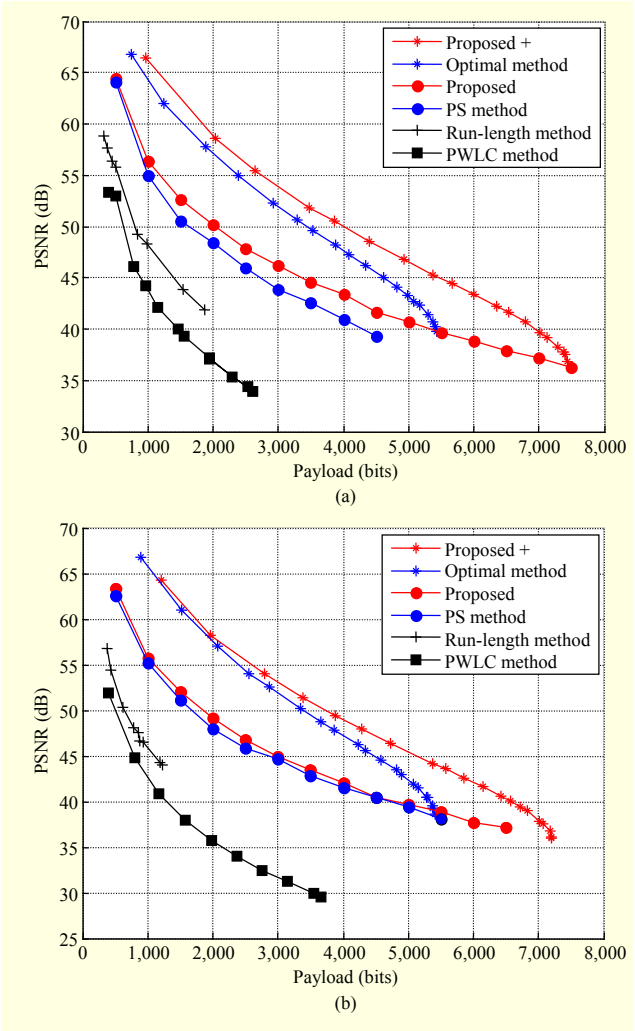


Fig. 11. Comparison of PSNR for (a) English (273 × 560) and (b) Chinese (513 × 262).

terms of visual quality. It shows that the original PS method produces fuzzier images than the proposed PS method.

In Figs. 10 and 11, the PS method improved by [23] is labelled as "Optimal method," our generalized PS is labelled as "Proposed," and our method improved by [23] is labelled as "Proposed +." In the Lena and Boat images in Fig. 10, the original PS cannot hide any data. Since the size of the location map is larger than the hiding capacity, there is no performance line for this PS method in Fig. 10. In the English and Chinese images, the proposed method is noticeably superior to the PS method.

2. Comparison with Other Approaches

Figures 10 and 11 also compare the performances of the PWLC [19], run-length [22], and optimal [23] methods. The run-length method cannot embed any data into Lena and Boat, because the size of the location map is bigger than the hiding capacity. Hence, there is no performance line for the run-length method in Fig. 10. In the English and Chinese images, the proposed method can achieve better performance than the run-length method. Since the location map of the run-length method cannot be compressed well, the performance is not good.

The PWLC method only embeds data into the boundary. It has to flip two pixels for 1 bit embedding, so the capacity performance as well as the distortion is worse than those for the proposed method.

The optimal method can be applied to any reversible watermarking method for binary covers to achieve better performance. Figures 10 and 11 show that the proposed method can achieve higher performance than the original PS method, and the proposed method with [23]'s improvement can achieve higher performance than the PS with [23]'s improvement.

V. Conclusion

In this paper, the original PS method is extended to include all possible pattern pairs by proposing the concepts of changeable and embeddable patterns, and a class map is virtually constructed to identify such patterns. To reduce the size of the location map (that is, the position information of all PF patterns), the run-length information is used. Experiments show that the proposed method works well with any kind of PM-PF pairs and produces better results compared with the original PS in terms of embedding capacity and visual quality.

References

[1] J. Tian, "Reversible Data Embedding Using a Difference Expansion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 8, Aug. 2003, pp. 890–896.

[2] Z. Ni et al., "Reversible Data Hiding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 3, Mar. 2006, pp. 354–362.

