Reversible Data Hiding in Encrypted Images using Local Difference of Neighboring Pixels

Ammar Mohammadi, and Mansor Nakhkash

Abstract—This paper presents a reversible data hiding in encrypted image (RDHEI), which divides image into non-overlapping blocks. In each block, central pixel of the block is considered as leader pixel and others as follower ones. The prediction errors between the intensity of follower pixels and leader ones are calculated and analyzed to determine a feature for block embedding capacity. This feature indicates the amount of data that can be embedded in a block. Using this pre-process for whole blocks, we vacate rooms before the encryption of the original image to achieve high embedding capacity. Also, using the features of all blocks, embedded data is extracted and the original image is perfectly reconstructed at the decoding phase. In effect, comparing to existent RDHEI algorithms, embedding capacity is significantly increased in the proposed algorithm. Experimental results confirm that the proposed algorithm outperforms state of the art ones.

Index Terms—Encrypted image, local difference, prediction errors, reversible data hiding.

I. INTRODUCTION

Reversible data hiding (RDH) intends to carry a secret data in a cover signal like image, while cover signal is reconstructed absolutely after perfect extraction of the embedded secret data. Most RDH schemes in plaintext image use correlation of neighboring pixels to embed secret data [1-5]. The RDH has many applications, among them are copyright protection, authentication, cover communication, and so on. The RDH in encrypted image (RDHEI) is also used in cloud computing to preserve content owner privacy and provide a platform to embed additional data in encrypted image including some necessary notifications, content owner identification, source and destination information and so on. In RDHEI, without knowing original content of the image or encryption key, additional data may be embedded in encrypted image. At the decoding phase, original image and embedded data can be restored and extracted respectively.

In recent years many research papers have been devoted to RDHEI [6]. Generally, they can be classified into three groups: reserving room before encryption (RRBE), vacating room by encryption (VRBE) and vacating room after encryption (VRAE). Also, the RDHEI methods embed data in encrypted image based on two fundamental approaches: compressing encrypted image [7-9] and exploiting the correlation between neighboring pixels [10-21]. In another classification, RDHEI schemes may be categorized to separable or joint methods. In separable methods, the data extraction and image decryption are performed separately, whereas the data extraction cannot be realized without knowing the information of decrypted image in joint methods.

The idea of compressing the encrypted image is introduced in [22, 23]. It is obvious that compressing the encrypted image vacates room to embed owner data. Using this idea, [7-9] perform RDHEI by VRAE that may be more practical than two other ones; however, high embedding capacity is not achievable in their scheme. Moreover, their scheme might not reversibly vacate room in the encrypted image under some circumstances. At the decoding phase, message extraction is separately done in all of them.

In most RDH scheme, the correlation of neighboring pixels exploits to calculate prediction errors of the original image and in turn, it will be used to embed the secret data. The correlation is, also, used in RDHEI to perform hiding data. Whole schemes will be discussed after use correlation of neighboring pixels to realize RDHEI. In 2013, [12] proposed a scheme of commutative reversible data hiding and encryption. Data hider first fragments the original image into a series of non-overlapping blocks each of which including two neighboring pixels. To encrypt original image, intensities of two neighboring pixels are masked by same pseudo-random bits. Thus, difference of these neighbors are not changed, provides spatial redundancy in encrypted image to embed owner data. There is a pseudo-randomly permutation process in this scheme to prevent some possible attacks that may occur because of information leakage, i.e. difference of neighboring pixels will not be encrypted. In 2014, Zhang et al. [14] presented a RDHEI that instead of embedding data in encrypted images directly, some pixels are predicted before encryption so that the owner data can be embedded in the prediction errors. Thus, part of these prediction errors will not be encrypted. Another attempt [13] introduces a RDHEI, in which, the cover image is divided into separate blocks and multigranularity encryption is applied to attain the encrypted image by random permutation of blocks and random permutation of pixels in each block. It preserves the correlation of neighboring pixels in each block that is used to calculate prediction errors. Data embedding is done by modifying these errors. In 2016, Xu and Wang [16] reported a RDHEI using correlation of sample pixels and non-sample ones. Sample pixels as reference points are used to calculate prediction errors of non-sample pixels. A stream cipher is used to encrypt sample pixels and a specific encryption procedure is planned to encrypt prediction errors of non-
sample ones. Accordingly, a part of prediction errors that are more frequent is not encrypted and this leads a procedure to embed owner data in encrypted image. This procedure may be organized using a modified version of histogram shifting and difference expansion technique. In this scheme, a part of prediction errors will not be encrypted. Huang et al. [15] also propose a new framework for RDHEI. Their new framework allows the numerous RDH schemes directly be exploited in the encrypted domain. They divide image to different blocks and use the correlation of neighboring pixels in each block to embed owner data. They encrypt whole pixels in a block by using a similar random integer. Thus, it will preserve correlation of pixels in a block that makes redundancy to embed owner data in encrypted image. Although persevering correlation of neighboring pixels makes possibility to embed data in an encrypted block, it may lead to an effective know/chosen-plaintext attack. However, using block permutation, they try to protect against this attack by random distribution of blocks in the whole image. In 2019, Yi an Zhou [20] proposed a high embedding capacity scheme that exploits correlation of pixels in a block to embed owner data in the encrypted image. Their scheme includes four main steps, image blocking, pixel grouping, pixel labeling, and payload embedding. In this scheme, the image encryption involves two procedures: block permutation and pixel modulation. All pixels in a selected block are added by a similar random integer to organize pixel modulation. In this procedure, the correlation of neighboring pixels in blocks is preserved to be exploited by data-hider in order to embed owner data. However, in this procedure, i.e. by adding similar random integer to whole pixels in a block, difference of pixels in a block will not be private. Thus, it may lead to information leakage in the encrypted image. However, security analysis has confirmed the robustness of this scheme in withstanding brute-force and know/chosen-plaintext attacks. This scheme significantly improves hiding capacity rather than previous ones.

