JJEE
Jordan Journal of Electrical Engineering
Volume 8, Number 1, March 2022
Pages 90-101
ISSN (Print): 2409-9600, ISSN (Online): 2409-9619

An Optimized Fragile Image Watermarking Method for Tamper Detection and Recovery Using SPIHT and Reed-Solomon Coding

Mahmoud Alnaanah[1*], Moath Alsafasfeh[2], Ahmad Aljaafreh[3], Amir Abu-Al-Aish[4]

[1] Department of Electrical Engineering, Al-Hussein Bin Talal University, Maan, Jordan
E-mail: mahmoud.alnaanah@ahu.edu.jo
[2] Department of Computer Engineering, Al-Hussein Bin Talal University, Maan, Jordan
[3] Department of Computer and Communications Engineering, Tafila Technical University, Tafila, Jordan
[4] Department of Communications Engineering, Al-Hussein Bin Talal University, Maan, Jordan

Received: September 05, 2021 *Revised: October 02, 2021* *Accepted: October 09, 2021*

***Abstract*—** Tamper detection and recovery have been successfully modeled as a source-channel coding problem for an erasure channel. Set partitioning in hierarchical trees (SPIHT) and Reed Solomon (RS) codes have been proven very effective for source and channel coding. This paper presents an optimized tamper detection and recovery method that provides high recovery quality, high tolerable tampering ratio (TTR) and optimized speed. The proposed method compresses each block of the image separately - instead of compressing the whole image - which eliminates the need to store the SPIHT stream in the watermark, and provides a larger space for RS parity information. The proposed method also optimizes the speed of RS encoder and decoder by minimizing the code symbol size. The RS code message is composed by taking one symbol at a time from each block, which leads to reducing the encoding time dramatically. The obtained experimental results show a competing recovery quality of the proposed method while having high TTR.

***Keywords*—** Image watermarking; Tamper detection; Tamper recovery; SPIHT; Reed Solomon code.

1. INTRODUCTION

With the advance of digital image editing applications, image tampering becomes easier than ever. For images that have sensitive information, as in forensics and medical applications, tampering is very tempting and some measures are necessary to detect, and possibly recover, any tampering. Many methods have been proposed for image authentication [1, 2]. Some of these methods rely on finding a signature for the image, which is stored separately from the image itself. Signature-based methods are incapable of detecting the exact location of tampering or recovering it. Watermarking has been proven as an excellent solution for tamper detection and recovery problem [3–5]. The watermark is embedded inside the image itself, which means it will not be lost due to transmission problems or due to change in the image format.

In general, image watermarking for tamper detection and recovery starts by partitioning the image into non-overlapping blocks, then reference information is generated by compressing each block, which minimizes the required space for reference information. A hash is generated for each block to detect any tampering. In fragile watermarking, the watermark is stored in the least significant bits (LSBs) of the image. The reference information is used to recover the tampered blocks. To minimize tamper coincidence problem - where the block and its related reference information are both lost due to tampering - the reference

* Corresponding author
Article's DOI: 10.5455/jjee.204-1630865748

information is usually stored in a block that has maximum-distance from the one to which the reference information corresponds. Also, multiple copies of the reference information could be stored in the watermark to increase the probability of recovery. Maximum-distance approach might help when the tampering occurs in an isolated area; however, it makes the recovery sensitive to the tampering pattern, even if the tampered area is small. Having multiple copies of reference information increases the watermark-wasting problem. One solution for tamper coincidence problem is to perform channel-coding for reference information and store the generated parity bits instead of the reference bits [6–8]. The main advantage of using channel-coding is the efficient use of the available watermark space while maintaining a tolerable tampering ratio (TTR) independent of the tampering pattern.

Sarreshtedari and Akhaee proposed an efficient method that relies on source-coding the whole image using set partitioning in hierarchical trees (SPIHT) algorithm and using Reed Solomon (RS) code for channel coding [8]. Source-coding the whole image gave their method the advantage of having high quality recovery using smaller bit per pixel (bpp), especially because the image is source-encoded before deleting its LSBs. However, the SPIHT stream needs to be stored in the watermark, which limits the space available for parity information and results in limited TTR. Sarreshtedari's method has another disadvantage of using large symbol size for RS code, which results in increasing coding/decoding time. This paper presents an optimized tamper detection and recovery method that tries to solve the problems with Sarreshtedari's method and provides high recovery quality with high TTR.

One important aspect of any tamper detection and recovery method is the block size into which the image is partitioned. Using larger blocks has more advantages over using smaller ones. One of them is having better recovery quality, because a larger block usually has more redundancy, which is approximated more effectively. To illustrate this, a 512×512 grayscale version of the famous Lena image [9] is partitioned into blocks with the sizes shown in Table 1. Each block is compressed using SPIHT algorithm [10] with 1 bpp (17 ×17 is the minimum permissible block size by the used SPIHT algorithm [11]). Table 1 shows the peak signal to noise ratio (PSNR) between the compressed and the original Lena images. It also exhibits how increasing the block size increases the compressed image quality for the same bpp. It is also noticeable that the compression quality decreases dramatically as the block size decreases, and; therefore, block sizes like 4×4, 2×2, or even 1×1, (as when halftoning is used [12, 13]) result in a highly degraded image quality.