[3] H.J. Kim et al., "A Novel Difference Expansion Transform for Reversible Data Embedding," *IEEE Trans. Inf. Forensics Security*, vol. 3, no. 3, 2008, pp. 456–465.

[4] V. Sachnev et al., "Reversible Watermark Algorithm Using Sorting and Prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 7, July 2009, pp. 989–999.

[5] C.-Y. Yang and W.-C. Hu, "High-Performance Reversible Data Hiding with Overflow/Underflow Avoidance," *ETRI J.*, vol. 33, no. 4, Aug. 2011, pp. 580–588.

[6] S. Kang, H.J. Hwang, and H.J. Kim, "Reversible Watermark Using an Accurate Predictor and Sorter Based on Payload Balancing," *ETRI J.*, vol. 34, no. 3, June 2012, pp. 410–420.

[7] H.-K. Pan, Y.-Y. Chen, and Y.-C. Tseng, "A Secure Data Hiding Scheme for Two-Color Images," *IEEE Symp. Comput. Commun.*, Antibes, France, July 4–6, 2000, pp. 750–755.

[8] Y.-C. Tseng and H.-K. Pan, "Data Hiding in 2-Color Images," *IEEE Trans. Comput.*, vol. 51, no. 7, July 2002, pp. 873–878.

[9] M. Wu, E. Tang, and B. Lin, "Data Hiding in Digital Binary Image," *IEEE Int. Conf. Multimedia Expo*, New York, USA, Aug. 1–3, 2000, pp. 393–396.

[10] M. Wu, "Multimedia Data Hiding," Ph.D. dissertation, Dept. Electr. Eng., Princeton Univ., Princeton, NJ, USA, Apr. 2001.

[11] M. Wu and B. Liu, "Data Hiding in Binary Images for Authentication and Annotation," *IEEE Trans. Multimedia*, vol. 6, no. 4, Aug. 2004, pp. 528–538.

[12] H. Yang and A.C. Kot, "Data Hiding for Text Document Image Authentication by Connectivity-Preserving," *IEEE Int. Conf. Acoust., Speech, Signal Process.*, Philadelphia, PA, USA, vol. 2, Mar. 18–23, 2005, pp. 505–508.

[13] H. Yang and A.C. Kot, "Pattern-Based Data Hiding for Binary Image Authentication by Connectivity-Preserving," *IEEE Trans. Multimedia*, vol. 9, no. 3, Apr. 2007, pp. 475–486.

[14] H. Yang and A.C. Kot, "Binary Image Authentication with Tampering Localization by Embedding Cryptographic Signature and Block Identifier," *IEEE Signal Process. Lett.*, vol. 13, no. 12, Dec. 2006, pp. 741–744.

[15] Y. Lee, H. Kim, and Y. Park, "A New Data Hiding Scheme for Binary Image Authentication with Small Image Distortion," *Inf. Sci.*, vol. 179, no. 22, Nov. 2009, pp. 3866–3884.

[16] J. Cheng and A.C. Kot, "Objective Distortion Measure for Binary Text Image Based on Edge Line Segment Similarity," *IEEE Trans. Image Process.*, vol. 16, no. 6, June 2007, pp. 1691–1695.

[17] C.-H. Tzeng and W.-H. Tsai, "A New Approach to Authentication of Binary Images for Multimedia Communication with Distortion Reduction and Security Enhancement," *IEEE Commun. Lett.*, vol. 7, no. 9, Sept. 2003, pp. 443–445.

[18] H. Yang, A.C. Kot, and S. Rahardja, "Orthogonal Data Embedding for Binary Images in Morphological Transform Domain: A High-Capacity Approach," *IEEE Trans. Multimedia*, vol. 10, no. 3, Apr. 2008, pp. 339–351.

[19] C.-L. Tsai et al., "Reversible Data Hiding and Lossless Reconstruction of Binary Images Using Pair-Wise Logical Computation Mechanism," *Pattern Recogn.*, vol. 38, no. 11, Nov. 2005, pp. 1993–2006.

[20] Y.-A. Ho et al., "High-Capacity Reversible Data Hiding in Binary Images Using Pattern Substitution," *Comput. Standards Interfaces*, vol. 31, no. 4, June 2009, pp. 787–794.

[21] G.R. Robertson, M.F. Aburdene, and R.J. Kozick, "Differential Block Coding of Bilevel Images," *IEEE Trans. Image Process.*, vol. 5, no. 9, Sept. 1996, pp. 1368–1370.