All five aforementioned methods use VRBE approaches. For the decoding phase, these schemes are separable. Original image perfectly is reconstructed in these schemes. All of them do not encrypt difference of some pixels of the original image because of exploiting correlation of the neighboring pixels to embed data in encrypted image. Some attacks may be possible in these schemes because of information leakage.

In 2011, Zhang [10] proposed a RDHEI in accordance with VRAE procedure so that owner data is embedded in image by modifying a small proportion of encrypted image. They take advantages of spatial correlation in natural image to extract data. This scheme is a joint one. Data extraction and image reconstruction perfectly will be done just under some circumstances in this scheme. This scheme was later improved by Hong et al. [11] using spatial correlation between neighboring blocks and a side-match technique. In 2016, Zhou et al. [18] improved two previous schemes in hiding capacity and accuracy of data extraction and original image recovery. They exploit several binary public keys to embed data that are selected according to a criterion of maximizing the minimum Hamming distance among all keys. At the decoder side, a powerful two-class SVM classifier is exploited to separate encrypted and non-encrypted image patches. The reported methods in [10], [11] and [18] use VRAE approaches. At the decoding phase, these schemes are joint and the original image is perfectly reconstructed just under some circumstances.

Using a traditional RDH method, Ma et al. [21] realize a RDHEI that reserves room before encryption by embedding LSBs of some pixels into other ones. After encryption, the positions of these vacated LSBs are used to embed data. Also, Puteaux and Puech [19] present two high capacity RDHEI schemes by RRBE. In one of them, perfect reconstruction of the original image is guaranteed. Their scheme employs the correlation of two neighboring pixels so that a pixel can be predicted by adjacent one. Therefore, prediction errors may be calculated in plaintext image that are used to make a location binary map. After encryption, the data can be embedded in MSBs of encrypted pixels using this location binary map. This scheme guarantees the perfect original image reconstruction at decoding phase and is separable.

This paper reports a RDHEI scheme that uses local difference of separated blocks of the original image to compute prediction errors and obtains a feature for each block analyzing the prediction errors. Such a feature shows the amount of bits embedded in the blocks and we call it block capacity feature (BCF). The BCFs are compressed and encrypted as overhead data to be used for decoding phase. Both the overhead and owner data are hierarchically embedded in encrypted image that leads to marked encrypted one. At the decoding phase, at first, the overhead data is extracted and then, it is decrypted and decompressed to bring out the BCFs. Using these BCFs, owner data can be extracted and original image may be reconstructed separately. Implementing of our proposals and comparing the results with those of the existent RDHEI algorithms, we demonstrate the significant improvement in embedding capacity.

II. PROPOSED SCHEME

In this section, we present our proposed scheme in details including, analyzing hiding capacity, vacating room before encryption, encrypting original image, embedding owner and overhead data, extracting data and recovering original image. Fig. 1 indicates an overview of the proposed scheme. Fig. 1a demonstrates the procedure for embedding owner data in encrypted image that leads to construct the marked image and Fig. 1b illustrates the data extraction process and the recovery of the original image. According to Fig. 1a, in order to embed data, the image is first divided into non-overlapping blocks. By pre-processing, the BCF of each block is determined and exploited to embed data after encrypting the original image. The encrypted image is obtained using bitwise XOR of the original one with a stream cipher that is created by an encryption algorithm with an input key. Moreover, the BCFs are compressed and encrypted that we consider them as overhead data. The overhead and owner data are embedded in
the encrypted image employing the same BCFs in a hierarchical procedure and the resultant is the marked encrypted image. At the decoder side, as shown in Fig. 1b, the data extraction and original image reconstruction separately can be done. According to this figure, the image is first divided into non-overlapped blocks. Then, overhead data are extracted, decrypted and decompressed to bring out BCFs. It should be noted that the extraction of BCFs themselves needs the knowledge of BCFs. This dilemma is solved by a hierarchically procedure that is described later. Finally, using the BCFs, the owner data and original image, respectively, will be extracted and reconstructed.

**A. Analyzing Hiding Capacity**

In [19], which introduces a high capacity RDHEI, the prediction error $(e)$ is calculated according to

$$p - \rho = e$$  \hspace{1cm} (1)

that is difference between a pixel intensity $(p)$ and its prediction amount $(\rho)$. In [19], they prove that if the condition

$$|e| < 64$$  \hspace{1cm} (2)

is satisfied, a data bit can be embedded in form of replacing the MSB bit of $p$ by a data bit and at the recovery phase, the MSB of the pixel $p$ can be retrieved without error using $\rho$. The algorithm of [19] embeds at most one data bit in each pixel. Considering condition (2), we are inspired that the capacity of each pixel may be increased if the condition is replaced as

$$|e| < 2^n \quad 0 \leq n \leq 7$$  \hspace{1cm} (3)

In this way, the embedding capacity of pixel $p$ denoted as $n'$ bits, is given by

$$\begin{cases} n' = 8 - n - 1 & n \neq 0 \\ n' = 8 & n = 0 \end{cases}$$  \hspace{1cm} (4)

According to (4), $n' = 7$ never is realized because when $n \neq 0$, a bit should be devoted for $e$ sign and 1 is subtracted to obtain $n'$ in (4). For $n = 0$, (3) gives $e = 0$ and therefore, 8 bits of data can be embedded in $p$. In other words, when the pixel intensity is exactly equal to the prediction one, it can be replaced entirely by the data bits. Regarding $n = 7$, prediction error could be any value in $-128 < e < 128$ range that means there is no capacity to embed data bits. When $n = 6$, just 1 bit capacity ($n' = 1$) is provided as reported in [19]. We demonstrate data embedding process as follows in more details:

