Signal Processing 108 (2015) 56–68

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Data embedding in random domain

Mustafa S. Abdul Karim*, KokSheik Wong

Faculty of Computer Science & Information Technology, University of Malaya, Kuala Lumpur, Malaysia

ARTICLE INFO

Article history:
Received 14 May 2014
Received in revised form
9 August 2014
Accepted 12 August 2014
Available online 3 September 2014

Keywords:
DeRand
Random domain
Universal data embedding
Universal parser

ABSTRACT

A universal data embedding method based on histogram mapping called DeRand (Data embedding in Random domain) is proposed. DeRand theoretically defines redundancy in any digital signal by applying the universal parser such that high entropy random signals can certainly be utilized for data embedding. First, DeRand recursively parses a random signal into a set of tuples each of certain length until there exist some tuples of zero occurrences in the histogram. Then, tuples that occur in the histogram are associated with those of zero occurrences. Next, a tuple (of non-zero occurrence) is mapped to its corresponding associated tuple to embed "1", while the tuple is left unmodified to embed "0". DeRand is universal, reversible, applicable to any random signal and scalable in terms of embedding capacity and signal quality. Experimental results show that DeRand achieves an embedding capacity up to 4909 bits in random signal of size 256 Kbytes. In addition, the quality of the processed signal ranges from 0.0075 to 395.67 in terms of MSE.

1. Introduction

Multimedia data is massively generated nowadays thanks to the advanced yet low cost capturing and storage technologies. In addition to the conventional network traffic such as web accesses, email communication, and transmission of e-commerce data, these multimedia data are communicated and shared among various users across continents. The advent of innovative social network services at no cost and the deployment of online content store further multiplied the utilization of network. For example, it is reported that around 28,000 photos and 72 h of videos are uploaded each minute to Instagram and Youtube [1], respectively. Thus, the communication channels, including Internet and cellular phone networks, convey a mixed streams of information (appearing in various formats and coding structures) sent from multiple sources. These streams appear as random data when being transmitted over the communication channels. Generally, any unintelligible data can be considered as a random data (or random signal hereinafter) from the perspective of a third party who has no access to its original (intelligible) form. These random data include encrypted signal, records in database, and data uploaded to cloud-storage, which need to be managed for efficient utilization of the communication bandwidth and storage space. Here, for digital data management purposes, data embedding technologies provide conveniences to achieve annotation, authentication, watermarking, etc. However, fundamentally, data embedding is a feature-dependent process, where features of a host are modified in certain domain and coding structure in order to embed data [2–6]. For example: (a) [7–9] manipulate the features of the host image in the spatial domain; (b) [10–12] process the host image in the frequency domain; (c) [13] is restricted to compressed video; (d) [14] is applicable only to audio; (e) [15] embeds data in text, and; (f) [16–19] are restricted to encrypted signals. Therefore, the interchangeability among most data embedding methods is generally restricted by its domain or media with certain features. In other words, the definition of feature is necessary to achieve

* Corresponding author. Tel.: +603 7967 6417; fax: +603 7957 9249.
E-mail addresses: mustafa.s.abdulkarim@ieee.org, mustafa.alwahaib@siswa.um.edu.my (M.S. Abdul Karim), koksheik@um.edu.my (K. Wong).

http://dx.doi.org/10.1016/j.sigpro.2014.08.037
0165-1684/

data embedding by these methods. On the other hand, it is technically challenging to extract features from a random signal, which is unintelligible. To the best of our knowledge, data embedding in the random domain has not been considered in the literature.

Karim and Wong [20] proposes a reversible data embedding method that can be universally applied to any encrypted signal. This method operates solely in the universal domain [20] and hence it is not dependent on the underlying features of the encrypted host. For that, this method can be applied, theoretically, to any signal. However, this method completely changes the format of the host due to the design of the data embedding mechanism. In particular, all segments of the host are mapped to the Golomb–Rice Codewords [21,22], which are then modified to accommodate a payload while maintaining the bitstream size. Thus, it is impossible to preserve the semantic of the host after achieving data embedding. As a result, the original encrypted host must be completely restored prior to decryption. In addition, there is no control on the embedding capacity and host distortion (viz., not scalable) in [20].

On the other hand, Ong et al. [23] put forward a reversible data embedding method that offers scalability in terms of embedding capacity and perceptual quality in image. In this method, the ability to control the perceptual quality (in the host image) is utilized to achieve perceptual encryption (viz., image scrambling), where the semantic of the image is intentionally masked by the designed substitution operation. Here, the payload is embedded into the image, along with the side information required to restore the original image. The substitution operation is achieved by modifying the histogram of the image. In particular, the probability $P(x)$ of each intensity level x in the image is analyzed. Basically, x is classified into two sets based on its probability, namely, $G_1=\{x: P(x)>0\}$ and $G_2=\{x: P(x)=0\}$. Hence, to embed "1", $x\in G_1$ is mapped to $y\in G_2$, and to embed "0", the value $x\in G_1$ is left unmodified, i.e., no mapping is performed. This method successfully overcomes the underflow and overflow problems in the conventional histogram shifting methods [24,25]. However, it fails to embed data when the entire range of intensity levels is occupied, i.e., $P(x)>0$ for all $x\in[0,2^{\lambda}-1]$, where λ is a positive integer greater than zero. In such case, [23] partitions the image into non-overlapping blocks and handles each block individually. Furthermore, this method is verified only with images, and it depends on the statistical features of the image in the spatial domain. Hence, it is not feasible to apply [23] for handling a random signal, which consists of high entropy data.

This paper proposes a universal, reversible and scalable data embedding method that is applicable to any random signal based on histogram mapping. Unlike the traditional histogram mapping methods, the proposed method can *certainly* define redundancy in any given signal by applying the universal parser [20], even when all bins in the histogram are occupied, i.e., $P(x)>0$ for all $x\in[0,2^{\lambda}-1]$. In particular, the universal parser recursively partitions the signal into segments using increasing length. Theoretically, we prove that as the length of the segments increases, the probability of defining redundancy (i.e., $\exists x'|P(x')=0$) increases. In other words, the amount of redundancy changes as the length of the segments changes. This change in redundancy is exploited to embed a payload into the random signal by histogram mapping. The proposed data embedding method achieves the following properties: (1) applicable to any random signal; (2) reversible, and; (3) scalable in terms of carrier capacity and progressive quality degradation. To the best of our knowledge, the proposed method is the first of its kind to achieve reversible data embedding in random signal and the first universal data embedding method that offers scalability in carrier capacity and quality degradation.

The rest of this paper is organized as follows: Section 2 describes the possible applications of data embedding in the random domain. Section 3 presents the theoretical study of data embedding using histogram mapping in the random domain. Section 4 proposes DeRand (Data embedding in Random Domain) based on histogram mapping. The results of empirical study of are detailed in Section 5, and Section 6 concludes this paper.

2. Applications of data embedding in random domain

It is assumed that any unintelligible signal which losses its semantic due to intentional encryption or un-intentional loss of access to the original format is called a random signal. This includes signals generated by a random number generator. Hence, random signals occupy a significant percentage among all digital signals exchanged through the communication channels and those stored in the storage systems. Unfortunately, the random signals are, by and large, still left unexplored in the applications of information processing. For that, data embedding in the random domain contributes in filling up this deficiency through some applications, such as data hiding for covert communication (i.e., steganography) in random signals. On the other hand, reversible data embedding enables the insertion of external information while preserving the file size of the original signal. This feature can be exploited to reduce the bandwidth required to communicate/store the random signal by embedding (or hosting) one segment of a random signal into another random signal. Hence, compression is gained and the total size of the random signals is reduced. The reduction in total signal size is a significant contribution in bandwidth utilization, especially for massive data transmission and storage purposes. As another application, a third party (e.g., a cloud administrator who has no access to the original format of the data) needs to embed data in random signals for annotation and management purposes, which include the insertion of a hash value in its corresponding random signal for integrity checking. The aforementioned potential applications justify the need to consider data embedding in the random domain.

