Double Phase Modular Steganography Using Error Images of Lossy Wavelet Transforms

Masoud Afrakhteh and Jeong A. Lee

***Abstract*—An error image of Lossy Wavelet Transform differentiates a typical cover image from its wavelet transformed cover image which is resulted from applying a desired image compression factor (ICF) to the respective cover image. The proposed method uses such error images for finding where and to what extent a target pixel in a cover image must hold secret bits. The method is also able to embed maximal embedding rates on gray-scale cover images, up to four bits per pixel (bpp), preserving a human vision-imperceptible PSNR value and less detectability. The detectability level is shown by Ensemble classifiers using second order SPAM as their input features. Additionally, the method embeds minimal rates, up to one bpp, while being roughly as undetectable as state-of-the-art schemes such as HUGO (highly undetectable stego) and EA (edge adaptive).**

***Index Terms*—Error image, JPEG image compression rates, ensemble classifier, information hiding, steganography.**

I. Introduction

Steganography conceals the very existence of secret information in terms of bits. It usually embeds secret data into some type of digital cover media, such as image, video, audio, or the like. The manipulated image or video looks innocent, and the message cannot be detected with the human eye. On the other hand, exposing the existence of any hidden information in a cover image is what steganalysis does. Such steganalytic algorithms are able to estimate the probable existence of secret bits in different ways. If steganalysis detects hidden information with a minimum probability of error, the steganographic scheme is broken.

There are two factors that have to be considered in designing a modern steganographic scheme, namely embedding rate and undetectability, with a trade-off between them. The higher the embedding rate, the greater is the detectability. Some approaches are more concerned about embedding capacity, with higher imperceptibility levels provided by greater peak signal-to-noise ratio (PSNR) values, but there are many that aim to be more undetectable rather than having higher PSNR values. Least significant bit (LSB) replacement [1] embeds information in the LSB of a pixel, independently of its value. The LSB is directly replaced by the secret bit. This adds some unwanted statistical artifacts, by which the existence of secret bits can be exposed. Such artifacts are paired with values in a histogram of the stego image created by the LSB replacement method. This makes detection easier for a Chi-square attack [2]. LSB matching (LSBM) [3] applies minor changes after LSB replacement, because it randomly increments or decrements the LSB of a pixel according to a pseudo random number generator (PRNG), if the secret bit does not match the pixel's LSB. Unlike LSB replacement and LSBM, which deal with the pixel values independently, LSB matching revisited (LSBMR) [4] modifies the LSBM algorithm in such a way that the choice of incrementing or decrementing the pixel value is no longer random. It performs the operation by using a pair of pixels as a unit. The first pixel value changes in such a way that one secret bit is saved in its LSB, and the second secret bit equals a function of the two modified pixel values. Both LSBM and LSBMR are undetectable with a Chi-square attack because, statistically, the probability of change is the same as that of the increment/decrement performed, either randomly or by using a function. Although the asymmetry artifacts of LSB replacement are almost completely avoided, they can still be detectable using stronger steganalytic attacks. These LSB-based approaches do not consider the difference between the pixel and its neighbors. Edge adaptive (EA) image steganography [5] embeds secret bits based on the LSBMR method. It begins embedding from the edge regions as much as possible, while keeping other smooth areas as they are. The maximum embedding capacity of this approach is limited to one bpp while the visual quality and security of stego images prove to be better than LSB-based and edge-based methods. Another approach is using high-dimensional image models to perform HUGO [6]. This method calculates distortions corresponding to modification of each pixel by ± 1 and sets the stego image pixel value as the minimum of these numbers. The best embedding order starts from pixels with the highest to lowest cost of embedding, which is ascertained by an additive distortion measure. Security is evaluated by training SVM (support vector machine)-based steganalyzers using second-order subtractive pixel adjacency model (SPAM) features [7]. A filter suppresses the stego image content and exposes the added noise in the stego image. Dependencies between neighboring pixels are modeled as a higher-order Markov chain. The resulting sample transition probability matrix is a vector feature that is a SPAM of covers. The second-order Markov chain results in a second SPAM including 686 features for a typical stego image. Finally, the undetectability level of the mentioned methods is benchmarked utilizing the second SPAM as input features to ensemble classifiers [8]. They prove to have better performance compared to SVM-based steganalyzers in terms of both time and accuracy. The classifier has to be trained with a database of pictures to detect information more accurately, so BOSS (break our steganographic system) version 1.01 was used to create sufficiently many stego images. The BOSS database consists