Let’s represent the intensity of a pixel as $p = p_7p_6p_5p_4p_3p_2p_1p_0$ with 8 bits from LSB ($p_0$) to MSB ($p_7$), and express $D = \{d_0, d_1, ..., d_{n'-1}\}$ as a set of $n'$ bits of data that should be embedded. The marked pixel, $[p]$, that carriers $n'$ bits is calculated by

$$[p] = \sum_{k=1}^{n'}(2^{8-k} \times d_{k-1}) + \sum_{k=n'+1}^{8}(2^{8-k} \times 2^{8-k})$$  \hspace{1cm} (5)

In this equation, any $\sum_{i=n}^{m}$ with $n > m$ is considered as the empty sum, i.e. 0. Thus, a brief description of embedding data in $p$ is that using prediction error, $n$ is calculated ((3)), then the value $n'$ is achieved using (4) and finally, $n'$ bits of data are embedded in $p$ with respect to (5). As an example, let’s
Where \( n' \) at the decoding side, it is obvious the most significant \( n' \) bits in \([p]\) are the embedded bits. It remains to describe how to obtain the original intensity of \( p \). First of all, one can obtain the value of \( n \) by knowing \( n' \) and vice versa. For the case \( n = 0 \), \( p = \rho \) and for \( n = 7 \), \( p \) is not changed in embedding procedure. For \( 0 < n < 7 \), there is only one unique value for \( p \) that gives the actual error \( e \) and satisfies (3) and we rewrite it in another form as

\[
-2^n < e = (p - \rho) < 2^n \quad (6)
\]

This is because any change in actual value of the most significant \( n' \) bits of \( p \) results in at least \( \pm 2^{n-n'} \) change in value of \( p \), which increases or decreases the error at least \( 2^{n-n'} = 2^{n+1} \), i.e. the new error will satisfy \( e_n > e + 2^{n+1} \) or \( e_n < e - 2^{n+1} \). According to (6), \( e_n \) would, therefore, satisfy the conditions \( e_n > 2^n \) or \( e_n < -2^n \) that is in contradiction with the restriction (6). Consequently, one value for \( p \) can just give the restriction (6).

For the sake of brevity, image encryption was not considered. The recovery process of the encrypted pixels and extraction data are explained in more detail in Subsection II.F.

B. Vacating Room Before Encryption

In this section, we describe our technique to vacate room before encryption in order to embed data bits after encryption.

In [3], a prediction technique that is local in the difference of the original image pixels is used to propose a RDH scheme for medical images. This scheme calculates the difference of the neighboring pixels from a base one in separable blocks of the image to construct prediction errors. In our proposed scheme, we introduce a RDHEI that uses the idea of local difference to vacate room before encryption.

Let’s assume a host image consists of \( M \times L \) pixels and we divide this image to \( N_b \) blocks with size of \( m \times l \), that denoted by \( B = \{ b_1, b_2, ..., b_k, ..., b_{N_b} \} \). This image blocking may be formed with the sizes of \( 2 \times 2, 3 \times 3 \) or \( 4 \times 4 \). In these blocks, there is one leader pixel and the rest of the pixels are as followers. Also, whole pixels in the image can be categorized in three different sets \( P_l, P_f \) and \( P_b \). The \( P_l \) and \( P_f \) sets, respectively, comprise the leader and the follower pixels in all blocks. The \( P_b \) set includes pixels of a few rows or columns outside of the blocks that are placed at the borders of the image. The \( P_l, P_f \) and \( P_b \) sets are denoted as

\[
P_l = \{ p_{l_1}, p_{l_{i2}}, ..., p_{l_{iK}}, ..., p_{l_{N_b}} \} \quad (7)
\]

\[
P_f = \{ p_{f_1}, p_{f_{i2}}, ..., p_{f_{iK}}, ..., p_{f_{N_b}} \} \quad (8)
\]

\[
P_b = \{ p_1, p_2, ..., p_{K'}, ..., p_K \} \quad (9)
\]

where \( K' \) is the number of border pixels and \( p_{l_k} \) and \( p_{K} \) denote one leader pixel and one border pixel respectively. Also, \( P_{l_k} \) indicates the set of the follower pixels for the \( k \)th block.

Fig. 2 shows the \( k \)th block of the image with size \( 3 \times 3 \), including leader and follower pixels.

In this figure, \( P_{l_k}(i) \), \( 1 \leq i \leq 8 \), denote 8 follower pixels of \( P_{l_k} \) set.

There is a correlation between \( p_{l_k} \) and \( P_{l_k} \) pixels and the local difference between \( p_{l_k} \) and \( P_{l_k}(i) \) pixel can be viewed as a measure for the correlation. This difference that is called prediction error is given by

\[
E_k(i) = P_{l_k}(i) - p_{l_k} \quad 1 \leq i \leq 8 \quad (10)
\]

The prediction errors, because of having less redundancy than the original pixels, provide more embedding capacity. Considering \( 3\times3 \) block of Fig. 2, prediction error matrix is generated by computing \( E_k(i) \) for \( 1 \leq i \leq 8 \) (Fig. 3).

In order to employ the embedding algorithm, described in Subsection A, for a block \( k \), we obtain \( d_{n_k} = \max(|E_k|) \) and put it in (3) instead of \(|e| \) to compute \( n \) that is now denoted as \( n_k \). In turn, \( n'_k \) may be attained from (4) by replacing \( n \) with \( n_k \). Because the absolute value of any prediction error in the block are not more than \( e_{m_k} \), \( n'_k \) is the minimum embedding capacity that can be realized for each follower pixel in block \( k \). The amount \( n'_k \) is considered as the BCF of the block \( k \). The number of pixels in a block with size of \( m \times l \) is, therefore, \((m \times l - 1)\) so that the whole embedding capacity for the block \( k \) is given by

\[
S_{B_k} = (n'_k)(m \times l - 1) \quad (11)
\]

Considering the number of blocks in image is \( N_b \), the total embedding capacity will be

\[
S = \sum_{k=1}^{N_b} S_{B_k} \quad (12)
\]

Set \( C_b = \{ n'_1, n'_2, ..., n'_K, ..., n'_{N_b} \} \) is the set of BCFs for blocks 1 to \( N_b \). This set is required in decoding phase to extract the embedded data and recover the original image. Hence, its members are, hierarchically, compressed and encrypted to make the overhead data. After encryption of original image, using \( C_b \), the overhead and owner data are embedded.