3. Theoretical study on histogram mapping

In this section, data embedding by histogram mapping is studied theoretically.

Definition 1. Let $X=\{x_i\}_{i=1}^{N}$ be a finite set of discrete uniformly distributed ordered elements where each element $x_i\in X$ is derived from a finite set of alphabets $\mathbb{A}$, i.e.,

$x_i \in \mathbb{A}$, where N is a positive integer greater than zero. Since X is uniformly distributed, the probabilities $P(x_1) = P(x_1) = \cdots = P(x_N) = 1/N$ hold true. □

Definition 2. The universal parser [20] parses the set X into X_λ by the ordering function $\mathbb{D}(X,\lambda)$ [20], which partitions X into tuples each of length λ as follows:

$$\mathbb{D}(X,\lambda) = X_\lambda = \{T_1^\lambda, T_2^\lambda, \ldots, T_{t=\frac{N}{\lambda}}^\lambda\}, \tag{1}$$

where T_t^λ denotes the t-th λ-tuple that appears in X. □

Definition 3. The set X_λ satisfies $X_\lambda \subseteq \mathbb{A}_\lambda$ where $\mathbb{A}_\lambda$ is the set of all possible arrangements of alphabets with length λ, which has the cardinality of

$$|\mathbb{A}_\lambda| = \underbrace{|\mathbb{A}| \times |\mathbb{A}| \times \cdots |\mathbb{A}|}_{\lambda} = |\mathbb{A}|^\lambda \quad \square \tag{2}$$

Definition 4. The complement set $\dot{X}_\lambda$ of X_λ is defined as follows:

$$\dot{X}_\lambda = \mathbb{A}_\lambda / X_\lambda, \tag{3}$$

where the cardinality $|\dot{X}_\lambda| = |\mathbb{A}_\lambda| - |X_\lambda|$ holds true. □

Based on Definitions 3 and 4, $\mathbb{A}_\lambda = X_\lambda \cup \dot{X}_\lambda$.

Corollary 1. *Given the set $X_\lambda \subseteq \mathbb{A}^\lambda$, the probability of occurrences of the elements in X_λ is computed as*

$$P(X_\lambda) = \sum_{t=1}^{|X_\lambda|} P\left(T_t^\lambda \in X_\lambda\right) = \frac{|X_\lambda|}{|\mathbb{A}^\lambda|}. \quad \square \tag{4}$$

Proof. Since the elements of X are uniformly distributed, the tuples in X_λ are also uniformly distributed. As $X_\lambda \subseteq \mathbb{A}_\lambda$, each tuple assumes that the constant probability of $P(T_t^\lambda) = 1/|\mathbb{A}_\lambda|$. Hence, $\sum_{t=1}^{|X_\lambda|} 1/|\mathbb{A}^\lambda| = |X_\lambda|/|\mathbb{A}^\lambda|$ □

Corollary 2. *Following the previous corollary,*

$$P\left(\dot{X}_\lambda\right) = 1 - \frac{|X_\lambda|}{|\mathbb{A}^\lambda|} \quad \square \tag{5}$$

The histogram of $\mathbb{A}_\lambda$ is denoted by the symbol $\mathbb{H}(\mathbb{A}_\lambda)$ and defined as

$$\mathbb{H}(\mathbb{A}_\lambda) = \{h_{T_t^\lambda}\}_{t=1}^{|\mathbb{A}^\lambda|}, \tag{6}$$

where $h_{T_t^\lambda}$ is the count of the tuple T_t^λ in $\mathbb{A}_\lambda$. To facilitate the discussion without lost of generality, it is assumed that $\mathbb{A}_\lambda$ consists of uniformly distributed elements where $h_{T_t^\lambda} = 1 \forall T_t^\lambda \in \mathbb{A}_\lambda$. That is, each tuple occurs exactly once in $\mathbb{A}_\lambda$.

Let $\mathbb{A} = \{0,1\}$ be the set of alphabets, $d \in \{0,1\}$ be the 1-bit payload to be embedded in a tuple in X_λ, and $\dot{X}_\lambda \neq \phi$. The process of data embedding using histogram mapping is achieved by selecting a tuple $T_t^\lambda \in X_\lambda$ and a tuple $\dot{T}_t^\lambda \in \dot{X}_\lambda$, followed by the mapping:

$$T_t^\lambda \leftarrow \begin{cases} \dot{T}_t^\lambda & \text{if } d = \text{“1”}, \\ T_t^\lambda & \text{otherwise.} \end{cases} \tag{7}$$

Fig. 1(a) presents an example of data embedding using the proposed histogram mapping. Here, $X_2 = \{01, 10, 11\}$, $\dot{X}_2 = \{00\}$ and the frequency of occurrences $\mathbb{H}(\mathbb{A}_2) = \{h_{00} = 0, h_{01} = 1, h_{10} = 1, h_{11} = 1\}$. As an illustration, the tuples $T_1^2 = 01$ and $\dot{T}_1^2 = 00$ are selected from the sets X_2 and $\dot{X}_2$, respectively, for data embedding using histogram mapping. The tuple $T_1^2 = 01$ is mapped to $\dot{T}_1^2 = 00$ to embed “1”, but left unmodified to embed “0”. In other words, X_2 and $\dot{X}_2$ encodes “1” and “0”, respectively. Note that $T_1^2 = 01$ and $\dot{T}_1^2 = 00$ should be saved as side information for perfect restoration of the original X_2.

As another example, Fig. 1(b) shows the histogram $\mathbb{H}(\mathbb{A}_2) = \{h_{00} = 1, h_{01} = 1, h_{10} = 1, h_{11} = 1\}$, where the probability of all tuples are greater than 0. Thus, in this

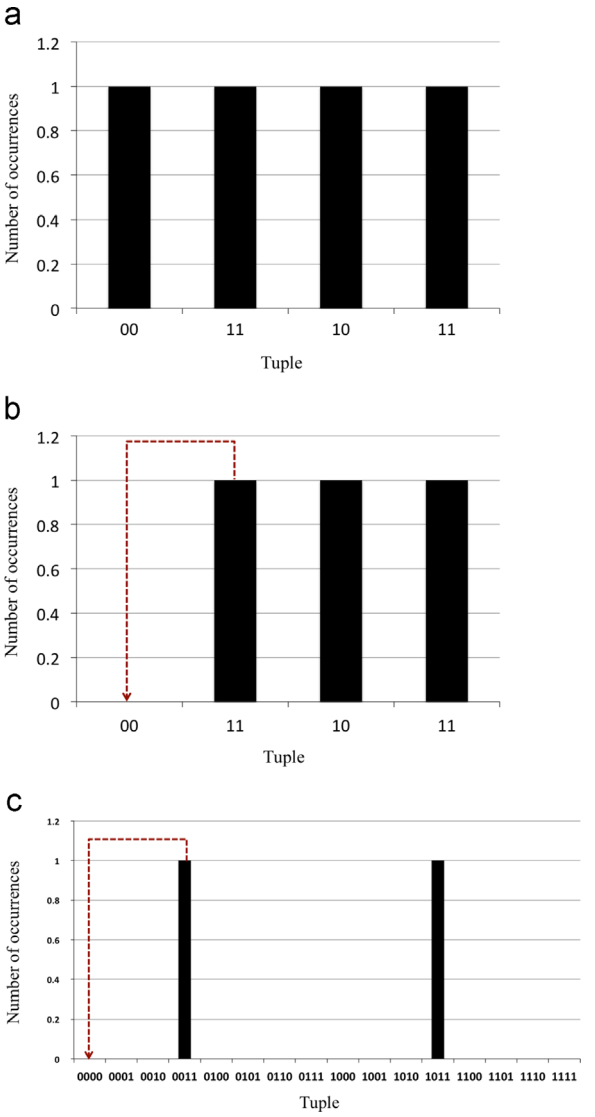

Fig. 1. Example of data embedding using histogram mapping. (a) Since “00” does not exist, it can be utilized for data embedding using histogram mapping. For example, “11” is mapped to “00” to embed “1”, and the value remains intact (i.e., no mapping is performed) to embed “0”. (b) All $2^2 = 4$ bins (possible arrangements) are occupied in the histogram. Hence, data embedding using histogram mapping cannot be achieved directly. (c) Using the universal parser, the elements in (b) are grouped at $\lambda = 4$. Data embedding using histogram mapping can be achieved similar to (a).

example, the cardinality of $\dot{X}_2$ is zero. However, there must be at least one tuple $\dot{T}_t^{\lambda} \in \dot{X}_\lambda$ (i.e., $|\dot{X}_\lambda| \geq 1$) in order to achieve data embedding by histogram mapping. In other words, data embedding by histogram mapping fails when $|\dot{X}_\lambda| = 0$, which is the case in the example shown in Fig. 1(b). To overcome this problem, X_λ is redefined to vacate room in the histogram for mapping. The following theorem shows that increasing λ in the universal parser [20] increases the probability to achieve $|\dot{X}_\lambda| > 0$.