Manuscript received January 23, 2014; revised March 10, 2014.

The authors are with the Computer Systems Laboratory, Chosun University, Korea (e-mail: masoudafrakhteh@gmail.com, jalee@chosun.ac.kr).

of 10,000 8-bit grayscale images of size 512 × 512 pixels.

On the other hand, in classical steganography, multiple base notational system (MBNS) [9] segments the secret bits into several partitions of the same size (32 bits) and applies a modular function to alter the cover image. Thien and Lin [10] also proposed an LSB-based scheme to hide four bpp in a 512 × 512 host image based on a modulus operation. Such classical steganographic schemes can embed up to four bpp, while HUGO and EA cannot. The classical methods are fully detectable by ensemble classifiers; however, they can be less detectable to a Chi-squared attack, a completely outdated steganalysis method.

The current work embeds in the spatial domain owing to the simplicity of the algorithmic nature and ease of mathematical analysis. In addition, spatial-domain techniques can carry the largest messages (embedding rate) compared to transform domains [11], because transformation domain techniques can embed only in nonzero coefficients, whereas all pixels can be utilized in the spatial domain. Modern steganographic schemes are intended to be undetectable, rather than stressing the PSNR value, so the current scheme is also shown to be more undetectable while it can even embed more than one bpp. With classical steganography, the proposed method embeds up to the maximum reported embedding capacity of four bpp [10]. The algorithm has some similarities to the MBNS method, although the bases are calculated in a different way.

This paper is organized as follows. Section II describes the proposed method. The experimental results are compared and evaluated with respect to modern and classical steganographic schemes in Section III. Finally, Section IV highlights and discusses the conclusions, based on the results.

II. Proposed Method

The proposed method pre-processes the cover image sized M × N pixels in order to create an error image. The required error image is computed by applying a suitable image compression factor (ICF), such as the one suggested in Joint Photographic Experts Group (JPEG 2000) compression.

1) Apply a wavelet transform to the cover image. The cover image can be transformed using a JPEG *ICF*, a real value greater than one specifying the target compression ratio, defined as the ratio of input image size to output compressed size. For example, a value of 2.0 implies that the output image size will be half of the input image size or less. A higher value implies a smaller file size and reduced image quality.
2) After the wavelet transform is applied, the matrix of coefficients is scalar-quantized to reduce the number of bits representing them, at the expense of quality. The reduction level factor (RLF) has minimum and maximum values of one and eight, respectively.
3) The compressed image resembles the original image with some imperceptible added noise. The amount of noise has a direct relation to compression level and embedding rate. The larger the embedding rate, the more noise**.**
4) The noise becomes greater if a greater *ICF* is used. In this regard, for every pixel$_{i,j}$ from CoverImg$_{M\times N}$ where i ≤ M and j ≤ N:

$$Error\operatorname{Im}g_{i,j} = Cover\operatorname{Im}g_{i,j} - Compressed\operatorname{Im}g_{i,j} \quad (1)$$

According to the pixel values from the error image, multiple bases are calculated for every corresponding pixel. $Base_{M\times N}$ represents the matrix of multiple bases:

$$Base_{i,j} = \left\lfloor \log_2^{(|Error\operatorname{Im}g_{i,j}|+1)} \right\rfloor + OEF \quad (2)$$