Table 1. PSNR for Lena image when partitioned into blocks and each block is compressed using SPIHT algorithm with 1 bpp.

Block size [Pixels]	17×17	32×32	64×64	128×128	256×256	512×512
PSNR [dB]	28.49	35.73	37.20	38.38	39.17	39.83

Another advantage of using larger blocks is reducing the probability of false-negative, i.e., tampered blocks that are falsely identified as untampered ones. A percentage of the watermark is assigned as a hash for tamper detection, and as the block size increases, the

watermark size increases, which provides more bits for the hash. Notice that $P_f = 2^{-n_H}$, where P_f is the probability of false-negative and n_H is the number of bits in the hash.

The main disadvantage of using a larger block size is that more untampered pixels are included in the tampered blocks, which makes the detection coarser. However, with the advance of digital cameras, image size grows rapidly and therefore a larger block size is not a problem for images with large sizes. When each block of the encoded image is secured independently - using the same secret key - makes the encoded image vulnerable to collage attack [14], where blocks from an image are used to tamper with other blocks in the same image or a different one that is secured with same secret key. The solution for collage attack is to make each block secured using a unique secret key, which can be achieved by combining the main secret key with the block raster order and a unique serial number for each image. In the proposed method, the solution for tamper detection and recovery problem is optimized by taking the following points into account:

- Using a large block size: this improves the recovery quality and reduces the false-negative probability, as mentioned before. Another advantage of using larger blocks is reducing their total number, which reduces the symbol size for RS code (as explained in section 3.1), and that reduces the time required for encoding and decoding [15].
- Using SPIHT algorithm for source-coding the image blocks: SPIHT compression algorithm has many advantages besides providing a high compression ratio. One of them is progressive encoding, which gives the ability to truncate the compressed stream at any point and still be able to decode the image but with a lower quality. This fits tamper recovery problem as the space available for the watermark might vary depending on the assigned LSBs for the watermark and the number of bits in the hash.
- Using RS coding: this has an advantage over other erasure channel codes (such as fountain codes [7, 16]) of being optimal when it comes to guaranteed TTR, and this makes it fit the tamper recovery problem where the available space is limited.
- Optimizing RS coder speed: the coding complexity of RS code increases in an exponential manner as the symbol size increases [17, 18]; and therefore, dividing the message information into smaller pieces will reduce the coding time dramatically. In the proposed method, the encoded message is composed by taking one symbol from each block; this will optimize symbol size and reduce the coding time dramatically while maintaining the same TTR. More details are presented in section 3.
- Enhancing the security measures: to make the proposed method robust against attacks, especially collage attack, the bits of the watermark are XORed and permuted using a unique secret key for each block. The uniqueness of the secret key is guaranteed by combining the original key with the raster order of the block and a unique serial number for each image.

2. RELATED WORK

Korus and Dziech [7] modeled tamper detection and recovery as an erasure communication channel. In their method, the image is divided into non-overlapping 8×8 blocks, and three LSBs in each block are assigned for the watermark. Five most significant bits (MSBs) in each block are compressed by quantizing their discrete cosine transform (DCT) coefficients into K bits. The DCT information is then channel-encoded using random

linear fountain (RLF) codes to produce $N = 160$ bits code, which in addition to 32 bits message digest (MD5) hash makes the watermark. The ratio $\lambda = K/N$ is called the code rate, and it controls the recovery quality and the TTR. Two code rates were used; the first one is $\lambda = 1$, which produces a lower recovery quality with a higher TTR of 50%. The second is $\lambda = 2$, which provides a higher recovery quality with a lower TTR of 33%. Korus's method recovery quality is limited because of using a small block size.

Sarreshtedari and Akhaee [8] proposed a method where the whole original image is source-coded, i.e., compressed using SPIHT algorithm with a rate of 1 bpp. The source-coded version of the image is then channel-coded using RS code. The original image is divided into non-overlapping 8×8 blocks, and 2 or 3 LSBs are assigned for the watermark, which consists of a portion of the RS codeword along with 32 bits MD5 hash. The TTR is 33% and 60% when 2 and 3 LSBs are used for the watermark, respectively. In the decoding stage, the tampered blocks are detected using the hash bits, then the input of the RS decoder is collected from all untampered blocks. The SPIHT decoder information is recovered and passed to the SPIHT decoder to get the original image, which is used to recover the tampered blocks. Since the whole image is compressed using SPIHT algorithm and even before deleting its LSBs; this provides better recovery quality. However, not only the added parity information has to be stored in the watermark, but also the SPIHT stream. This could be avoided if the blocks of the image are source-coded individually instead of compressing the whole image. Sarreshtedari's method also suffers from security vulnerabilities [19], such as the inability to detect malicious changes applied to the RS code bits in the watermark, as the hash is generated only for the MSBs of the image block.