C. Encrypting Original Image

The intensity of the \((i,j)\)th pixel is encrypted using bitwise
“exclusive or” of the original intensity with a stream cipher as follows.

\[ \hat{O}(i,j) = S(i,j) \oplus O(i,j) \]  \hspace{1cm} (13) 

Set \( S \) includes random bytes that are uniformly distributed and may be generated using AES in the CTR mode with an input key (\( K_e \)). It should be noted that classifying the pixels (with regard to (7), (8) and (9)), the encrypted image is also, composed of three different encrypted pixels set as follow.

\[ \hat{p}_l = \{ \hat{p}_{11}, \hat{p}_{12}, ..., \hat{p}_{i_k}, ..., \hat{p}_{i_{N_b}} \} \]  \hspace{1cm} (14) 
\[ \hat{p}_r = \{ \hat{p}_{r1}, \hat{p}_{r2}, ..., \hat{p}_{r_k}, ..., \hat{p}_{r_{N_b}} \} \]  \hspace{1cm} (15) 
\[ \hat{p}_b = \{ \hat{p}_1, \hat{p}_2, \hat{p}_3, ..., \hat{p}_k', ..., \hat{p}_k'' \} \]  \hspace{1cm} (16)

Clearly, the original image may be rebuilt using bitwise “exclusive or” operation as

\[ O(i,j) = S(i,j) \oplus \hat{O}(i,j) \]  \hspace{1cm} (17)

\section*{D. Embedding Overhead and Owner Data}

Considering the fact that extracting the data and recovering the original image are performed with knowledge of \( C_b \), this set should be embedded in the encrypted image. Hence, \( C_b \) is hierarchically compressed, encrypted and embedded along with the owner data in the encrypted image. To do so, \( C_b \) is divided into \( l \) subset in form of \( C_b = \{ N_1, N_2, ..., N_l \} \). Each \( N_i \) includes \( N_b/l \) BCFs, which belongs to the same number of neighboring blocks. For example, \( N_i = \{ n_i, n_2, ..., n_{N_b} \} \) consists of the first \( N_b/l \) BCFs. Since these features belong to the neighboring blocks and are more correlated, more efficient compression may be obtained by compressing the difference of these features. Therefore, the difference of a feature with previous one is computed in each \( N_i \) and the emerging set is denoted \( D N_i \). One assumption is that no difference is obtained for the first feature, e.g. \( n_1 \) in \( N_i \), and it remains intact for data extraction and, hence, it is substituted in \( D N_i \) without any change. Each \( D N_i \) is then, compressed and encrypted to yield \( \hat{N}_i \) set, which will be included in \( \hat{N}_b = \{ \hat{N}_1, \hat{N}_2, ..., \hat{N}_l \} \) as overhead data.

Now, we explain how to embed both overhead and owner data in the host image. Let’s assume the set of blocks corresponding to each \( N_i \) is \( \hat{B}_i \) that is composed of \( (N_b/l) \) blocks, e.g. \( B_1 = \{ b_{11}, b_{12}, ..., b_{i_k}, ..., b_{i_{N_b}} \} \). The set of all blocks can be rewritten as \( B = \{ B_1, B_2, ..., B_l \} \) and after encryption, it may be expressed as \( \hat{B} = \{ \hat{B}_1, \hat{B}_2, ..., \hat{B}_l \} \), in which any \( \hat{B}_i \) is composed of \( N_b/l \) encrypted blocks, e.g. \( \hat{B}_1 = \{ \hat{b}_1, \hat{b}_2, ..., \hat{b}_k, ..., \hat{b}_{N_b/l} \} \). Fig. 4 illustrates such grouping procedure for 3 x 3 blocks.

At the beginning of data embedding, the \( \hat{N}_b \) set as overhead data should be embedded in \( \hat{B} \). It is hierarchically embedded as : \( \hat{N}_1 \) in \( \hat{B}_1 \), \( \hat{N}_2 \) in \( \hat{B}_2 \), \( \hat{N}_l \) in \( \hat{B}_{l-1} \) and \( \hat{N}_l \) in \( \hat{B}_{l-1} \) and the marked blocks \( [\hat{B}_1], [\hat{B}_2], [\hat{B}_{l-1}] \) and \( [\hat{B}_{l-1}] \) are made. After embedding any \( \hat{n}_i \) in \( \hat{B}_i \), the residual capacity is used to embed owner data. The total capacity of last block (\( \hat{B}_l \)) is allocated just for owner data. The embedding algorithm for each block is the one described in Subsection A so that (5) is used to embed data in every \( \tilde{p}_{k'}(i) \), \( 1 \leq i \leq (m \times l - 1) \), where \( n' \) and \( p \) are replaced by \( n_i \) and \( \tilde{p}_{k'}(i) \) respectively.

It should be noted that just \( \hat{N}_i \) data is not embedded in the above procedure and that is why we introduce the border pixels \( \hat{p}_b \) (Fig. 4). In the next section, a new procedure to embed \( \hat{N}_i \) in \( \hat{p}_b \) set will be described.