For embedding rates up to one bpp, the optimal extension fields (OEF) factor is equal to zero. It does not increment unless the embedding rate is between two and four bpp. The entire procedure is illustrated in Fig. 1 and described in two phases as below:

$$\underset{\text{Cover Image}}{\begin{array}{cccc} 121 & 124 & 120 & 121 \\ \hline 123 & 123 & 126 & 121 \\ \hline 119 & 136 & 156 & 138 \\ \hline 126 & 130 & 130 & 132 \end{array}} - \underset{\text{Compressed Image using ICF}}{\begin{array}{cccc} 117 & 120 & 125 & 123 \\ \hline 119 & 124 & 128 & 124 \\ \hline 127 & 135 & 141 & 139 \\ \hline 129 & 135 & 138 & 135 \end{array}} = \underset{\text{Error Image}}{\begin{array}{cccc} 4 & 4 & -5 & -2 \\ \hline 4 & -1 & -2 & -3 \\ \hline -8 & 1 & 15 & -1 \\ \hline -3 & -3 & -8 & -3 \end{array}} \rightarrow \underset{\text{Calculated Base Matrix}}{\begin{array}{cccc} 2 & 2 & 2 & 1 \\ \hline 2 & 1 & 1 & 2 \\ \hline 3 & 1 & 4 & 1 \\ \hline 2 & 2 & 3 & 2 \end{array}}$$

Fig. 1. Process of calculating *Base* matrix for 4×4 pixels of a typical cover image (*OEF* = 0).

Phase 1: The secret bits are partitioned into non-overlapping 32-bit blocks. The decimal value *D* of every block is represented using a corresponding pseudorandom number *RND*. Vector D' saves the conversion of D into the base of *RND* and includes a list of integer values. *RND* is computed using a PRNG and the desired embedding rate, which is measured in bpp; *RND* increases for higher embedding rates:

$$RND = \left\lfloor \left(\frac{EmbRate}{8} + \text{PRNG}+1\right)\times 10 \right\rfloor + \left\lfloor \frac{EmbRate}{8}\times 2 \right\rfloor \times 2^{32} \quad (3)$$

Phase 2: The algorithm reads the corresponding *Base* from $Base_{M\times N}$ according to the embedding order, which is line-by-line from top-to-bottom of the cover image. Pixels with a *Base* value less than two are skipped and left the way they are. Thus, only pixels with a base greater than or equal to two may be embedded. Using the decimal values of every element of vector D', one can convert every decimal value into the multiple-base notational system [9]. Thus, vector D'' keeps conversion(s) of every decimal value from vector D' into *n* multiple bases. The embedding is applied in two ways:

Case 1: For every element of vector D'', say $D''_{l\le n}$, greater than or equal to two, directly change the pixel value using the following reversible formulation:

$$Stego\operatorname{Im}g_{i,j} = \left\lfloor \frac{Cover\operatorname{Im}g_{i,j} - D''_l}{Base_{i,j}} \right\rfloor \times Base_{i,j} + D''_l \quad (4)$$

D''_l can easily be extracted by dividing the stego image pixel value into the corresponding *Base* value of the same pixel.

Case 2: For every D_l'' whose value is either zero or one, LSBMR [4] is applied to minimize the embedding effect for two pixel units. Note that, in contrast to the LSB matching concept, there might be some pixels between two such pixels to which LSB matching is not applied.