In [18], Sarreshtedari et al. proposed an enhancement of the previously proposed method in [8]. The enhancement is based on joint source–channel coding (JSCC) by providing unequal error protection for the bit planes of the encoded SPIHT stream using dynamic programming optimization approach. The method in [18] is further improved by Gu et al. [20] by utilizing quadtree decomposition in the bit-planes of SPIHT stream. The proposed methods in [18] and [20] provide higher TTR with a decent recovery quality at lower tampering ratio. However, the proposed methods require sending the optimization parameters along with the encoded image, which increases the complexity of exchanging the encoded images. Additionally, the recovery quality at higher tampering ratios depends on the expected tampering ratio and can have values lower than the ones provided by the proposed method in this paper.

In [19], Fan and Wang highlighted some security issues of Sarreshtedari's method [8], such as the inability to detect tampering that affects only the RS code in the watermark because the hash is generated only for MSBs of the image blocks. Fan and Wang also proposed a fragile tamper detection and recovery method. In their method, SPIHT coding (with 0.75 bpp) is applied on the image blocks instead of the whole image, and for channel-coding, they used repeated coding instead of RS code. The use of repeated coding instead of RS coding increases the watermark-wasting problem and makes their method sensitive to tampering pattern.

Sarreshtedari et al. proposed a method for tamper detection and recovery in JPEG image format [21]. The proposed method tolerates JPEG recompression and noise attacks, whilst fragile watermarking is vulnerable against these types of attacks. In their method,

Sarreshtedari et al. modeled the tampering detection and recovery as a source-channel coding problem. SPIHT is used for source coding, and low-density parity check (LDPC) is used for channel coding. Hash bits are not used, and instead, a direct comparison between the recovered and the tampered images is used for tamper localization. Successful recovery requires that the tampering is within the correcting capability of the LDPC code; otherwise, the recovered image will be a random pattern. After quantization, the channel-coded information is embedded in the DCT domain. The tampering rate must not exceed the correction threshold of the LDPC code, which is 8.6%.

3. DESCRIPTION OF THE PROPOSED METHOD

The encoding and decoding stages of the proposed method are described in this section, and they are illustrated in Figs. 1 and 2, respectively. An example input image is used for illustration, the image size is 512×512 pixels, the block size is 32×32 pixels, and 2 LSBs are used to store the watermark.

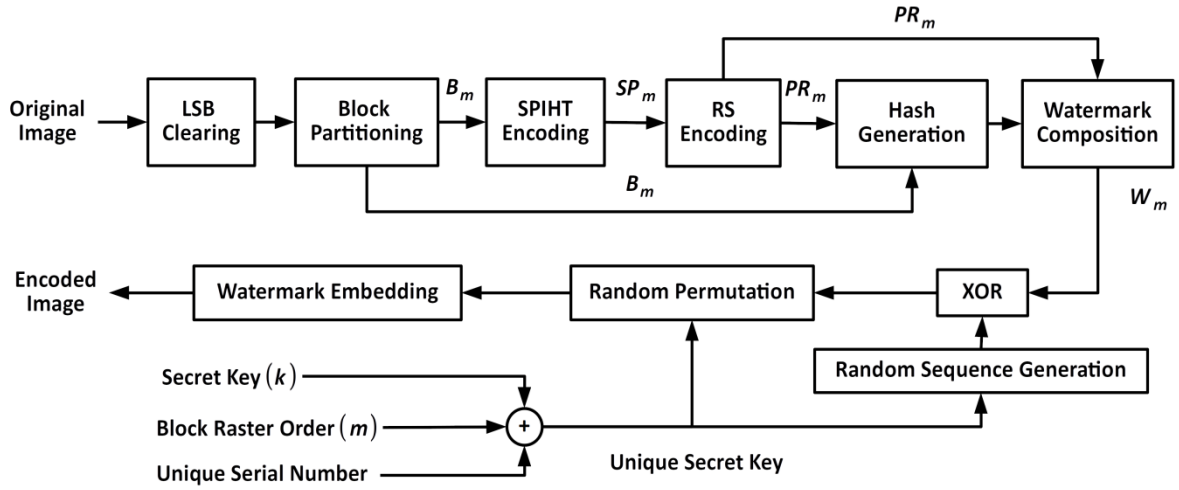

Fig. 1. Block diagram for the encoding stage of the proposed method.