Theorem 1. *Given two sets $\dot{X}_\lambda$ and $\dot{X}_{\lambda+K}$, the following inequality holds true for $K, \lambda > 0$:*

$$P(\dot{X}_\lambda) < P(\dot{X}_{\lambda+K}). \tag{8}$$

Proof. Since $\mathbb{A}_\lambda$ and $\mathbb{A}_{\lambda+K}$ are finite sets, the following holds true:

$$\frac{1}{|\mathbb{A}^\lambda|} > \frac{1}{|\mathbb{A}^{\lambda+K}|}. \tag{9}$$

On the other hand, since X is also a finite set, there are more segments in X_λ when compared to $X_{\lambda+K}$ and hence the following inequality holds true:

$$|X_\lambda| > |X_{\lambda+K}|. \tag{10}$$

Therefore,

$$\left|X_\lambda\right| \times \frac{1}{|\mathbb{A}^\lambda|} > \left|X_{\lambda+K}\right| \times \frac{1}{|\mathbb{A}^{\lambda+K}|}. \tag{11}$$

Hence, based on Corollary 1:

$$P(X_\lambda) > P(X_{\lambda+K}) \tag{12}$$

Now,

$$-P(X_\lambda) < -P(X_{\lambda+K}) \tag{13}$$

and

$$1 - P(X_\lambda) < 1 - P(X_{\lambda+K}) \tag{14}$$

Based on Corollary 2, Eq. (14) can be re-expressed as follows:

$$P(\dot{X}_\lambda) < P(\dot{X}_{\lambda+K}) \qquad \square \tag{15}$$

Informally, Theorem 1 states that the probability of the elements in the complement set $\dot{X}_\lambda$ increases when the original signal X is parsed at a larger λ.

Based on Theorem 1, the histogram mapping in the example shown in Fig. 1(b) is achievable by increasing the value of λ from 2 to 4. Hence, the newly obtained histogram (viz., frequency of occurrences) based on $\lambda = 4$ (see Fig. 1(c)) is $\mathbb{H}(\mathbb{A}_4) = \{h_{0000} = 0, h_{0001} = 0, h_{0010} = 0, h_{0011} = 1, h_{0100} = 0, h_{0101} = 0, h_{0110} = 0, h_{0111} = 0, h_{1000} = 0, h_{1001} = 0, h_{1010} = 0, h_{1011} = 1, h_{1100} = 0, h_{1101} = 0, h_{1110} = 0, h_{1111} = 0\}$. As expected, many tuples from $\mathbb{A}_4$ are of zero count in the histogram of X_4 as shown in Fig. 1(c). Thus, the cardinality of the set $\dot{X}_4$ is 14. Hence, for the set $X = \{00, 01, 10, 11\}$, histogram mapping is feasible at X_4 but impossible at X_2. The next theorem shows that data embedding by histogram mapping can be *certainly* achieved in X when it is parsed at the maximum length $\lambda = N$, i.e., the limiting case of $X_{\lambda=N}$.

Theorem 2. *A finite set X with N elements induces the set X_N using Eq. (1) (i.e., $\mathbb{D}(X, N)$) and the inequality $|\dot{X}_N| \geq 1$ holds true.*

Proof. Since the cardinality of the set of alphabets $|\mathbb{A}| \geq 2$, the cardinality below holds true:

$$|\mathbb{A}_N| = |\mathbb{A}|^N \geq 2^N. \tag{16}$$

Since, $|\dot{X}_N| = |\mathbb{A}_N| - |X_N|$ and the cardinality $|X_N| = 1$ (based on Eq. (1)), the following holds true:

$$|\dot{X}_N| = |\mathbb{A}_N| - 1 \geq 2^N - 1 \geq 1 \tag{17}$$

for the fact that N is an integer with $N > 0$. Therefore, $|\dot{X}_N| \geq 1$. $\square$

Theorem 2 ensures that histogram mapping is always viable in any set X if the set is parsed at length N to X_N using Eq. (1). Informally, this theorem shows that the histogram mapping can be universally applied to any discrete signal X regardless of its media, domain, underlying coding structure and statistical features.

4. The proposed DeRand

In this section, the proposed DeRand (Data embedding in Random domain) method is presented. Fig. 2 presents the schematic diagram of the proposed DeRand. In Stage 1, the input signal X is parsed by the universal parser [20] to generate X_λ. In Stage 2, the tuple association is performed, which includes the construction of histograms for X_λ and $\dot{X}_\lambda$. In Stage 3, histogram mapping is applied based on the input payload D. The following subsections further detail the operations involved in each stage.

4.1. Parsing by using Universal Parser [20]

The input signal X is parsed by using the universal parser [20] defined in Eq. (1). Here, the user is prompted to select an initial value of λ. However, if the selected λ results in $|\dot{X}_\lambda| = 0$, the value of λ is increased to $\lambda \leftarrow \lambda + 1$ (based on Theorem 1) and the parsing process is performed again. In other words, λ is increased repeatedly until $|\dot{X}_\lambda| > 0$ is achieved. This stage is captured by steps 2–14 in Algorithm 1. Ideally, the initial value of λ is unity. However, it is less probable to obtain $|\dot{X}_\lambda| > 0$ with a small value of λ. Hence, it is recommended to initiate the algorithm at $\lambda = 7$ based on the empirical findings presented in Section 5.

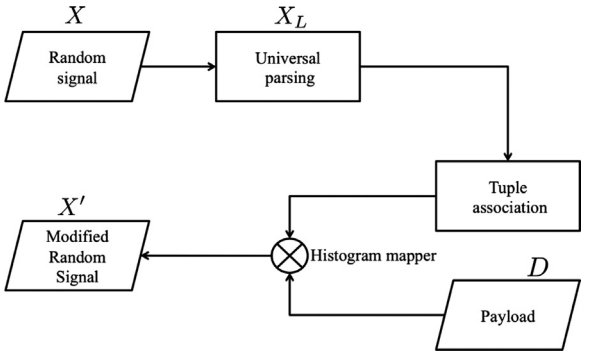

Fig. 2. The schematic diagram of the proposed DeRand.

4.2. Tuples association

Once the signal X is parsed at a length λ such that $|\dot{X}_\lambda| > 0$, the process of tuple association takes part. This includes the construction of two sets, namely, A_λ and B_λ, where $1 \le |A_\lambda|, |B_\lambda| \le \min\{|X_\lambda|, |\dot{X}_\lambda|\}$. The set $A_\lambda = \{T^\lambda_{P(i)}\}_{i=1}^{\rho} \subseteq X_\lambda$ consists of ρ tuples that are of the highest occurrences in $\mathbb{H}$, where ρ is a positive integer such at $\rho \le \hat{\rho}$ for $\hat{\rho} = \min\{|X_\lambda|, |\dot{X}_\lambda|\}$. Here, $P(i)$ is the index sorted accordingly to the probability of occurrence of $T^\lambda_{P(i)}$ in $\mathbb{H}$ such that $T^\lambda_{P(1)}$ has the highest occurrence among the tuples in X_λ, followed by $T^\lambda_{P(2)}$, and so forth. On the other hand, the set $B_\lambda = \{\dot{T}^\lambda_i\}_{i=1}^{\rho} \subseteq \dot{X}_\lambda$ consists of any ρ tuples from the set $\dot{X}_\lambda$. Next, a tuple $T^\lambda_{P(i)} \in A_\lambda$ is associated with a unique tuple $\dot{T}^\lambda_i \in B_\lambda$. For example, if $X_2 = \{00, 11\}$, then these two tuples are associated with the two tuples in $\dot{X}_2 = \{01, 10\}$. A possible association is "00" with "01", and "11" with "10". The process of tuple association is achieved by invoking steps (12)–(15) in Algorithm 1. This association is utilized for data embedding as detailed in the next sub-section.