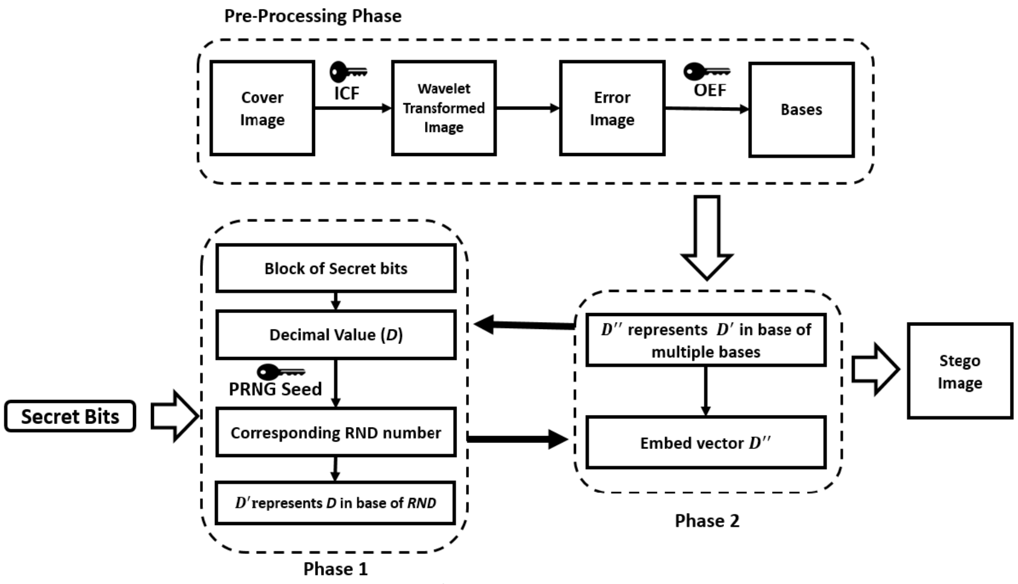

Fig. 2. Structure of the proposed embedding method.

The structure of the proposed scheme is modeled in Fig. 2. Phases 1 and 2 are repeated for every pixel from the cover image. If some of the secret bits have not been embedded yet, *ICF* must be increased towards 10,000. If the secret bits are still partially embedded, the OEF must be incremented until the secret bits are embedded completely. The following algorithm extracts the secret bits from the stego-image:

```
Get Base matrix, Payload, PRNG seed number, ICF and OEF
Set NextBlock to True
Initialize SumRndBases and SumBases to 0
Initialize RndBases and MulBases to 1
Repeat the following for every pixel of the stego image
    If the Corresponding Base value of the current pixel is ≥ 2 then
        If NextBlock is true then
            Calculate RND according to the given payload size and PRNG
            Set NextBlock to False
            Set Bases and RndBases to 1
            Set SumBases and SumRndBases to 0
        Let M save the remainder of the division of the stego pixel value into its corresponding Base value
        If M is either 0 or 1 then
            Make two pixel units
            Set M1 to the LSB of the 1st pixel
            Set M2 to the function of 1st and 2nd pixels as proposed in the LSBMR approach
            Let M be M1 and M2 in order
        Increment SumBases with M × MulBases
        Add MulBases× Base to MulBases
        If SumBases plus MulBases is greater than RND then
            Increment SumRndBases with SumBases× RndBases
            Add RndBases× RND to RndBases
            If SumRndBases plus RndBases is greater than the maximum decimal value of a block of 32 bits then
                Set NextBlock to True
                Show binary form of SumRndBases as part of the secret bits extracted
                Initialize SumRndBases to 0
                Initialize RndBases to 1
            Else
                Initialize SumBases to 0
                Initialize MulBases to 1
Until payload is extracted completely
```

TABLE I: A NUMERIC ILLUSTRATION OF EMBEDDING AND EXTRACTING.

Initialization	1st 32-bit block		**0...00000001101001**		1st block Corresponding *RND* Number: 100		
	Decimal Value		**105** = $(15)_{100}$ = 5 + 1 ×100				
Phase 1	$D' = \{5,1\}$			5 =$(12)_{4,3}$ = 2+ 1 ×3			1 = $(1)_2$ =1
Phase 2	$D_l'' = \{2,1,1\}$			2		1	1
Pre-Processing Phase	Cover Image		80	87	101	157	165
	Base numbers		0	3	1	4	2
Embed If $Base \geq 2$	Stego Image	if Case 1: $D_l'' \geq 2$	No Embedding	86	No Embedding	-	-
		if Case 2: $D_l'' < 2$	No Embedding	-	No Embedding	157	165
Extraction	If Case 1:Modular function		No Extracting	Mod(86,3)=**2**	No Extracting	-	-
	If Case 2:LSBMR[4]		No Extracting	-	No Extracting	LSB(157)=**1**	LSB($\left\lfloor\frac{157}{2}\right\rfloor$+165)=**1**