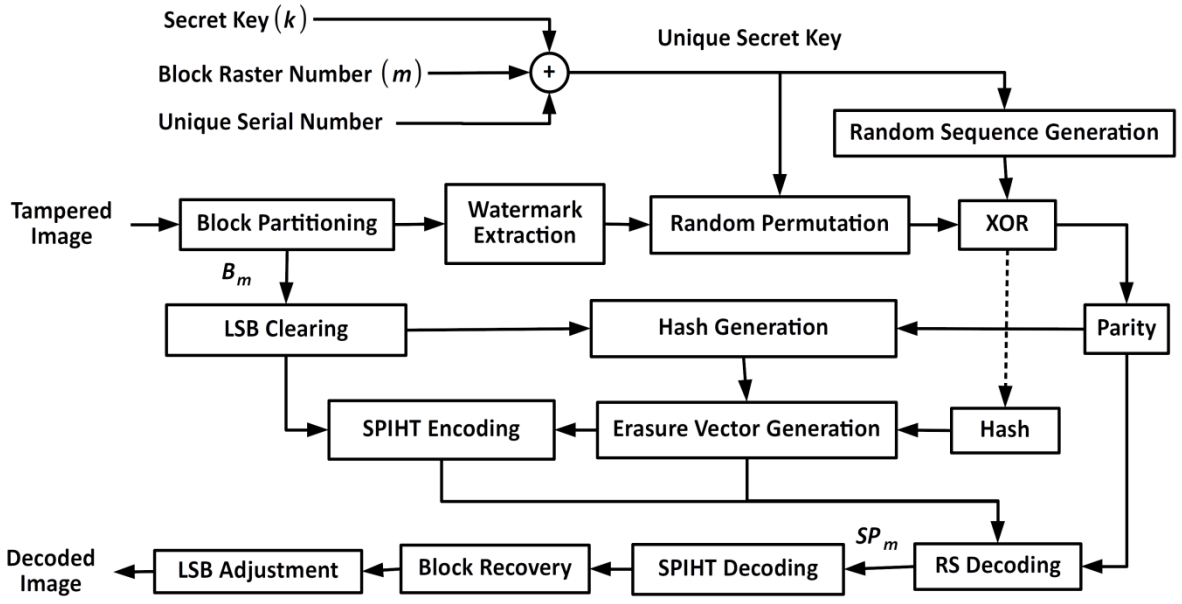

Fig. 2. Block diagram for the decoding stage of the proposed method.

3.1. The Encoding Stage

The encoding stage starts with the original grayscale image I_O that has a size of $d_X \times d_Y$ pixels, where d_X and d_Y are its width and height, respectively. For I_O, the LSBs assigned to the watermark are cleared by storing zeros in them. The resulting image I_C is then divided into non overlapping blocks B_{ij}, where i and j are the vertical and the horizontal indices of each block. The raster order is indicated by m as in B_m for block number m. The size of each block is $d_B \times d_B$ pixels (e.g., 32×32 pixels), d_X and d_Y are supposed to be multiples of d_B. The total number of blocks in the image is $N_B = \frac{d_X}{d_B} \times \frac{d_Y}{d_B}$. The watermark W_m stored in each block has a total number of bits $n_W = d_B \times d_B \times n_L$, where n_L is the number of LSBs assigned for the watermark (e.g., 2 bits). For the example image, $n_W = 32 \times 32 \times 2 = 2048$ bits.

The next step is to assign a number of bits for the parity of the RS code in each block (denoted by n_R), and leave the rest of the bits in the watermark for the hash (denoted by n_H.) The RS parity bits n_R are supposed to be multiple of $(R-1) \times n_S$, where n_S is the symbol size for RS code in bits and R is the reciprocal of the RS code-rate, or:

$$R = \frac{Message\ Length + Parity\ Length}{Message\ Length} \tag{1}$$

R determines the TTR for the method (as will be illustrated in the following paragraphs). The possible values for n_H are given by:

$$n_H = mod(n_W, (R-1) \times n_S) + h \times (R-1) \times n_S \tag{2}$$

where $h = 0, 1, 2, 3, \ldots$, and mod is the modulo operation. For the example image, if R is selected to be 3 and n_S is 10 bits, then the possible values for n_H are $mod(2048, (3-1) \times 10) + h \times (3-1) \times 10 = 8, 28, 48, 68, \ldots$, n_H is selected to be 28 bits.

The codeword length for RS code is $N_M + N_R$, where N_M symbols are assigned for the message and N_R symbols are assigned for the parity. The message symbols are equal to the number of blocks, and the parity symbols are equal to the message symbols times $(R-1)$, or:

$$\begin{aligned} n_R &= (R-1) \times N_M \\ &= (R-1) \times N_B \end{aligned} \tag{3}$$

The symbol size n_S for RS code is calculated as:

$$n_S = floor(log_2(R \times n_B)) + 1 \tag{4}$$

For the example image, $N_B = \frac{512}{32} \times \frac{512}{32} = 2$ and if $R = 3$ then $n_S = floor(\log_2(3 \times 256)) + 1 = 10$ bits. Using 10 bits for RS code symbol will result in a codeword length of $2^{n_S} - 1$ or 1023 symbol. Since the available number of parity symbols (in this example $1023 - 256 = 767$) is larger than the needed number (in this example 512), puncturing is used to get rid of the extra unneeded parity symbols. The number of punctures is:

$$N_C = 2^{n_S} - R \times N_B - 1 \tag{5}$$

For the example image, $N_C = 2^{10} - 3 \times 256 - 1 = 255$.