For scalability purposes, $\rho \in [1, \hat{\rho}]$ is selected by the user. Here, as ρ increases, more tuples from X_λ are utilized for data embedding and hence the embedding capacity increases. However, increasing ρ causes more distortion (e.g., sum of bit-wise absolute difference) in the modified signal. Therefore, there is a trade-off between the embedding capacity and distortion, similar to the conventional data embedding method.

Algorithm 1. Embedding data in DeRand.

1: Let the user pick $\lambda \in [1, N]$
2: Set $i \leftarrow 1$ and $t \leftarrow 1$
3: Generate $X_\lambda = \mathbb{D}(X, \lambda)$
4: Construct $|\dot{X}_\lambda|$
5: **if** $|\dot{X}_\lambda| = 0$ **then**
6: $\lambda \leftarrow \lambda + 1$
7: Go to Step 4
8: **else**
9: Go to Step 15
10: **end if**
11: Let the user pick $\rho \in [1, \hat{\rho}]$
12: Construct the sets $A_\lambda = \{T^\lambda_{P(i)}\}_{i=1}^{\rho} \subseteq X_\lambda$ and $B_\lambda = \{\dot{T}^\lambda_i\}_{i=1}^{\rho} \in \dot{X}_\lambda$
13: Read one tuple T^λ_t from X_λ
14: **if** $T^\lambda_t = T^\lambda_{P(i)} \in A_\lambda$ **then**
15: Read one bit d from payload D
16: **if** $d = 1$ **then**
17: $T^\lambda_t \leftarrow \dot{T}^\lambda_i$
18: **end if**
19: **end if**
20: $t \leftarrow t + 1$
21: $t > \frac{N}{\lambda}$ **then**
22: Halt
23: **else**
24: Go to Step 13
25: **end if**

Algorithm 2. Extraction of payload and restoration of X in DeRand.

1: Set $i \leftarrow 1$
2: Restore $N, \lambda, \rho, A_\lambda$ and B_λ from side information (Section 4.3)
3: Define the associated tuples from A_λ and B_λ
4: Divide X' into segments of unified length λ
5: Read tuple τ_i
6: **if** $\tau_i = T^\lambda_t \in A_\lambda$ **then**
7: Extract "0" from payload
8: Go to Step 15
9: **end if**
10: **if** $\tau_i = \dot{T}^\lambda_t \in B_\lambda$ **then**
11: Extract "1" from payload
12: Map $\tau_i = \dot{T}^\lambda_t$ to its associated tuple T^λ_t
13: Go to Step 15
14: **end if**
15: $i \leftarrow i + 1$
16: **if** $i > \frac{N}{\lambda}$ **then**
17: Halt
18: **else**
19: Go to Step 5
20: **end if**

4.3. Data embedding

The process of data embedding commences by reading a tuple T^λ_t. Then, it is verified if $T^\lambda_t = T^\lambda_{P(i)} \in A_\lambda$ holds true for $1 \le t \le N/\lambda$. Next, to embed $d = \text{"1"} \in D$, the tuple T^λ_t is mapped to $\dot{T}^\lambda_i$, where D is the payload in binary representation. In order to embed $d = \text{"0"} \in D$, no mapping is performed. This process continues until all the N/λ tuples of X_λ are mapped or when all bits in D are embedded. The process of tuple association is achieved by invoking steps (13)–(20) in Algorithm 1. Note that N, λ, ρ, A_λ and B_λ should be stored as the side information for perfect reconstruction of the original signal X_λ. For the current implementation, the signal X is concatenated to the side information S, i.e., $X \leftarrow S \oplus X$. For file-size preserving purposes, $|S| = N + \lambda + \rho + |A_\lambda| + |A_\lambda|$ bits of the original input signal, i.e., the number of bits occupied by S, is removed and embedded as part of the payload.

4.4. Extraction of payload and restoration of X

The payload extraction and reconstruction of the original signal X are performed by Algorithm 2. Initially, $N, \lambda, \rho, A_\lambda$ and B_λ are restored from the side information as discussed in Section 4.3. Next, $\rho \times 2$ associated tuples are defined by reading each tuple T^λ_t and its corresponding (viz., associated) tuple $\dot{T}^\lambda_t$ from A_λ and B_λ, respectively. The modified signal X' is then segmented into tuples each of λ bits, and each tuple τ is analyzed. In particular, if $\tau_i = T^\lambda_t \in A_\lambda$, then "0" is extracted. On the other hand, if $\tau_i = \dot{T}^\lambda_t \in B_\lambda$, then "1" is extracted and this tuple is restored to its associated T^λ_t. When $\tau_i \notin A_\lambda$ and $\tau_i \notin B_\lambda$, it implies that no information is embedded. Regardless of the membership of τ_i, the decoder continues to read the next tuple that follows and repeat the aforementioned analysis until all N/λ tuples are considered.

4.5. Example of data embedding using DeRand

In this section, an example is presented to walkthrough the proposed data embedding and extraction processes.

Assume that $X=\{0001\}$ is the input signal and $\lambda=2$. Then, the set $X_2=\{00,01\}$ is obtained by applying the universal parser. Here, the cardinality $\rho=|X_2|=\frac{4}{2}=2$. In other words, X_2 consists of two tuples, namely, $T_1^2=00$ and $T_2^2=01$. Let $A_2=X_2=\{00,01\}$ and hence the set $B_2=\dot{X}_2=\{10,11\}$. Then, the process of tuple association is carried out. As an illustration, the tuple $T_1^2=00$ is associated with $\dot{T}_1^2=10$, and the tuple $T_2^2=01$ is associated with $\dot{T}_2^2=11$. It should be noted that a tuple in A_2 can be associated with any tuple in B_2 as long as the association is consistent throughout the embedding and extraction processes. To ensure reversibility, the association map, $\lambda=2$, A_2, and B_2 are stored as side information. Next, suppose $D=\{1,0\}$ is the payload. In order to embed D into X_2, the proposed histogram mapping is applied. Specifically, the first bit in D, i.e., "1", is embedded in $T_1^2=00$. The embedding is achieved by mapping $T_1^2=00$ to its associated tuple $\dot{T}_1^2=10$. On the other hand, the second bit in D, i.e., "0", is embedded into $T_2^2=01$, where no mapping is performed to indicate that "0" is accommodated in $T_2^2=01$. Hence, the resulting signal is $X_2'=\{10,01\}$.

To extract D and restore the original signal X_2, the aforementioned side information are restored. Then, each tuple in $X_2'=\{10,01\}$ is verified. In particular, since the tuple $10\in B_2$, "1" is output as the embedded information D_1, followed by the inverse mapping of "10" to its corresponding associated tuple "00". Similarly, since $01\in A_2$, "0" is output as D_2. However, no inverse mapping is performed. Hence, the embedded information $D=\{D_1,D_2\}=\{1,0\}$ and the original signal $X_2=\{00,01\}$ are obtained.