The embedding procedure is illustrated in Table I. The decimal value of the first 32-bit block is calculated as 105 and converted into the base of *RND* 100, which is $(15)_{100}$ equal to $5 + 1 \times 100$. Numbers 5 and 1 are again converted using those pixels with their corresponding *Base* greater than or equal to 2. Thus, 5 is represented as $(12)_{4,3}$. Numbers 1 and 2 have to be embedded in two different ways, as stated in Phase 2. The extraction is also performed in two different ways. Finally, the extracted values are shown in boldface. Two pixels with grayscale values 80 and 101 are excluded, because their corresponding *Base* values are zero and one, respectively. Pixels with values of 157 and 165 have formed a pair of pixel units to be given to LSBMR [4] for embedding (one in both of them). In this case, two pixels are spared from being changed, yet the message bits are still extractable. According to the LSBMR algorithm, in the worst case, only one of the two pixels is incremented or decremented.

III. Experiment and Results

One of the most important aspects of any performance evaluation is to use a standard data set with a variety of image textures. The proposed scheme employs the image database of BOSS version 1.01, which consists of 10,000 grayscale images sized 512×512 pixels. This database is used also in modern steganographic schemes with embedding rates less than or equal to one bpp.

TABLE II: Detectability Comparison between the Proposed Method (DPMS-E) and HUGO and EA Approaches.

BOSS 1.01 database		Average testing error over 10 splits (p_e) using 2nd SPAM features		
Payload (bpp)	Capacity (bits)	EA [5]	HUGO [6]	DPMS-EW (PSNR, ICF, OEF)
0.05	13,101	0.4717	0.5000	0.4417 (66.56 dB, 4.1 , 0)
0.1	26,214	0.4309	0.4844	0.4162 (63.54 dB, 4.6, 0)
0.2	52,428	0.3381	0.4469	0.3654 (60.49 dB, 5.6, 0)
0.3	78,643	0.2549	0.4010	0.3137 (58.83 dB, 6.8, 0)
0.4	104,857	0.1920	0.3600	0.2358 (57.58 dB, 8.5, 0)

Modern steganography is more concerned with undetectability levels than imperceptibility, so the PSNR value is always supposed to be high. In this regard, HUGO and EA are modern steganography methods in which the LSB matching concept is applied to manipulate the LSB of the pixels. The undetectability level is shown using an error probability provided by ensemble classifiers using second-order SPAM features. A decreasing error probability indicates a greater chance of detection. HUGO has been proven to have the greatest error probability compared to the EA and LSB-matching methods [6]. As depicted in Table II, our proposed method, Double Phase Modular Steganography with the help of the Wavelet transform Error images (DPMS-WE), contributes more than EA in terms of lesser detectability (higher average testing error) and is almost as undetectable as HUGO. On the other hand, DPMS-WE is able to embed up to four bpp, whereas HUGO and EA can embed only up to one bpp. Furthermore, the complexity of DPMS-WE code is similar to that of classical schemes and can be implemented more easily.

TABLE III: A Comparison to the Classical Steganographic Schemes in Terms of Achieved PSNR Value.