The number of erased symbols N_E plus the number of punctures N_C must not exceed the correcting capability of RS code, which equals $2t$, where:

$$\begin{aligned} t &= floor((2^{n_S} - N_M - 1)/2) \\ &= floor((2^{n_S} - N_B - 1)/2) \end{aligned} \tag{6}$$

Therefore:

$$N_E + N_C = 2\ floor\ ((2^{n_S} - N_B - 1)/2) \quad (7)$$

or

$$N_E = 2\ floor\ ((2^{n_S} - N_B - 1)/2) - N_C$$
$$= 2\ floor\ ((2^{n_S} - N_B - 1)/2) - 2^{n_S} - R \times N_B - 1 \quad (8)$$

Since 2^{n_S} is an even number, then:

$$N_E = 2\ floor\ ((-N_B - 1)/2) + R \times N_B + 1 \quad (9)$$

If N_B is an odd number, then:

$$2\ floor\ ((-N_B - 1)/2) = -N_B - 1 \quad (10)$$

or

$$N_E = -N_B - 1 + R \times N_B + 1$$
$$= (R - 1) \times N_B \quad (11)$$

If N_B is an even number, then:

$$2\ floor\ ((-N_B - 1)/2) = -N_B - 2 \quad (12)$$

or

$$N_E = -N_B - 2 + R \times N_B + 1$$
$$= (R - 1) \times N_B - 1 \quad (13)$$

Each tampered block will result in R erased symbols (the message symbol generated from the block and the parity symbols stored in the block), or $N_E = R \times N_T$ where N_T is the number of tampered blocks. Since N_T is an integer, then:

$$N_T = floor\left(\frac{(R - 1) \times N_B - (1 - mod(N_B, 2))}{R}\right) \quad (14)$$

For large N_B, $N_T \approx \frac{R-1}{R} N_B$, or:

$$TTR \approx \frac{R - 1}{R} \quad (15)$$

$TTR = 50\%, 66.6\%, 75\%, and\ 80\%$ for $R = 2, 3, 4, and\ 5$.

After dividing the image I_C into blocks, each block is compressed using SPIHT algorithm. The number of bits assigned for the compressed stream is $n_M = n_R/(R - 1)$. For the example image, the number of bits assigned for the SPIHT stream in each block is $nR/(R - 1) = (n_W - n_H)/(R - 1) = (2048 - 28)/(3 - 1) = 1010$ bits, which results in a rate of 0.986 bpp. Let SP_m represents the SPIHT stream for the block B_m. After compressing all of the blocks, the input message for the RS encoder is composed by taking one symbol from each SP_m stream, and it is then encoded. The RS encoding process continues for the remaining symbols in SP_m. The parity symbols resulting from the RS encoding process are appended to form PR, which is the parity information that will be stored in the watermark W_m. The length of PR is $(R - 1) \times N_B$; therefore, the parity to be stored in W_m is the rows in PR at positions $m, m + N_B, \dots, m + (R - 1) \times N_B$. The remaining hash bits n_H are calculated in the next step.

The hash bits are found for the concatenation of the MSBs of each block along with the parity symbols. MD5 hash is selected with the output as binary numbers instead of printable characters to fully utilize the available hash bits. The probability of false-negative equals to 2^{-n_H}. The number of bits in the MD5 hash is larger than n_H; therefore, only the leftmost n_H

bits are taken from the MD5 hash. The parity bits and hash bits are then appended to produce the watermark, which is secured in the next step.

Two steps are taken to secure the watermark; the first one is XORing the watermark with a random sequence, and the second one is randomly permuting the bits of the watermark. The permutation of the watermark bits is necessary to prevent the random XOR sequence from being known by any intruder, which could be done by independently generating the watermark bits and XORing them with the stored one. The randomization seed for the permutation and the random sequence generation is determined using a combination of a secret key k, the raster order of the block and a unique serial number for the image. In this way, each block is secured with a unique number and cannot be used to tamper with another block using collage attack. The watermark is then embedded in the designated LSBs of each block, and the encoded image I_E is now ready.

3.2. The Decoding Stage

The decoding starts with the image I_T , which is an encoded image that possibly has been tampered with. The watermark stored in each block is extracted and inversely permuted, then it is XORed with the same random sequence that has been used in the encoding stage. The randomization seed for permutation and random sequence generation is determined as in the encoding stage.

For each block, an MD5 hash is generated for the MSBs and the stored parity symbols, then the leftmost n_H bits of the generated hash are compared with the stored hash bits. If they are different, then the block is marked as tampered. If the number of tampered blocks exceeds the recovery capability, then the tampered blocks cannot be recovered; otherwise, they are fully recoverable. The positions of the erasures in the message and in the parity symbols are determined using an erasure vector V_E, which is generated by knowing the position of the tampered blocks. The length of V_E is $R \times N_B$, and each tampered block with a raster order m will indicate an erasure in V_E at positions $m, m + N_B, \dots , m + R \times N_B$.

The parity information PR is stored separately, then the LSBs in which the watermark was stored are cleared by storing zeros in them. Each untampered block is compressed using SPIHT algorithm, and zeros are stored instead of the SPIHT data for the tampered blocks. Let SP denotes the SPIHT compressed data for the image blocks. The number of bits in each row of SP is n_M. PR and SP are concatenated to form the RS code stream RC, which has $R \times N_B$ rows and each row has n_M bits.