5. Experimental results and discussion

In this section, the performance of DeRand is studied empirically. Ten random sequences of integers in the range of $[0,255]$ are generated as the random signals for experiment purposes. For the rest of the discussion, the phrase *random test signal* refers to the randomly generated test

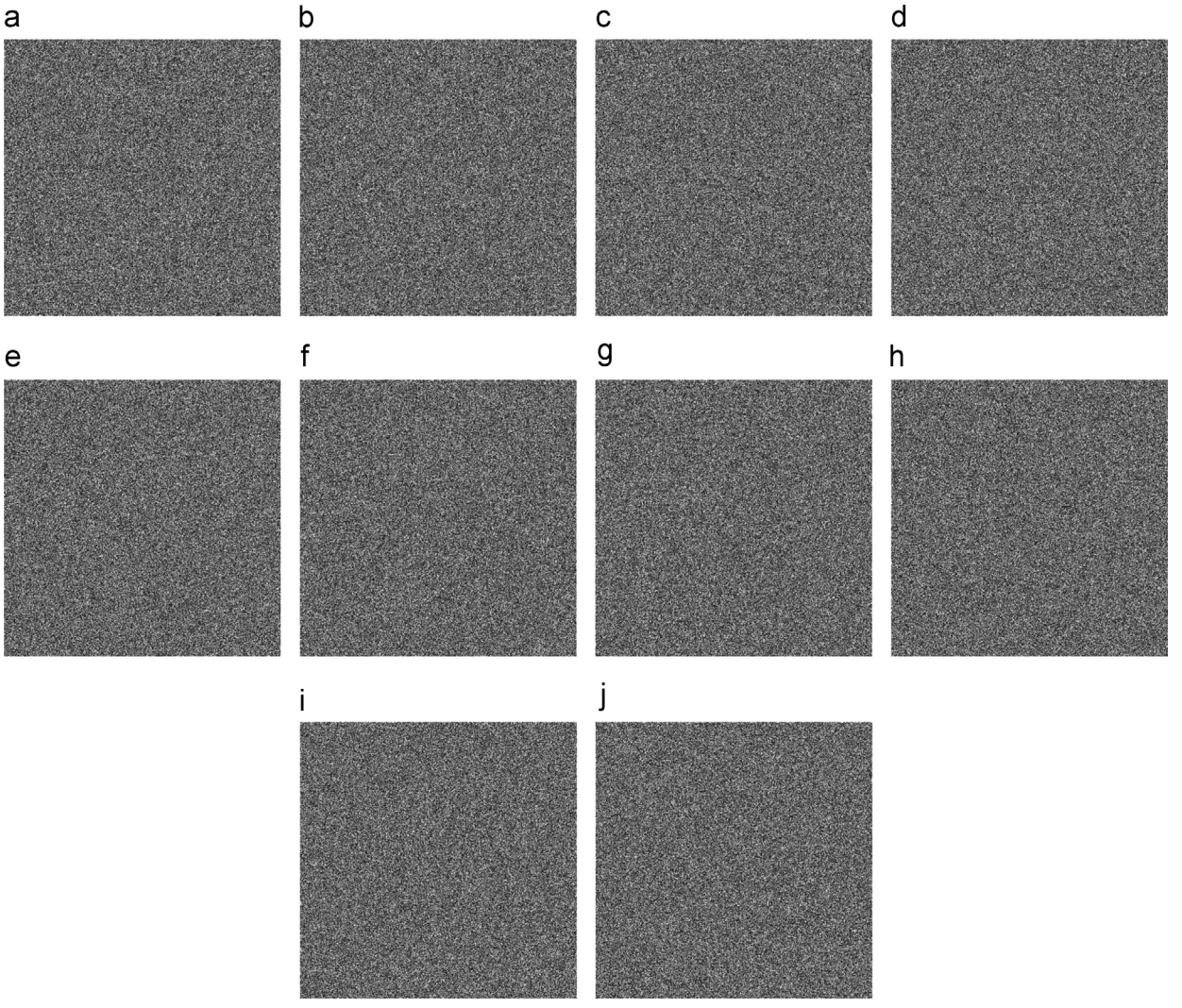

Fig. 3. The set of 10 random signals each visualized as an 8-bit image. These random signals are generated by using Matlab function "randn", which is based on the ziggurat method for random variables generation [26]. (a) R1. (b) R2. (c) R3. (d) R4. (e) R5. (f) R6. (g) R7. (h) R8. (i) R9. (j) R10.

signal. Each random test signal is visualized as an image in Fig. 3 using the resolution of 512×512 pixels. For additional comparison purposes, DeRand is applied to the standard test image Lenna, which act as the representative non-random test signal. The experiments are carried out using Matlab (version 7.12.0.635-R2011a) operating on OS X 10.9.2 (13C1021) platform and the performance of DeRand is evaluated based on three criteria, namely, embedding capacity, signal quality, and file-size increment.

5.1. Embedding capacity

Hereinafter, embedding capacity C refers to the number of bits from the set D embeddable into the host signal X. The amount of C indicates the efficiency of DeRand in defining redundancy (or venue) in X to accommodate D. Here, C depends on three factors, namely, the statistical features of X, λ and ρ.

First, we consider the results when setting $\rho=1$, viz., only one tuple (i.e., T_1^{λ}) is allowed to be mapped to $\dot{T}_1^{\lambda}$. When using Lenna as the representative non-random test signal, $C=1201$, 1532, 419 and 67 bits are achieved for $\lambda=$ 9, 10, 13 and 22, respectively. Here, C is low (with respect to the size of the signal $|X|$) because only one tuple is allowed for mapping. In addition, the side information for $\rho=1$ consumes $|S|=|T_1^{\lambda}|+|\dot{T}_1^{\lambda}|+\lambda+N+\rho$ bits of the embedding capacity. To facilitate the discussion, the effective embedding capacity E is defined as follows:

$$E = C-(|A_{\lambda}|+|B_{\lambda}|+\lambda+N+\rho). \tag{18}$$

Results reveal that E is 1174, 1502, 380 and 1 bits when $\lambda=$ 9, 10, 13 and 22, respectively, for the case of Lenna. Nonetheless, it is observed that the low effective embedding capacity (e.g., $E=1$ bit at $\lambda=22$) is a general trend in all test signals when they are parsed at large λ's. In particular, the number of occurrences of T_1^{λ} in the finite set X_{λ} reduces when λ increases. Hence, the number of mapped tuples and C decrease when λ increases.

On the other hand, the effective embedding capacity E increases when ρ increases. Table 1 records the value of E attained by setting ρ to different values when $\lambda=8$ and 13. For example, $E=380$ at $\rho=1$ as reported earlier. However, the effective embedding capacity is increased to $E=13{,}217$ bits by setting $\rho=50$. In general, E consistently increases when ρ increase as suggested by Table 1. Similar trend is observed when ρ increased for different λ. In order to study the trend of E due to increasing ρ, Fig. 4(a) plots $\Delta_E(\rho)=E(\rho+1)-E(\rho)$, where $\Delta_E(\rho)$ refers to the effective embedding capacity when using the parameter ρ. Here, $\Delta_E(\rho)$ generally decreases when ρ increases. This is because, as ρ increases, the newly utilized tuples for histogram mapping are of small number of occurrences in $\mathbb{H}$. At the same time, the side information increases as suggested by Eq. (18). For that, E is small for large ρ. This suggests that the complete utilization of the range $\rho\in[1,\hat{\rho}]$ (e.g., by setting $\rho=\hat{\rho}$) is not required to obtain high E because as ρ approaches to $\hat{\rho}$, E increases insignificantly.

Among the parameters considered, the highest effective embedding capacity of $E=106{,}216$ bits is achieved with $(\lambda,\rho)=(8,40)$ as reported in Table 1. Note that $E(8,40)>E(13,1050)$. This is because, the number of mapped tuples at $\lambda=8$ is significantly larger than that of $\lambda=13$. The results in Table 1 also suggest that E is scalable, depending on the parameter values of (λ,ρ).

Table 1
The effective embedding capacity E and SSIM of the test image Lenna achieved by using various values of ρ with $\lambda=8$ and $\lambda=13$.