Payload (bpp)	Capacity (bits)	Cover Image	Embedding Method	PSNR (dB)
2.82	740,000	*Man*	MBNS [9]	38.10
			DPMS-EW (ICF=10000, OEF=13)	34.76
4.0	1,048,576	*Lake*	Thien and Lin [10]	34.80
			DPMS-EW (ICF=10000, OEF=13)	34.30

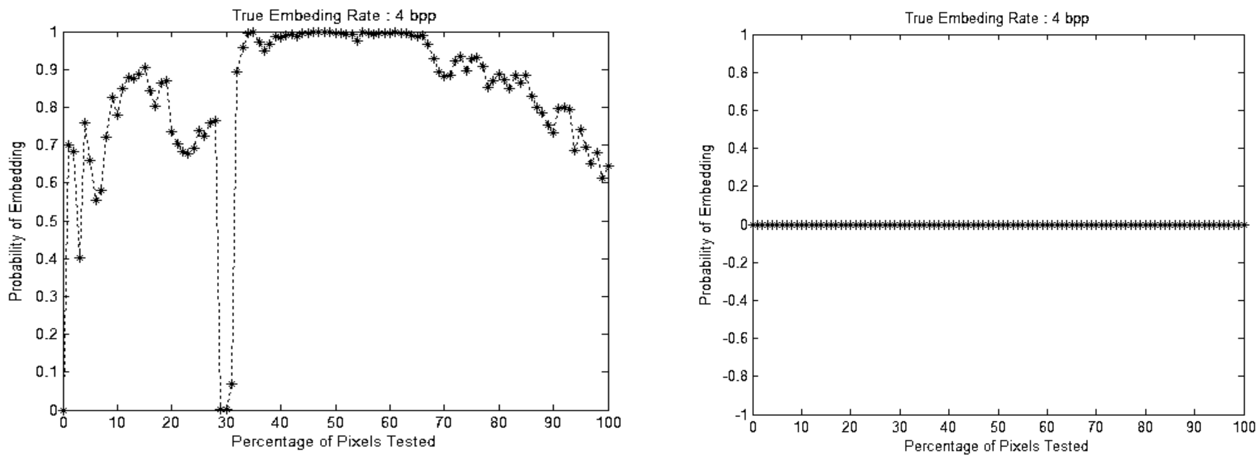

Fig. 3. Left, Chi-squared attack applied to Thien and Lin [10] and, right, the attack applied to DPMS-EW.

As shown in Table III, our proposed method was applied to the same images and embedded with the same number of bits as for classical steganography. It is shown that the proposed method performs better compared to other classical steganographic schemes also in terms of detectability level. The current scheme was completely undetectable by

Chi-squared attack. On the other hand, while Thien and Lin have a greater PSNR value, their method was detectable with a Chi - square attack which is completely outdated. When we embedded up to four bpp using the Thien method, the resulted stego-image PSNR value decreased to 34.80 dB. In addition, as shown in Fig. 3, we can see the fluctuated line implying that roughly 80% of the pixels are detected as changed with a probability of one, while this value equals zero (right diagram- a horizontal line at the zero level) for 100% of the pixels and an embedding rate of 4 bpp. Using our scheme, the PSNR level is smaller than Thien and MBNS, 34.76 dB and 34.30 dB, showing that the PSNR value does not necessarily prove lower detectability.

Finally, Fig. 4 shows the last grayscale image out of 10,000 images provided by the BOSS version 1.01 image database. In a simple experiment, we attempted to prove that higher ICF values would result in more undetectability of the stego image. Theoretically, PSNR values will be smaller if we employ a higher ICF. Note that the proposed method introduces two error images. One is computed in Phase 1 of the algorithm. As with either of Fig. 5 or Fig. 6, another error image is calculated between the stego image and the original image to show where, and to what extent, the impact of embedding has occurred. In Fig. 5, the experiment of embedding a payload of 0.4 bpp using an ICF of 7.3 is conducted. As can be seen, the secret bits are scattered throughout the cover image (PSNR 57.51 dB), and we can hardly see the detailed texture of the image (Fig. 4). On the other hand, Fig. 6 represents the detail of the image texture when the secret bits are not well scattered but lumped in the upper area of the error image (PSNR 57.18 dB, ICF 44).