The next step is to correct the erased rows in RC by dividing each row into symbols, where each symbol has n_S bits. One symbol is taken from each row of RC to form an RS code vector V_R that has $R \times N_B$ symbols with each symbol has n_S bits. The code vector V_R and the erasure vector V_E are fed into the RS decoder, and the output is the corrected version of V_R. The decoding process is repeated for the remaining symbols of RC until all of its rows are corrected. The first N_B rows of RC represent the corrected SP stream that has the SPIHT encoded data for the image. The tampered blocks are recovered by decoding the corresponding rows in the corrected SP using the SPIHT algorithm. The result of this step is the recovered image I_R.

The LSBs in I_R, in which the watermark was stored, are adjusted by storing 1 in the left most (i.e., most significant) LSB and zeros in the remaining bits. By doing this, the PSNR of the recovered image is enhanced [6]. The output of this step is the final decoded image I_D.

4. RESULTS AND DISCUSSION

To get a statistical overview about the effect of choosing different number of LSBs, block size, and code-rate (represented by its reciprocal R), the proposed method is applied to 1000 random images chosen from BOWS-2 data-set [22]. The tampering ratio is increased from 0 (i.e., no tampering) to the TTR applicable for the chosen R, then the average value of the PSNR for the 1000 images is plotted against the tampering ratio. The average PSNR is plotted for a block size of 32×32 and 64×64 pixels. The average PSNR is shown in Fig. 3 for R = 2, 3, 4, and 5, respectively.

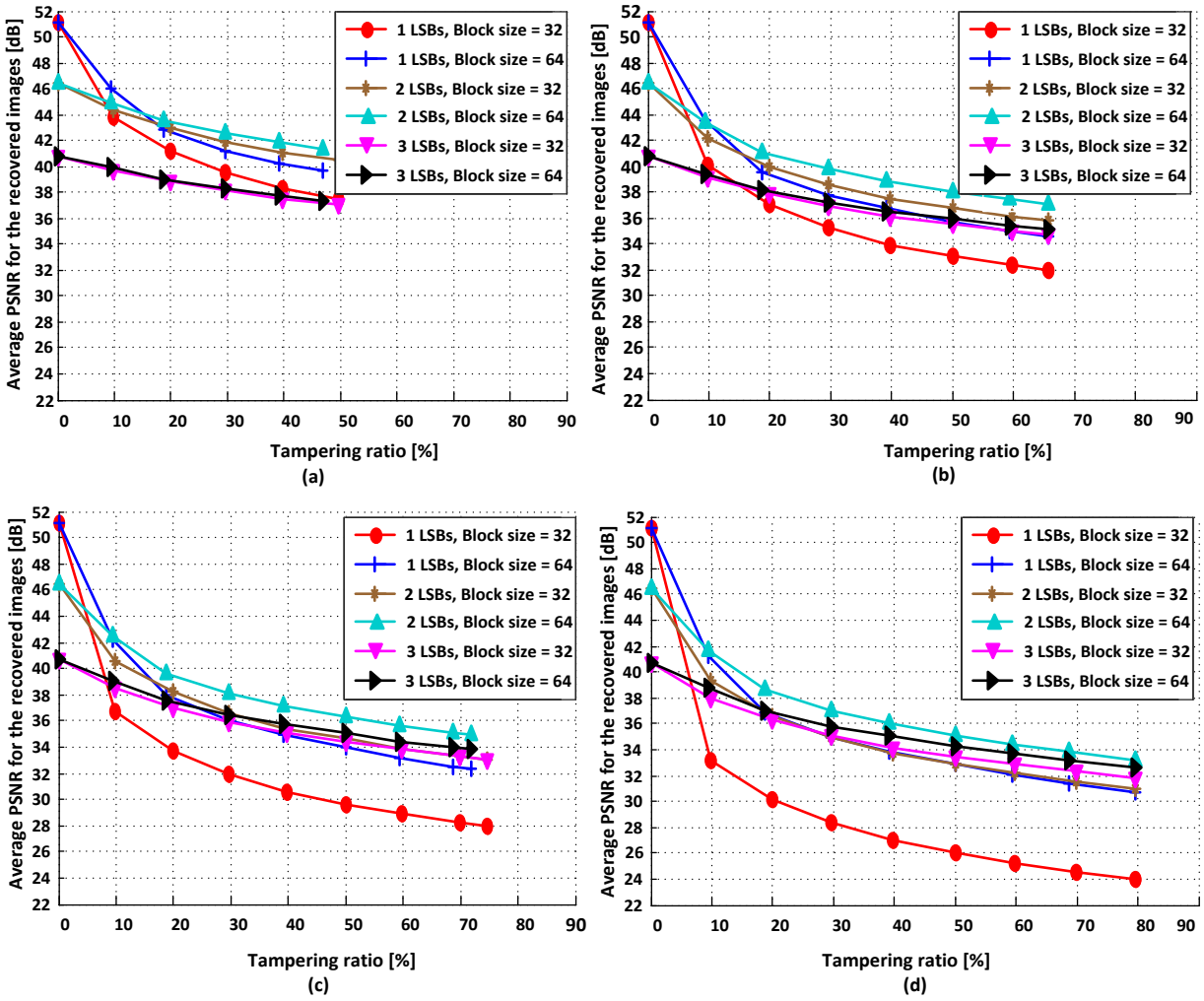

Fig. 3. Average PSNR for 1000 random images - chosen from BOWS-2 data-set - for code-rate reciprocal R values of: a) 2; b) 3; c) 4; d) 5.