\section*{E. Embedding Data in the Boarder Pixels}

The \( \hat{N}_i \) is the only member of \( \hat{N}_b \) set, which is not embedded in \( \hat{B}_i \). In this section, the process of embedding \( \hat{N}_i \) in the encrypted border pixels (\( \hat{p}_b \)) is explained. To embed \( \hat{N}_i \) in \( \hat{p}_b \) set, a room should be vacated in \( p_b \) before encryption. As mentioned in Subsection A, the embedding capacity for each pixel is determined by analyzing its prediction error. Therefore, each \( p_{k'} \) pixel in \( p_b \) is predicted based on its previous one \( (p_{k'-1}) \), i.e. the prediction error is computed as

\[ e_{k'(k'-1)} = p_{k'} - p_{k'-1} \]  \hspace{1cm} (18) 

A bit embedding capacity will be provided in MSB bit of the \( p_{k'} \) pixel when this error satisfies

\[ |e_{k'(k'-1)}| < 64 \]  \hspace{1cm} (19) 

As a result, the prediction error for all border pixels is computed using (18) and set \( E_{p_b} = \{ e_{21}, e_{32}, ..., e_{k'(k'-1)}, ..., e_{K'(k'-1)} \} \) will be obtained for \((K' - 1)\) errors. Considering the fact that the boarder pixels
are neighbors and impulsive changes rarely happen between neighboring ones, it is expected that the prediction errors mostly satisfy (19). However, it may not be satisfied in edge regions. In order to extract any data embedded in border pixels including those do not satisfy (19), extra information in form of labels assigned to every error in $E_{pb}$, set, is required. The label set $L_E = \{l_2, l_3, \ldots, l_{k'}, \ldots, l_{K'}\}$, $l_{k'} \in \{0, 1, 2\}$ indicates $(K' - 1)$ labels corresponding to $(K' - 1)$ errors. The determination of a $l_{k'}$ corresponding to $e_{k'}(k' - 1)$ will be according to Algorithm 1, in which, $[.]$ and $\lfloor . \rfloor$ are the floor and the absolute functions, respectively.

The set $L_E$ is compressed and encrypted as $\hat{L}$ that is concatenated with $\hat{N}$ using (20) to generate $\hat{\mathcal{L}}N$ set.

$$\hat{\mathcal{L}}N = \hat{L} \cup \hat{N}$$

The set $\hat{\mathcal{L}}N$ is embedded in $\hat{P}_{b}$ pixels to construct the marked encrypted border pixels $[\hat{P}_{b}]$. To do so, the $(K' - 1)'th$ bit of $\hat{\mathcal{L}}N$ ($\hat{\mathcal{L}}N_{k'-1}$) is hided in $\hat{P}_{b}$ as

$$[\hat{P}_{b}] = 128 \times \hat{\mathcal{L}}N_{k'-1} + (\hat{P}_{b}, \mod 128), \quad 2 < k' \leq K'$$

(21)

Algorithm 1: Labeling procedure for each $e_{k'}(k' - 1)$.

\begin{algorithm}
\begin{algorithmic}
\For{$k' = 2$ to $K'$}
\State $d_{MSB} = |[P_{k'}/128] - [P_{k'-1}/128]|$
\If{$([e_{k'}(k' - 1)] \leq 64)$}
\State $l_{k'} = 0$
\ElsIf{$([e_{k'}(k' - 1)] \geq 64)$ and ($d_{MSB} = 0$)}
\State $l_{k'} = 1$
\Else
\If{$([e_{k'}(k' - 1)] \geq 64)$ and ($d_{MSB} = 1$)}
\State $l_{k'} = 2$
\EndIf
\EndIf
\EndFor
\end{algorithmic}
\end{algorithm}

Algorithm 2: Recovering $P_{fk}(i)$ from $P'_{fk}(i)$ and $e'(i)$ for $(1 \leq i < m \times l)$.

\begin{algorithm}
\begin{algorithmic}
\For{$i = 1$ to $(m \times l - 1)$}
\State $e'(i) = P'_{fk}(i) - p_{ik}$
\If{$(n^2_k = 8)$ or $(n^2_k = 0)$}
\State $P_{fk}(i) = P'_{fk}(i)$
\ElsIf{$[e'(i)] < 2^{(8-n^2_k)}$}
\State $P_{fk}(i) = P'_{fk}(i) + 2^{8-n^2_k}$
\Else
\State $P_{fk}(i) = P'_{fk}(i) - 2^{8-n^2_k}$
\EndIf
\EndFor
\end{algorithmic}
\end{algorithm}

(22)

Algorithm 3: Recovering $P_{k'}$ using $[P_{k'}], [P_{k'-1}], l_{k'}$, and $e_{k'}(k' - 1)$ (1 < $k' \leq K'$).

\begin{algorithm}
\begin{algorithmic}
\For{$k' = 2$ to $K'$}
\State $e_{k'}(k' - 1) = [P_{k'}] - [P_{k'-1}]$
\If{$([e_{k'}(k' - 1)] \geq 64)$ and ($l_{k'} = 0$)}
\State $p_{k'} = [P_{k'}] \text{ xor } 128$
\ElsIf{$(l_{k'} = 1)$}
\State $V_1 = [P_{k'}] \text{ and } 127$
\State $V_2 = p_{k'-1} \text{ and } 128$
\State $p_{k'} = V_1 + V_2$
\Else
\If{$(l_{k'} = 2)$}
\State $V_1 = [P_{k'}] \text{ and } 127$
\State $V_2 = p_{k'-1} \text{ and } 128$
\State $V_2 = V_2 \text{ xor } 128$
\State $p_{k'} = V_1 + V_2$
\EndIf
\EndIf
\EndFor
\end{algorithmic}
\end{algorithm}

(23)

In reconstructing the original image, a $\hat{b}_k$ should be reconstructed to its original one ($b_k$). Therefore, $\hat{b}_k$ is decrypted to $b_k$ including $[P_{fk}(i)]$ set and $p_{ik}$. Recovering $i'\text{'th}$ member of $[P_{fk}(i)]$ set ($[P_{fk}(i)]$) is performed in two steps. In the first step, $[P_{fk}(i)]$ is achieved from (23)

F. Extracting Data and Reconstructing the Original Image

In this section we explain the data bits extraction from marked encrypted blocks and recovering the original image. Let denote all reconstructed set or pixel by bold characters, e.g. $n_k$ is the reconstruction of $n_k$. The procedure to reconstruct $n_k$ is illustrated in the next section.