λ	ρ	E (bits)	SSIM
8	10	35,114	0.5604
	20	60,187	0.2923
	30	83,933	0.2471
	40	106,216	0.2439
13	50	13,217	0.4038
	100	31,986	0.2578
	150	38,557	0.2245
	200	43,776	0.2062
	250	47,894	0.1917
	300	46,594	0.1917
	350	49,955	0.1792
	400	52,929	0.1706
	450	59,208	0.1536
	500	61,331	0.1459
	550	63,192	0.1368
	600	64,877	0.1313
	650	66,402	0.1266
	700	67,760	0.1217
	750	69,012	0.1186
	800	70,163	0.1145
	850	71,217	0.1099
	900	72,177	0.1075
	950	73,036	0.1038
	1000	73,814	0.0999
	1050	74,526	0.0964

Next, for the random test signals, the effect of the high entropy nature of these signals (i.e., low redundancy) on C and E is apparent. For example, it is observed in Fig. 5(a) that as λ increases, the average value of C decreases generally, where the horizontal axis presents the average C for each λ considered. Here, maximum of the averages of C (i.e., 15 bits) is obtained when $(\lambda,\rho)=(14,1)$. Note that $|\dot{X}_{\lambda}|=0$ for $\lambda<14$ due to the high entropy nature of the random signals. Similar to the Lenna image, C of the random signals also increase when ρ increases. Table 2 records C for various values of ρ using R5 as the representative test random signal parsed at $\lambda=15$. Results suggest that, by increasing ρ from 1 to 50, C increases from 10 to 417 bits. Table 2 shows that C increases up to 3369 bits when $\rho=500$. However, this increment in C results in file size increment. This is because, in general, $E<0$ (see Eq. (18)) in the random signals. In other words, the size of the side information is always larger than C. Thus, C is insufficient to embed the side information along with the payload D. To realize data embedding in such case, the side information is appended to the modified signal, therefore causing file-size increment.

On the other hand, it is observed that $C(16,\rho)\geq C(15,\rho)$ when considering the same values of ρ. For example, Table 2 shows that $C(16,500)=4909\geq C(15,500)=3369$. This trend holds true for all values of ρ at $\lambda=16$ and 15. This is because the random signals are generated in the range $[0,255]$, i.e., 8-bit integer. Hence, the probability of having similar tuples at $\lambda=8\alpha$ (i.e., multiple of 8 such as 16, 24,32, …) is high. On the other hand, it is observed that

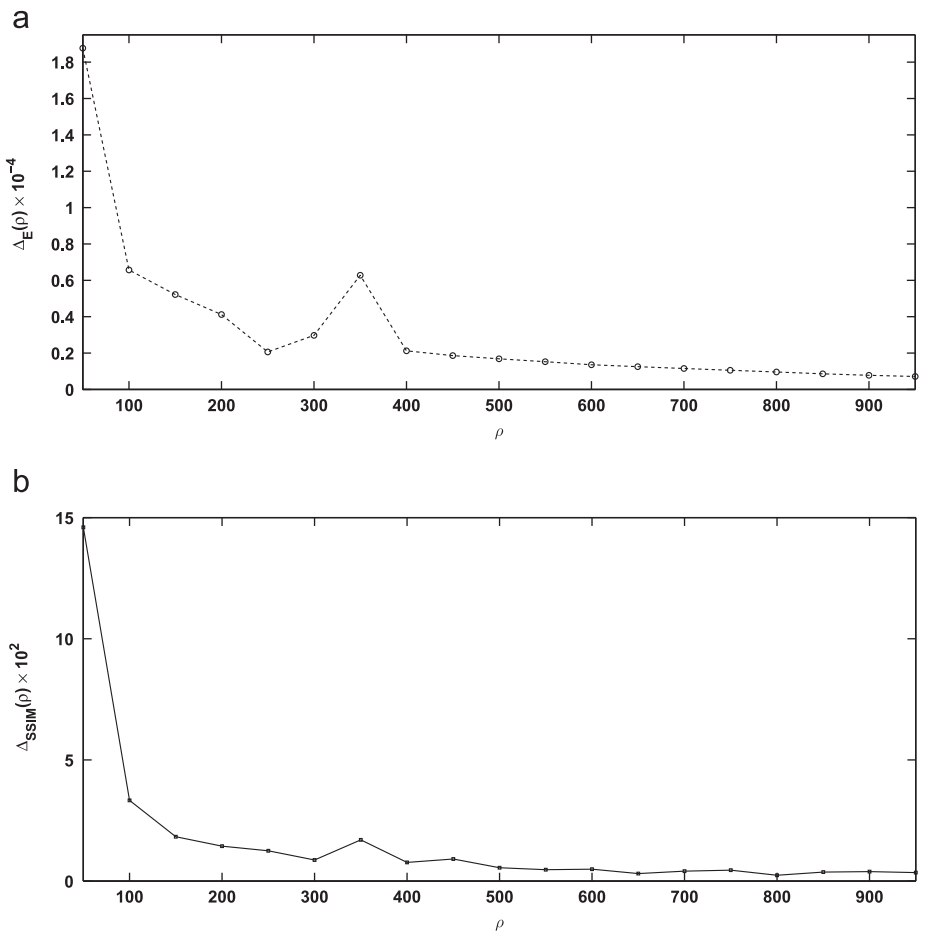

Fig. 4. Change in effective embedding capacity $\Delta_E(\rho)$ and visual quality $\Delta_{SSIM}(\rho)$ for the test image Lenna parsed at $\lambda = 13$. (a) $\Delta_E(i)$. (b) $\Delta_{SSIM}(\rho)$.

the amount of increment in C decreases as ρ increases. Fig. 6(a) shows the plot of $\Delta_C(\rho) = C(\rho+1) - C(\rho)$ using R5 as the representative random test signal parsed at $\lambda = 16$, where $C(\rho)$ refers to the embedding capacity when using the parameter ρ. Generally, as ρ increases, the increment in C decreases. For example, at $\lambda = 16$, as ρ increases from 50 to 100, C increases from 623 to 1173 bits. However, as ρ increases from 300 to 350, C merely increases from 3210 to 3710 bits. This is because, as ρ increases, the newly utilized tuples for histogram mapping have low number of occurrences in the histogram of X_λ. Similar trend is also observed for the other nine random test signals, and we omit the discussion here.

In Table 3, the embedding capacities achieved by applying the proposed DeRand to encrypted signals are compared with those reported in [20]. In particular, DeRand is applied to the standard test image Elaine, which is encrypted by using 10 different encryption schemes. These schemes include AES-256, DES-56, Blowfish-448, Twofish-256, Seprent-256, TEA-128, RC6-2048, ICE-64, MARS-1248 and MISTY-128. The details of these encryption schemes can be found in [20]. Table 3 suggests that [20] achieves higher embedding capacity when compared to DeRand. Specifically, for encrypted signal, the average embedding capacity of [20] is 349,058 bits, but the average embedding capacity of DeRand is 5782 bits. This is because, [20] exploits entropy coding to achieve data embedding in the encrypted signal and hence more redundancy (i.e., data embedding venues) are defined. However, the entropy coding in [20] causes severe distortion in the modified signal. In other words, [20] transforms the encrypted signal into a completely different form (i.e., from that of the original encrypted signal) due to data embedding. In contrast, the proposed method is able to control the distortion and carrier capacity as detailed in Section 5.2. More importantly, [20] fails to locate venues for data embedding when handling random signals (due to large side information – see Table 4), which is the targeted class of signal in this work.