In Fig. 4, the proposed method is compared with two related methods, nalemy i) Sarreshtedari and Akhaee method [8] and ii) Korus and Dziech method [7]. The average PSNR is calculated for 1000 random images from BOWS-2 data-set and plotted against

tampering ratio up to the TTR. In both methods and the proposed in this paper method, the tampered blocks are replaced by the recovered ones, then the LSBs used to store the watermark are set to zeros and the most significant LSB is then set to 1 to enhance the PSNR of the recovered image; this has been done in [6–8].

The average PSNR decreases as the TTR increases because the recovered blocks - due to compression - have lower quality compared to the original. Increasing the block size will increase the image quality because larger blocks has better compression quality compared to small ones, as seen before with Lena image in Table 1. Increasing the number of LSBs for the watermark increases the recovered image quality because more space is provided for the compressed reference data in the watermark.

In Korus's method, for $\lambda = 1$ and $\lambda = 2$ the achieved TTR is 50% and 33%, respectively. For Sarreshtedari's method, 2 and 3 LSBs are used, with a TTR of 33% and 60%, respectively. Only the case of 2 LSBs is chosen for the proposed method as a good compromise between the tampered and the original image quality. Figs. 4(a) and (b) show the PSNR values for blocks sizes 32×32 and 64×64, respectively. A block size of 64×64 might seems very coarse for an image with size of 512×512; however, this block size or even a larger one is quite acceptable for current time photography size that reaches tens of megapixels. The effect of increasing the block size on enhancing the PSNR value is apparent by comparing Figs. 4(a) and (b).

By looking at Fig. 4, the proposed technique shows superior quality and higher recovery ratio when compared to Sarreshtedari's and Korus's methods, especially in the case of 64×64, and it can be seen that it reaches higher TTR (up to 80%) with a good recovery quality, given that the quality can be further improved by choosing even a larger block size.

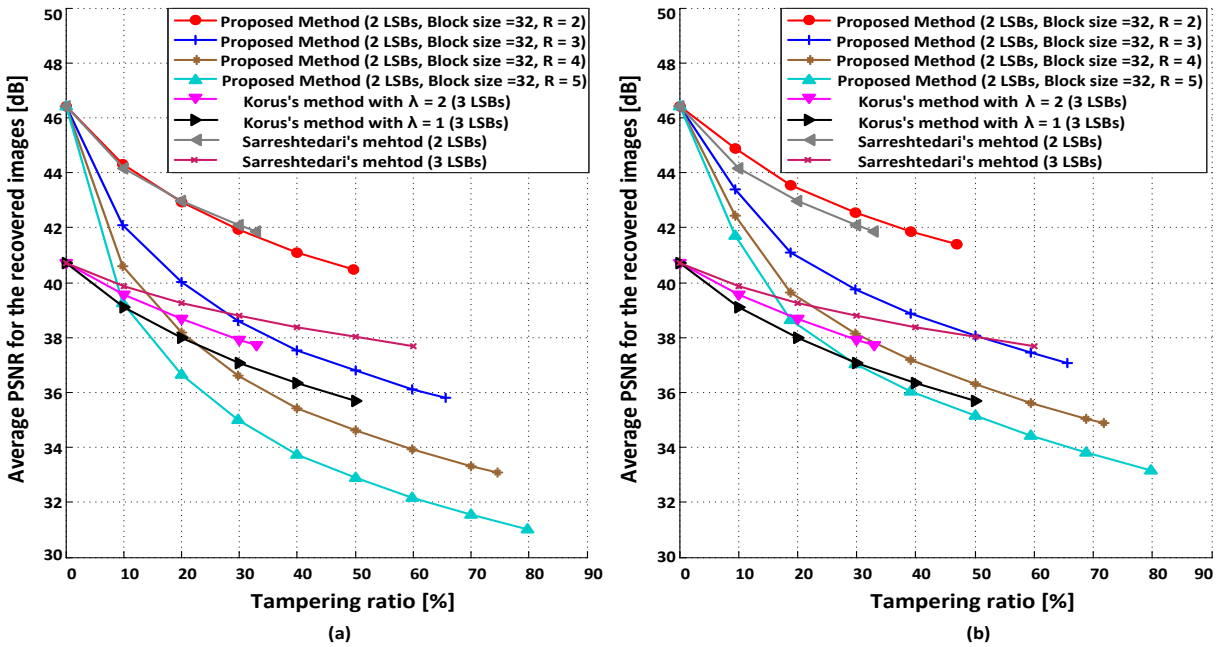

Fig. 4. A comparison between Sarreshtedari's, Korus's, and the proposed method, showing the average PSNR for 1000 images chosen from BOWS-2 data-set; block sizes in the proposed method are: a) 32×32; b) 64×64.