Having had $n_k$, data bits from any $[\hat{P}_{fk}(i)]$, $1 \leq i < m \times l$, of $[\hat{b}_k]$ are extracted according to

$$D_{i} = \sum_{i' = 1}^{2^8 - n_k} [\hat{P}_{fk}(i')] \times 2^{i'}$$

(22)}
from marked encrypted blocks and reconstructing original image is performed according to the aforementioned technique. In (2), the recovery process for all blocks of the marked encrypted image is performed according to the aforementioned technique to reform the original image.

### G. Reconstructing BCFs

In the previous section, we demonstrate extracting data from marked encrypted blocks and reconstructing original one, using BCFs. Herein, the reconstruction of BCFs will be explained. The first step is to restore $\hat{\Omega}$ that is begun with the extraction of $\hat{\mathcal{L}}$ from $[\hat{\rho}]$ set. Considering the fact that $\mathcal{L}$ is embedded in MSBs of $[\hat{\rho}]$ pixels, $\mathcal{L}$ is extracted for each $[\hat{\rho}_k]$ by

$$\hat{\mathcal{L}}_{k'} = [[\hat{\rho}_k]/128] \quad 1 < k' \leq K'$$

where $(k' - 1)$th bit of $\hat{\mathcal{L}}_{\hat{\mathcal{L}}_{k'-1}}$ is extracted from $[\hat{\rho}_k]$. The number of extracted bits is $K' - 1$. The $\mathcal{L}$ is the
concatenated of \( \hat{N}_1 \) and \( \hat{L} \). Regarding the number of \( \hat{L} \) bits is
stored at the beginning of \( \hat{L} \), \( \hat{L} \) and \( \hat{N}_1 \) can be separated. \( \hat{L} \) is
decrypted and decompressed to recover \( L_E \) set that is
employed to rebuild the border pixels. Furthermore, \( \hat{N}_1 \) is
decrypted and decompressed to \( N_1 \) and is used to extract data
embedded in \( \hat{B}_1 \) including owner data and \( \hat{N}_2 \). Similarly, \( \hat{N}_2 \) is
exploited to extract data from \( \hat{B}_2 \) including owner data and \( \hat{N}_3 \) and so forth.

The border pixels should also, be recovered to form the
original image. The process of recovering these pixels is,
actually, to obtain their MSBs. The set \( \{P_b\} = \{p_1, p_2, p_3, \ldots, p_K, \ldots, p_K'\} \) is the set of border
pixels after decryption. Because the first pixel of \( p_b \) in \( \{P_b\} \)
has remained intact during data embedding process, \( p_2 \)
can be reconstructed using \( p_1 \) and generally, \( p_{K'} \) may be
rebuild using \( p_{K'-1} \) employing Algorithm 3. In this algorithm
"xor" and "and" are bitwise operations and \( L_E = \{l_2, l_3, \ldots, l_K', \ldots, l_K\}, l_k' \in (0, 1, 2) \) is already available from
the extraction procedure that employs (24).

III. EXPERIMENTAL RESULT

We have conducted some experiments to demonstrate the
performance of the developed method for embedding data
with high capacity in encrypted images. Nine gray scale
images F16, Lena, Splash, House, Foster City, Woodland
Hills, Rough wall, Gravel, Baboon from the SIPI database and
three gray scale images Man, Crowd, Barbara from the
Miscellaneous database all 512×512 in size are used as test
images (Fig. 5). Also, BOWS2 original database, including
10000 greyscale images, are exploited to confirm high
capacity of the proposed algorithm in embedding data in an
encrypted image.

Total embedding capacity, the number of owner data bits,
overhead and \( L \) set ones provided by the proposed scheme are
exploited in Table I, for the test images. They are described for
different sizes of blocks: \( 2 \times 2, 3 \times 3 \) and \( 4 \times 4 \). The
total embedding capacity is the summation of the owner and
overhead data bits. Also, the number of owner data bits over
the number of pixels in test images, i.e. 512×512 in size, is
computed and denoted as bit per pixel (bpp) in the table.

It can be seen, by increasing the size of blocks, a
significant reduction in the number of overhead data bits is
occurred; however, the changes in the overall embedding
capacity are less. In other words, image blocking with \( 3 \times 3 \)
and \( 4 \times 4 \) blocks make less overhead data bits than \( 2 \times 2 \) one,
while there is no significant changes in the number of owner
data bits for the most test images. Therefore, image blocking
with \( 3 \times 3 \) blocks for the whole test images provides more
improvement compared to \( 2 \times 2 \) one. Also, using \( 4 \times 4 \) rather
than \( 2 \times 2 \) blocks enhances embedding capacity for the most
test images especially smoother ones.

In all test images, using \( 3 \times 3 \) blocks in comparison with
\( 4 \times 4 \) ones improves embedding capacity due to the fact that
there is more symmetry between leader pixel and follower
ones in this blocking. It provides better prediction that leads to
sharper errors and consequently better embedding capacity.
For a similar reason, smoother images, such as F16 and
Splash, provide more embedding capacity than rougher ones,
such as Woodland Hills, Rough Wall and Baboon, regardless
of block size.

For the image blocking with \( 2 \times 2 \) or \( 4 \times 4 \) sizes, the first 4
rows of the image are exploited as border pixels, i.e. 2048 bits
embedding capacity. Also, for image blocking with \( 3 \times 3 \)
sizes, the first two rows and two columns of the image are
used as border ones, i.e. 2044 bits embedding capacity. These
border pixels are labeled using Algorithm 1. The labels are
compressed using arithmetic coding to provide \( L \) set, the
number of them is tabulated in Table I. It illustrates that
border pixels in Baboon and House images are rougher than
others, so they provide less pure embedding capacity.