5.2. Signal quality

In this section, the distortion in the modified signal due to the application of DeRand is evaluated. First, we consider Lenna as the representative non-random test signal. Here, we emphasize that SSIM is considered merely as a mean to visualize the scalability feature of DeRand in terms of output signal quality. It is observed that the

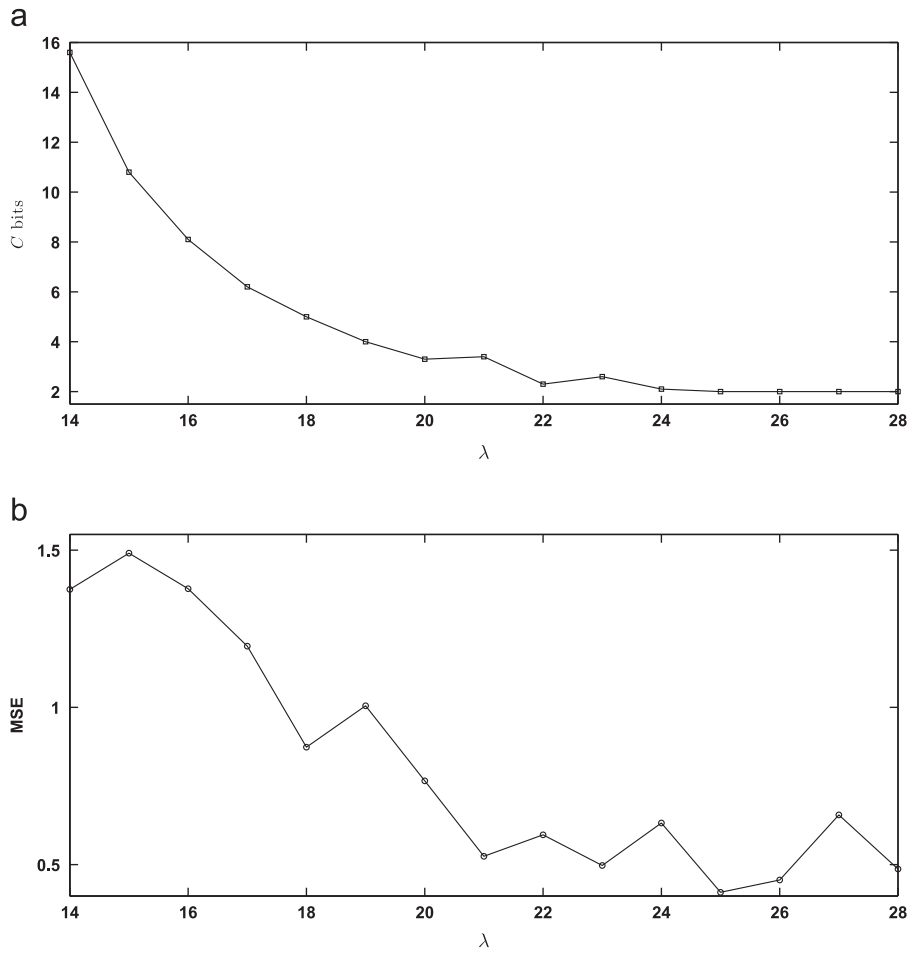

Fig. 5. The capacity (bits) and MSE of the random test signals. (a) The average capacity of 10 test (random) signals for $\rho=1$ versus λ. (b) The average MSE of 10 test (random) signal for $\rho=1$ versus λ.

Table 2
Embedding capacity C, MSE and the percentage of file size increment for the representative random test signal R5 achieved by using various values of ρ with $\lambda=15$ and $\lambda=16$.

λ	ρ	C (bits)	MSE	File-size increment (%)
	50	417	50.81	0.08
	100	783	90.07	0.15
	150	1133	112.66	0.23
	200	1483	149.02	0.30
15	250	1833	202.53	0.37
	300	2169	265.10	0.44
	350	2469	273.85	0.51
	400	2769	291.76	0.58
	450	3069	305.14	0.65
	500	3369	324.26	0.73
	50	623	56.64	0.09
	100	1173	94.19	0.16
	150	1710	134.18	0.24
	200	2210	169.64	0.32
16	250	2710	209.00	0.39
	300	3210	247.30	0.47
	350	3710	281.54	0.55
	400	4210	315.15	0.62
	450	4666	364.92	0.70
	500	4909	395.67	0.78

distortion is generally subtle (i.e., high SSIM value) when $\rho=1$. This is expected because only one tuple in X_λ is utilized for histogram mapping while the remaining tuples remain unchanged. For example, at $\lambda=9$, 10, 13 and 22, the corresponding SSIM values are 0.8775, 0.8299, 0.9087 and 0.9744, respectively. Here, the change in quality depends on λ. In particular, when λ is small, the number of tuples in X_λ (and consequently the number of modified tuples) is high. For that, high distortion occurs in the modified image, such as the aforementioned cases of $\lambda=$ 9 and 10. However, as λ increases, the number of tuples in X_λ decreases and hence less tuples are modified. For that, at a relatively large λ (e.g., $\lambda=13$ and 22), SSIM is high, indicating that the distortion is less severe for larger λ.

On the other hand, the distortion is more severe as ρ increases. The last column of Table 1 records the SSIM attained by different ρ values for Lenna parsed at $\lambda=13$. Here, by increasing ρ from 1 to 50, SSIM decreases from 0.9087 to 0.4038. Here, SSIM decreases because more tuples are mapped (i.e., modified) when ρ increases. Generally, as ρ increases, the degree of distortion increases, where the most severe distortion occurs when $(\lambda,\rho)=(13,1050)$ with SSIM=0.0964 as recorded in Table 1. Here, the value λ defines the maximum possible distortion. For example, in

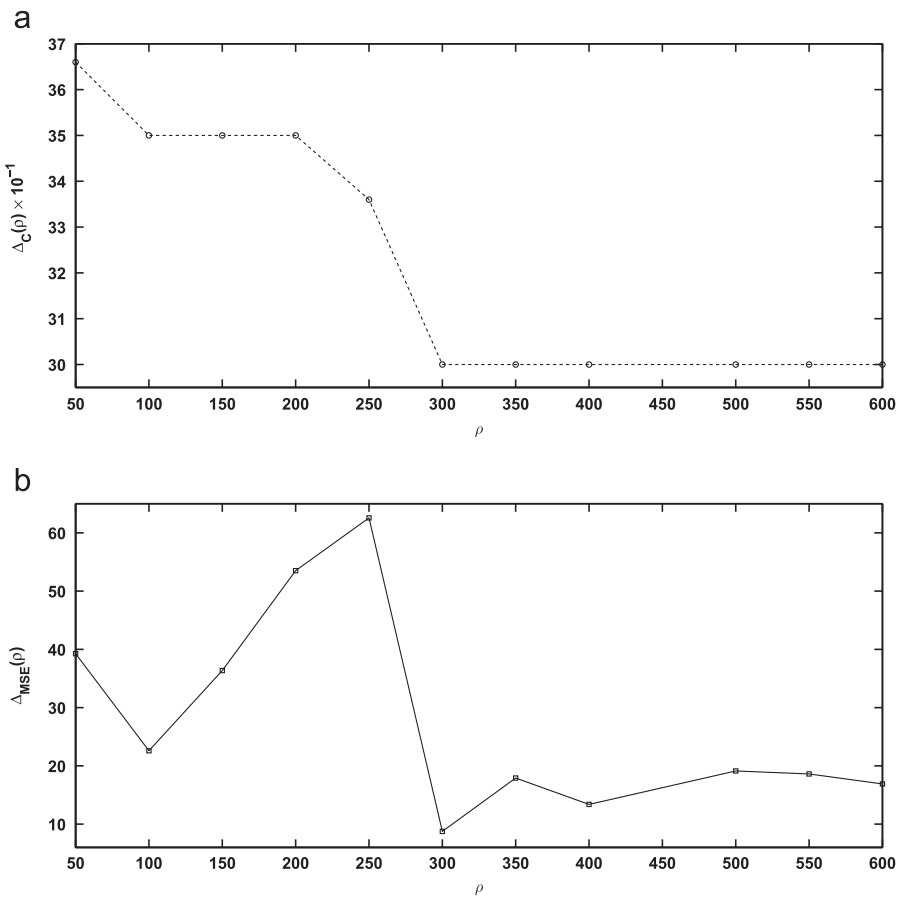

Fig. 6. Change in capacity $\Delta_C(\rho)$ and visual quality $\Delta_{MSE}(\rho)$ for the representative random test signal R5 parsed at $\lambda = 16$. (a) $\Delta_C(\rho)$. (b) $\Delta_{MSE}(\rho)$.