[22] K. Dong and H.-J. Kim, "An Efficient Pattern Substitution Watermarking Method for Binary Images," *Int. Workshop Digital Watermarking*, Seoul, Rep. of Korea, vol. 6526, Oct. 1–3, 2010, pp. 181–188.

[23] G. Xuan et al., "Reversible Binary Image Data Hiding by Run-Length Histogram Modification," *Int. Conf. Pattern Recogn.*, Tampa, FL, USA, Dec. 8–11, 2008, pp. 1–4.

[24] W.M. Zhang, B. Chen, and N.H. Yu, "Improving Various Reversible Data Hiding Schemes via Optimal Codes for Binary Covers," *IEEE Trans. Image Process.*, vol. 21, no. 6, 2012, pp. 2991–3003.

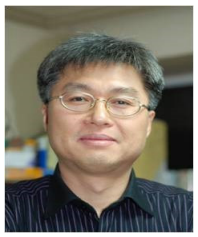
[25] W.M. Zhang, B. Chen, and N.H. Yu, "Capacity-Approaching Codes for Reversible Watermarking," *Int. Conf. Inf. Hiding*, Prague, Czech Republic, May 18–20, 2011, pp. 255–269.

Keming Dong received his BS degree in computer science from the Computer Science and Technology Department, Harbin Institute of Technology, China, in 2008. He joined Multimedia Security Lab., Center of Information Security and Technology, Korea University, Seoul, Rep. of Korea, in 2008, where he is currently pursuing his PhD degree in computer science. His research interests include multimedia security, data hiding, and machine learning.

Hyoung Joong Kim received his BS, MS, and PhD degrees in electrical engineering from Seoul National University, Rep. of Korea, in 1978, 1986, and 1989, respectively. From 1989 to 2006, he worked for the Department of Control and Instrumentation Engineering, Kangwon National University, Chuncheon, Seoul, Rep. of Korea. Since 2006, he has been with the Center of Information Security, Korea University, Seoul, Rep. of Korea, where he is now a professor. His research interests include parallel and distributed computing; multimedia computing; and multimedia security.

Yong Soo Choi received his BS, MS, and PhD degrees in computer science from the Department of Instrumentation and Control Engineering, Kangwon National University, Chuncheon, Rep. of Korea, in 1998, 2000, and 2006, respectively. From 2006 to 2007, he was a research professor with the Center for Technology Fusion in Construction, Yonsei University, Seoul, Rep. of Korea. From 2007 to 2013, he was a research professor with the Brain Korea 21 of Ubiquitous Information Security, Korea University, Seoul, Rep. of Korea. Since 2013, he has been an assistant professor of the Division of Liberal Arts and Teaching, Sungkyul University, Ansan, Rep. of Korea. He was a Delegate of Korea for ISO/IEC JTC1/SC29. He is a member of the IEEK Computer Society and also has been an Editor in Chief of Journal of the Institute of Electronics Engineers of Korea, the Section of Computer and Information. His research interests include multimedia signal processing, digital watermarking, steganography, image forensics and multimedia hashing.

Sang Hyun Joo received his BS and MS degrees in computer science from Dongguk University, Seoul, Rep. of Korea, in 1989 and 1994, respectively. He received his PhD degree in computer science, from Niigata University, Japan, in 1999. From 1994 to 1996, he worked as a research engineer for KaiTech, Seoul, Rep. of Korea. From 1999 to 2001, he worked as a research associate at Niigata University. Since then, he joined ETRI, where he is now a Director in Mobile Content Research Lab. Since 2012, he has worked for MPEG-21 UD (ISO/IEC 21000-22) as both a chairman and an editor. His research interests include media context and control, user description, visual communication technologies.



Byung Ho Chung received his BS, MS, and PhD degrees in computer science from Chungnam National University, Daejeon, Rep. of Korea, in 1988, 2000, and 2005, respectively. From 1988 to 2000, he worked for the Agency for Defense Development, Daejeon, Rep. of Korea, as a research engineer. Since 2000, he has been with ETRI, where he is currently a leader with the ICT Convergence Security Lab. His current research interests include trusted wireless IoT gateways; software-defined cloud security; multimedia-contents analysis and filtering for security; and safety-critical security for embedded devices.

ETRI Journal, Volume 37, Number 5, October 2015
http://dx.doi.org/10.4218/etrij.15.0114.1058