As in Fig. 5 and Fig. 6, the imperceptibility level is shown by the PSNR values, and the second experiment shows a smaller PSNR. Therefore, it is considered more visible, such that the detected error probability is also less than the one with ICF 7.3, because ICF 44 has bigger computed *Base* values, so the secret bits can be embedded using fewer pixels, while the change is much bigger. ICF 7.3 has smaller *Base* values, and the secret bits are depicted using more pixels, while the change is well distributed among more pixels and also much smaller. Feature extraction methods are more sensitive to the busy areas, and the stego image will be broken as soon as it embeds more in busy regions. ICF 7.3 tries to embed in smooth areas, whereas ICF 44 is more concentrated on busy areas, because smooth areas cannot tolerate more pixel value alteration, so edge areas have to hold more secret bits. Owing to this fact, the pattern of the 10,000th image matrix of calculated Bases has become clearer in Fig. 6 compared to the one in Fig. 5. This is revealed in the error image, because the texture is clearer when using ICF 44.

Fig. 4. The last cover image from BOSS 1.01 called “10000.pgm”.

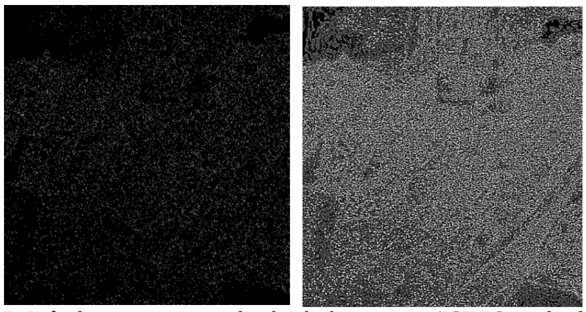

Fig. 5. Left, the error image, and right, the base matrix (ICF 7.3, payload 0.4 bpp, 57. 51 dB).

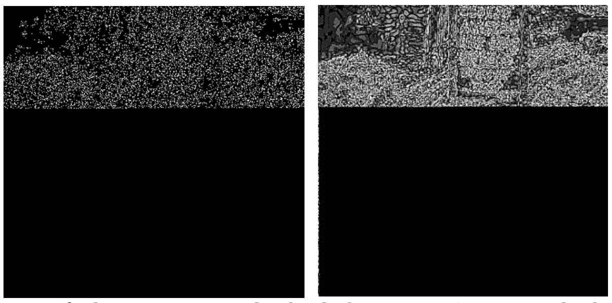

Fig. 6. Left, the error image, and right, the base matrix (ICF 44, Payload 0.4 bpp, 57. 18 dB).

Table IV benchmarks the experiment using 10,000 images from the BOSS database. It is clear that the experiment “2” is less detectable due to the greater error values and a complete scattering of the secret bits. Greater ICF values increase the probability of being detected by ensemble classifiers with less average testing error (P_e).

TABLE IV: DPMS-E EXPERIMENT RESULTS WITH DIFFERENT ICFS FOR THE SAME PAYLOAD.

BOSS 1.01 database		Experiment 1		Input Parameters		Experiment 2		Input Parameters	
Payload (bpp)	Capacity (bits)	P_e	PSNR (dB)	**ICF**	OEF	P_e	PSNR (dB)	**ICF**	OEF
0.05	13,101	0.4066	66.59	9.0	0	0.4417	66.56	4.1	0
0.1	26,214	0.3639	63.54	10.0	0	0.4162	63.54	4.6	0
0.2	52,428	0.3044	60.56	11.0	0	0.3654	60.49	5.6	0
0.3	78,643	0.2479	58.79	12.0	0	0.3137	58.83	6.8	0
0.4	104,857	**0.1884**	57.52	**13.0**	0	**0.2358**	57.58	**8.5**	0

IV. Conclusion

The proposed method showed how steganalysis would be affected by the JPEG2000 image compression factor and JPEG2000 decompression artifacts. They provide more confusion to steganalysis schemes. It was also proven that a smaller ICF number guarantees less detectability compared to the EA method and a detectability level close to HUGO’s, because the algorithm distributes the secret bits through the cover image more evenly and selects the right pixels to hold secret bits using a calculated *Base* matrix as a guide.