In Table II, we illustrate the efficiency of the proposed
algorithm in all sizes of the block for completely 10000
images of Bows2 original database. For image blocking with
\( 2 \times 2, 3 \times 3 \) and \( 4 \times 4 \) sizes, the number of 512, 680 and 768
blocks are used respectively as a set of blocks to form a \( B_1 \)
(Fig 4) in the most images. However, for rough ones, they can

| Image blocking | Block size = 2 × 2 | Block size = 3 × 3 | Block size = 4 × 4 |
|---------------|-------------------|-------------------|-------------------|
|               | Average | Worst | Best | Average | Worst | Best | Average | Worst | Best |
| Total capacity (bits) | 822808 | 267795 | 1435266 | 836987 | 175880 | 1670048 | 782210 | 93990 | 1733475 |
| Owner data (bits) | 674596 | 116979 | 1392770 | 780691 | 108224 | 1652360 | 750952 | 59286 | 1722771 |
| Overhead data (bits) | 148212 | 155816 | 42496 | 56296 | 67656 | 17688 | 31258 | 34704 | 10704 |
| Owner data (bpp) | 2.57 | 0.43 | 5.31 | 2.98 | 0.41 | 6.30 | 2.87 | 0.226 | 6.572 |
| \( L \) set (bits) | 85 | 718 | 21 | 123 | 1454 | 21 | 85 | 1639 | 21 |
| Image (gbpm) | - | 5547 | 1478 | - | 9448 | 1478 | - | 9448 | 1478 |

Table III: Comparison of Embedding Capacity of the Proposed Algorithm with Two High-Capacity DHEI Schemes for Test Images.

| Images | F16 | Lena | Splash | House | Foster | City | Woodland | Hills | Rough | Gravel | Baboon | Man | Crowd | Barbara |
|--------|-----|------|--------|-------|--------|------|----------|-------|-------|--------|--------|-----|-------|--------|
|        | [19] | 0.98 | 0.98   | 0.99  | 0.94   | 0.99 | 0.97     | 0.99  | 0.99  | 0.75   | 0.92   | 0.98 | 0.76   |
|        | [20] | 2.21 | 2.02   | 2.66  | 1.55   | 1.6  | 0.99     | 1.26  | 1.65  | 0.75   | 1.46   | 1.75 | 1.29   |
| Proposed scheme | \( 3 \times 3 \) | 2.86 | 2.57 | 3.04 | 2.48 | 2.36 | 1.39 | 1.48 | 2.07 | 1.16 | 2.03 | 2.56 | 1.97 |
|        | \( 4 \times 4 \) | 2.70 | 2.46 | 2.94 | 2.33 | 2.33 | 1.20 | 1.27 | 1.88 | 1.03 | 1.83 | 2.33 | 1.85 |
be selected in half because of increasing the number of bits in $L$ set, i.e. pure embedding capacity reduction in border pixels.

For image blocking with $3 \times 3$ and $4 \times 4$ sizes, the best and the worst embedding capacities are provided by image number 1478 and 9448 from Bows2 original database, respectively (Table II). For $2 \times 2$, the best and the worst ones are numbers 1478 and 5547, respectively. These three Bows2 images are shown in Fig. 6. As tabulated in Table II, image blocking with $4 \times 4$ size compared to others, improves embedding capacity for the best image. However, for the worst one, employing block with $2 \times 2$ size increases embedding capacity than others. In average, $3 \times 3$ blocking provides an improvement more than 0.4 bpp and 0.1 bpp, respectively, compared to $2 \times 2$ and $4 \times 4$ blocking. In terms of computational complexity, the use of small sizes for blocks increases the number of blocks, BCFs, overhead data and hence, the computational complexity.

In Table III, the proposed scheme is compared with two high capacity RDHEI schemes [19] and [20] for two different sizes of blocks. It should be noted that [20], in term of embedding capacity, generally improves all previous schemes [7-18, 19, 21]. The simulation of the cases in [20] is performed for the best $\alpha$ and $\beta$, and image blocking with $3 \times 3$ size. The $\alpha$ and $\beta$ are two parameters in range $1 \leq \alpha, \beta \leq 7$ that affect the embedding capacity. Also, the effect of pixels modulation, which reduces slightly embedding capacity, is included in simulation of the method in [20]. This effect can be different in each iteration due to the random nature of the pixel modulation process.

The embedding capacity for one of the two RDHEI schemes proposed in [19] that guarantees perfect reconstruction of the marked encrypted image is also demonstrated in Table III. In this scheme, except Barbara and Baboon, other test images illustrate embedding capacity near to 1 bpp, while in our proposed scheme for all test images, more than 1 bpp embedding capacity is achieved for both sizes of the image blocking. The proposed method and [20] provide better embedding capacity for smoother images such as F16 and Splash, and less embedding capacity for rougher ones such as Baboon, Rough wall and Woodland Hills.

For all test images and for both sizes of the image blocking, the proposed algorithm provides better embedding capacity than the scheme in [20]. This improvement for the House image is even more than 0.77 bpp. Additionally in our proposed scheme, there are no pixels or features of the original image that remain disclosed or unsecured and its security level depends on the security of the used encryption algorithm.

IV. CONCLUSION

In this paper, we propose a high capacity RDHEI that uses correlation of the pixels in the blocks of the image to compute local difference. In local difference, intensity of follower pixels are subtracted from leader one to form the prediction errors. By analyzing these errors, because of having less redundancy than original pixels, BCFs are brought out to vacate rooms before encryption. These features are compressed and encrypted to provide overhead data. Owner and overhead data are hierarchically embedded in the encrypted image to make marked encrypted one. At the decoding side, at first, overhead data is extracted and used to restore BCFs. Then, the original image is perfectly reconstructed and owner data is absolutely extracted using the features.

The comparison of the proposed scheme with other state of the art ones confirms that we improve significantly the image embedding capacity. Also, in our developed method, any desirable encryption algorithm can be exploited so that the security of encryption just depends on security of encryption algorithm.