5. CONCLUSIONS

In this paper, an optimized tamper detection and recovery method is proposed, in which image tampering and recovery were modeled as a source-channel coding problem for

an erasure channel. The proposed method divided the original image into large non-overlapping blocks and used SPIHT compression algorithm for source-coding, which provided high recovery quality. RS coding was used for channel-encoding the source-coded blocks, which provided a guaranteed TTR. To increase the speed of the RS coder, the source-coded image blocks were divided into symbols, and the message for the RS encoder was composed by taking one symbol from each block. In this way, the encoding time was reduced dramatically while maintaining the same TTR. The proposed method also provided enhanced security measures by randomly permuting the watermark and XORing it with a random sequence. The seed used for randomization was generated uniquely for each block by combining a secret key with the raster order of the block and a unique serial number for the image. When compared to related methods, the proposed method showed a competing recovery quality while providing a higher TTR.

REFERENCES

[1] A. Haouzia, R. Noumeir, "Methods for image authentication: a survey," *Multimedia Tools and Applications*, vol. 39, no. 1, pp. 1-46, 2008.

[2] P. Korus, "Digital image integrity–a survey of protection and verification techniques," *Digital Signal Processing*, vol. 71, pp. 1-26, 2017.

[3] C. Rey, J. Dugelay, "A survey of watermarking algorithms for image authentication," *EURASIP Journal on Advances in Signal Processing*, vol. 2002, no. 6, pp. 1-9, 2002.

[4] T. Liu, Z. Qiu, "The survey of digital watermarking-based image authentication techniques," *In 6th International Conference on Signal Processing*, vol. 2, pp. 1556-1559, 2002.

[5] K. Sreenivas, V. Kamkshi Prasad, "Fragile watermarking schemes for image authentication: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 7, pp. 1193-1218, 2018.

[6] X. Zhang, S. Wang, Z. Qian, G. Feng, "Reference sharing mechanism for watermark self-embedding," *IEEE Transactions on Image Processing*, vol. 20, no. 2, pp. 485-495, 2010.

[7] P. Korus, A. Dziech, "Efficient method for content reconstruction with self-embedding," *IEEE Transactions on Image Processing*, vol. 22, no. 3, pp. 1134-1147, 2012.

[8] S. Sarreshtedari, M. Akhaee, "A source-channel coding approach to digital image protection and self-recovery," *IEEE Transactions on Image Processing*, vol. 24, no. 7, pp. 2266-2277, 2015.

[9] Rice University, *Lena Image*. <https://www.ece.rice.edu/~wakin/images/>

[10] A. Said, W. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243-250, 1996.

[11] Rensselaer Polytechnic Institute, *Matlab Code for SPIHT*. <https://ecse.rpi.edu/~pearlman/SPIHT/EW_Code/MATLAB-SPIHT/_v1.0/_02-12-08.zip >

[12] L. Rosales-Roldan, M. Cedillo-Hernandez, M. Nakano-Miyatake, H. Perez-Meana, B. Kurkoski, "Watermarking-based image authentication with recovery capability using halftoning technique," *Signal Processing: Image Communication*, vol. 28, no. 1, pp. 69-83, 2013.

[13] J. Molina-Garcia, B. Garcia-Salgado, V. Ponomaryov, R. Reyes-Reyes, S. Sadovnychiy, C. Cruz-Ramos, "An effective fragile watermarking scheme for color image tampering detection and self-recovery," *Signal Processing: Image Communication*, vol. 81, pp. 115725, 2020.

[14] J. Fridrich, M. Goljan, N. Memon, "Cryptanalysis of the Yeung-Mintzer fragile watermarking technique," *Journal of Electronic Imaging*, vol. 11, no. 2, pp. 262-274, 2002.

[15] S. Lin, D. Costello, *Error Control Coding*, Scarborough: Prentice Hall, vol. 2, no. 4, 2001.

[16] Y. Ma, D. Yuan, H. Zhang, "Fountain codes and applications to reliable wireless broadcast system," *In 2006 IEEE Information Theory Workshop-ITW'06 Chengdu*, pp. 66-70, 2006.

[17] T. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*, John Wiley and Sons, 2020.

[18] S. Sarreshtedari, A. Abbasfar, M. Akhaee, "A joint source–channel coding approach to digital image self-recovery," *Signal, Image and Video Processing*, vol. 11, no. 7, pp. 1371-1378, 2017.

[19] M. Fan, H. Wang, "An enhanced fragile watermarking scheme to digital image protection and self-recovery," *Signal Processing: Image Communication*, vol. 66, pp. 19-29, 2018.

[20] Y. Gu, H. Yang, B. Yan, X. Wang, Z. Zhao, "Digital image self-recovery algorithm based on improved joint source-channel coding optimizer," *Multimedia Tools and Applications*, vol. 78, no. 15, pp. 21041-21064, 2019.

[21] S. Sarreshtedari, M. Akhaee, A. Abbasfar, "Source–channel coding-based watermarking for self-embedding of JPEG images," *Signal Processing: Image Communication*, vol. 62, pp. 106-116, 2018.

[22] European Network of Excellence ECRYPT, *2nd BOWS Contest (Break Our Watermarking System).* <http://bows2.ec-lille.fr/>