On the contrary, LSB matching-based methods, EA and HUGO embed at one bpp, whereas this scheme can embed at four bpp. However, the proposed scheme requires a *Base* matrix to extract the secret information successfully. Hence, there should be a solution to get the same *Base* matrix calculated from the stego image so that the existence of a cover image is not necessary. This can be investigated in future work.

It is worth mentioning that the current scheme can be slightly modified and applied easily in parallel processing languages, in particular, CUDA NVIDIA programming, because each block can be embedded independently from the other blocks. Further, the number of secret bits can be predicted according to the error image, which is computed using a wavelet transformed version of the cover image. This can also be investigated in future work.

REFERENCES

[1] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM Systems Journal*, vol. 35, no. 3-4, pp. 313-336, 1996.

[2] A. Westfeld and A. Pfitzmann, "Attacks on steganographic systems," *Information Hiding*, vol. 1768, pp. 61-76, January 2000.

[3] A. D. Ker, "Improved detection of LSB steganography in grayscale images," *Information Hiding*, vol. 3200, pp. 97-115, January 2005.

[4] J. Mielikainen, "LSB matching revisited," *Signal Processing Letters*, IEEE, vol. 13, no. 5, pp. 285-287, 2006.

[5] W. Luo, F. Huang, and J. Huang, "Edge adaptive image steganography based on LSB matching revisited," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 201-214, 2010.

[6] P. Bas, T. Filler, and T. Pevný, "Break our steganographic system: the ins and outs of organizing BOSS," *Information Hiding*, vol. 6958, pp. 59-70, January 2011.

[7] T. Pevny, P. Bas, and J. Fridrich, "Steganalysis by subtractive pixel adjacency matrix," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 215-224, 2010.

[8] J. Kodovský, J. Fridrich, and V. Holub, "Ensemble classifiers for steganalysis of digital media," *IEEE Transactions on Information* Forensics *and Security*, vol. 7, no. 2, pp. 432-444, April 2012.

[9] X. Zhang and S. Wang, "Steganography using multiple-base notational system and human vision sensitivity," *Signal Processing Letters*, IEEE, vol. 12, no.1, pp. 67-70, 2005.

[10] C. C. Thien and J. C. Lin, "A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function," *Pattern Recognition*, vol. 36, no. 12, pp. 2875-2881, 2003.

[11] A. Cheddad, J. Condell, K. Curran, and P. M. Kevitt, "Digital image steganography: Survey and analysis of current methods," *Signal Processing*, vol. 90, no. 3, pp. 727-752, 2010.

Masoud Afrakhteh completed his bachelor studies in software engineering from Islamic Azad University of Mashhad in Iran. From 2009 January to 2010 March, he did his master in computer science at UTM University of Malaysia. From 2010 April to 2010 September, he was a research assistant with the Department of Computer Science and Engineering, UTM university, Malaysia. Since 2010 September, he has been a Korean Government Scholarship Program PhD scholar at Chosun University in South Korea. His field of interest is cyber security, particularly using image information embedding and retrieval.

Jeong A. Lee received Ph.D in computer science from UCLA in 1990. From 1990 to 1995, she was an assistant professor with the Department of Electrical and Computer Engineering, University of Houston, Texas. Since 1995 she has been a professor and the head of Computer Systems Laboratory, Chosun University in Korea. From 2008 to 2009, she served as the director of Electrical and Computer Science and Engineering, National Research Foundation of Korea. Her research activities cover high-performance computer architecture, fast digital and CORDIC arithmetic, configurable computing, fault-tolerant computing, and computer security. She is the author and coauthor of more than 100 reviewed journal and conference papers. Professor Lee is a member of the National Academy of Engineering, Korea and an IEEE senior member.