REFERENCES

[1] X. Li, W. Zhang, X. Gui, and B. Yang, “A novel reversible data hiding scheme based on two-dimensional difference-histogram modification,” IEEE Transactions on Information Forensics and Security, vol. 8, no. 7, pp. 1091-1100, Jul. 2013.
[2] X. Li, B. Yang, and T. Zeng, “Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection,” IEEE Transactions on Image Processing, vol. 20, no. 12, pp. 3524-3533, Dec. 2011.
[3] P. Tsai, Y.-C. Hu, and H.-L. Yeh, “Reversible image hiding scheme using predictive coding and histogram shifting,” Signal processing, vol. 89, no. 6, pp. 1129-1143, Jun. 2009.
[4] V. Sachnev, H. J. Kim, J. Nam, S. Suresh, and Y. Q. Shi, “Reversible watermarking algorithm using sorting and prediction,” IEEE Transactions on Circuits Systems for Video Technology, vol. 19, no. 7, pp. 989-999, Jul. 2009.
[5] D. M. Thodi, and J. J. Rodriguez, “Expansion embedding techniques for reversible watermarking,” IEEE transactions on image processing, vol. 16, no. 3, pp. 721-730, Mar. 2007.
[6] Y.-Q. Shi, X. Li, X. Zhang, H.-T. Wu, and B. Ma, “Reversible data hiding: advances in the past two decades,” IEEE Access, vol. 4, pp. 3210-3237, 2016.
[7] X. Zhang, “Separable reversible data hiding in encrypted image,” IEEE Transactions on Circuits Systems for Video Technology, vol. 7, no. 2, pp. 826-832, Apr. 2012.
[8] M. S. A. Karim, and K. Wong, “Universal data embedding in encrypted domain,” Signal Processing, vol. 94, pp. 174-182, Jan. 2014.
[9] Z. Qian, and X. Zhang, “Reversible data hiding in encrypted images with distributed source encoding,” IEEE Transactions on Circuits Systems for Video Technology, vol. 26, no. 4, pp. 636-646, Apr. 2016.
[10] X. Zhang, “Reversible data hiding in encrypted image,” IEEE Signal Processing Letters, vol. 18, no. 4, pp. 255-258, Apr. 2011.
[11] W. Hong, T.-S. Chen, and H.-Y. Wu, “An improved reversible data hiding in encrypted images using side match,” IEEE Signal Processing Letters, vol. 19, no. 4, pp. 199-202, Apr. 2012.
[12] X. Zhang, “Commutative reversible data hiding and encryption,” Security and Communication Networks, vol. 6, no. 11, pp. 1396-1403, 2013.
[13] Z. Yin, B. Luo, and W. Hong, “Separable and error-free reversible data hiding in encrypted image with high payload,” The Scientific World Journal, vol. 2014, Apr. 2014.

[14] W. Zhang, K. Ma, and N. Yu, “Reversibility improved data hiding in encrypted images,” Signal Processing, vol. 94, no. 1, pp. 118-127, Jan. 2014.

[15] F. Huang, J. Huang, and Y.-Q. Shi, “New framework for reversible data hiding in encrypted domain,” IEEE Transactions on Information Forensics and Security, vol. 11, no. 12, pp. 2777-2789, Dec. 2016.

[16] D. Xu, and R. Wang, “Separable and error-free reversible data hiding in encrypted images,” Signal Processing, vol. 123, pp. 9-21, Jun. 2016.

[17] X. Zhang, J. Long, Z. Wang, and H. Cheng, “Lossless and reversible data hiding in encrypted images with public-key cryptography,” IEEE Transactions on Circuits Systems for Video Technology, vol. 26, no. 9, pp. 1622-1631, Sep. 2016.

[18] J. Zhou, W. Sun, L. Dong, X. Liu, O. C. Au, and Y. Y. Tang, “Secure reversible image data hiding over encrypted domain via key modulation,” IEEE Transactions on Circuits Systems for Video Technology, vol. 26, no. 3, pp. 441-452, Mar. 2016.

[19] P. Puteaux, and W. Puech, “An Efficient MSB Prediction-Based Method for High-Capacity Reversible Data Hiding in Encrypted Images,” IEEE Transactions on Information Forensics and Security, vol. 13, no. 7, pp. 1670-1681, Jul. 2018.

[20] S. Yi, and Y. Zhou, “Separable and Reversible Data Hiding in Encrypted Images using Parametric Binary Tree Labeling,” IEEE Transactions on Multimedia, vol. 21, no. 1, pp. 51-64, Jan. 2019.

[21] K. Ma, W. Zhang, X. Zhao, N. Yu, and F. Li, “Reversible data hiding in encrypted images by reserving room before encryption,” IEEE Transactions on Information Forensics and Security, vol. 8, no. 3, pp. 553-562, Mar. 2013.

[22] M. Johnson, P. Ishwar, V. Prabhakaran, D. Schonberg, and K. Ramchandran, “On compressing encrypted data,” IEEE Transactions on Signal Processing, vol. 52, no. 10, pp. 2992-3006, Oct. 2004.

[23] W. Liu, W. Zeng, L. Dong, and Q. Yao, “Efficient compression of encrypted grayscale images,” IEEE Transactions on Image Processing, vol. 19, no. 4, pp. 1097-1102, Apr. 2010.

Ammar Mohammadi received the B.Sc. degree from the Shahid Beheshti University, Tehran, Iran, in 2008, and the M.Sc. degree from Sahand University of Technology, Tabriz, Iran, in 2011. He is currently pursuing the Ph.D. degree at the Yazd University, Yazd, Iran. His research interests are image processing and information hiding.

Mansor Nakhkash received the B.Sc. and M.Sc. degrees in electrical engineering from the Isfahan University of Technology, Isfahan, Iran, in 1988 and 1991, respectively, and the Ph.D. degree in electrical engineering from the University of Liverpool, Liverpool, U.K., in 1999. From 1999 to 2001, he was a Post-Doctoral Research Assistant with the Department of Electrical Engineering, University of Liverpool. Since 2001, he has been with Yazd University, Yazd, Iran, where he is currently an Associate Professor with the Department of Electrical Engineering. His current research interests include microwave imaging, SAR signal processing, and digital data hiding.