Table 3
Comparison in terms carrier capacity (in bits) between Method [20] and the proposed DeRand using (λ, ρ)=(16,900).

Encryption scheme	Method [20]	DeRand
AES-256	336,221	5685
DES-56	335,700	5652
Blowfish448	335,809	5656
Twofish-256	400,621	6232
Seprent-256	400,422	6199
TEA-128	336,019	5692
RC6-2048	336,445	5652
ICE-64	336,440	5704
MARS-1248	336,901	5642
MISTY-128	335,999	5707

Table 1, at $\lambda = 8$, the lowest SSIM (viz., maximum distortion) is 0.2439 when $\rho = 40$. This is because, at $\lambda = 8$, it is impossible to increase ρ to more than 40 due to the fact that $|\dot{X}_8| = 40$. Fig. 4(b) shows the plot of $\Delta_{SSIM}(\rho) = SSIM(\rho) - SSIM(\rho+1)$, where $\Delta_{SSIM}(\rho)$ refers to SSIM when using the parameter ρ. Here, similar to the effective embedding capacity E, $\Delta_{SSIM}(\rho)$ decreases as λ increases for the same reason mentioned in the previous section. Fig. 7 shows the output Lenna image produced by DeRand. Here, the distortion is controlled, i.e., scalable, by using the parameters λ and ρ. In particular, DeRand offers a wide range of distortion by setting the aforementioned parameters. Specifically, the SSIM value ranges from 0.038 using (λ, ρ)=(16,5000) to 0.975 using (λ, ρ)=(22,1). It should be noted that this range of progressive degradation in image quality is wider than that attained by [23], which spans the interval of [0.040, 0.771].

Secondly, the random test signals are considered for data embedding. Instead of SSIM, the signal quality is measured by MSE (mean square error). Here, the random test signals behave similar to the Lenna test image when $\rho = 1$, where the average MSE of all random test signals decreases as λ increases as shown in Fig. 5(b). Here, the maximum of the averages of *MSE* is 1.5 and it is obtained when (λ, ρ)=(14,1). This suggests that the modified image resembles its original counterpart. Among the 10 randomly generated test signals, it is observed that the minimum MSE is 0.0075, which is obtained with R6 at $(\lambda, \rho) = (23, 1)$. This is because, in general, only small number of tuples in the random test signal are mapped (regardless the value of λ), which leads to limited distortion.

On the other hand, as shown in Table 2, MSE increases from 1.5 to 50.81 in the random test signal R5 (parsed at $\lambda = 15$) when ρ increases from 1 to 50. This increment in MSE is directly proportional to the increment of C due to

Table 4
Functional comparison between the proposed DeRand and the methods [20,23].

	DeRand	Method [20]	Method [23]
Reversibility	Yes	Yes	Yes
Universal	Yes	Yes	No
Scalable embedding capacity?	Yes	No	Yes
File-size preserving (non-random signals)	Yes	Yes	Yes
File-size preserving (random signals)	No ($<0.42\%$)	Failed to embed	Failed to embed
Progressive quality degradation?	Yes	No	Yes
Distortion range (SSIM) (non-random signals)	[0.038,0.975]	–	[0.040,0.771]
Distortion range (MSE) (random signals)	[0.0075,395.67]	Failed to embed	Failed to embed
Applicable to random signals	Yes	No	No
Number of passes required to achieve data embedding	2	3	2
Range of side information (non-random signals)	$[3\lambda+\rho+N, \lambda\times 2^{\lambda}+\rho+N]$	$\lambda\times 2^{\lambda}$	$[5\lambda, (2\lambda+\lvert A_{\lambda}\rvert+\lvert B_{\lambda}\rvert)\times \mathcal{B}]$

the increment of ρ, and the maximum distortion attained is MSE=395.67 at $(\lambda,\ \rho)=(16,500)$. Fig. 4(b) shows the graph of $\Delta_{\mathrm{MSE}}(\rho)=MSE(\rho+1)-MSE(\rho)$, where $\Delta_{\mathrm{MSE}}(\rho)$ refers to MSE when using the parameter ρ. It is observed that $\Delta_{\mathrm{MSE}}(\rho)$ increases when $\rho\le 250$ because many tuples are utilized for histogram mapping. However, $\Delta_{\mathrm{MSE}}(\rho)$ shows no significant change for $\rho\ge 300$. This is because, the tuples are of low number of occurrences for $\rho\ge 300$. Hence, the number of tuples utilized for histogram mapping is low. For that, the variation in distortion, i.e., $\Delta_{\mathrm{MSE}}(\rho)$, is smaller than those at $\rho\le 250$. This suggests that the increase in distortion is higher for smaller ρ, and vice versa. Similar trends are observed in other random test signals.

5.3. File-size preserving

File-size preserving refers to the scenario where the size of the modified signal is similar to that of its original counterpart. This feature is important when the bandwidth of the communication channel is limited. In DeRand, this feature is conditional. Particularly, for the test image Lenna (i.e., non-random signal), DeRand completely preserves the file-size because all the side information can be embedded along with the payload. For random signal, DeRand increases the file-size because there is no venue to embed all the side information. In other words, only concatenation $X\leftarrow S\oplus X$ is performed without removing $|S|$ bits from the original input signal X. However, the file-size increment is insignificant. Table 2 records the percentage of file-size increment for R5 as the representative random test signal. Here, for $(\lambda,\rho)=(15,50)$, the file-size is increased merely by 0.08%. This percentage increases gradually as ρ increases and it reaches $\sim 0.73\%$ for $\rho=500$. The increment in file-size for $\lambda=16$ is slightly higher than that of $\lambda=15$. For example, when $\rho=500$, the file-size is increased by $\sim 0.78\%$ for $\lambda=16$, which is 0.05% higher than that of $\lambda=15$ using the same ρ. This slight increment is due to the side information, which is proportional to λ as expressed by Eq. (18). Generally, the average of the file-size increment in Table 2 is only 0.42%.

5.4. Functional comparison with existing methods

For completion of discussion, Table 4 shows a functional comparison between the proposed DeRand and the methods in [20,23]. All three methods are reversible. DeRand and [20] are universal, but [23] is only applicable to image represented by an array of pixel values. DeRand and [23] offer scalability in terms of embedding capacity and quality degradation. However, [20] is not scalable and the host is completely distorted due to data embedding. On the other hand, [20,23] also fail to embed any data in random signal. However, by design, DeRand achieves its objective of embedding data in random signals. In addition, DeRand is file-size preserving for non-random signals and only increases the file-size insignificantly (in average 0.42% only) for random signals as detailed in Section 5.3. Furthermore, when handling image (i.e., non-random signal), DeRand offers a range of distortion with SSIM spanning the interval of $[0.038, 0.975]$, which is wider than the interval of $[0.040, 0.771]$ attained by Method [23]. This suggests that DeRand offers more scalability in terms of quality degradation. On the other hand, DeRand can control the distortion of the modified random test signals and the observed range is $[0.0075, 395.67]$ in terms of MSE. However, we could not study the range of the quality degradation in [20,23] when they are applied to the random signals because these methods fail to embed data. It should be noted that more distortion (than those reported in Table 1) could be attained theoretically by applying DeRand. However, we could not achieve a lower signal quality (i.e., more severe distortion) empirically due to the limitation in memory. In particular, the histogram of a signal parsed at $\lambda>16$ could not be handled since there are more than 2^{16} bins in the histogram to consider. Nonetheless, we should recall that SSIM is merely used as a mean to illustrate the scalable capability of DeRand, and the targeted signals of this study are the random signals. Pertaining the computational complexity, as a mean for comparison, we consider the number of passes to infer the algorithmic complexity. Here, when random signals are considered, DeRand is the only method that can embed data into such type of signal, and hence no comparison is needed when considering random signal. On the other hand, when encrypted signals are considered, DeRand is superior to [20] in terms of complexity. Specifically, for data embedding, DeRand parses the host for 2 times while [20] parses the host for 3 times. The additional pass in [20] is due to its entropy coding process. Hence, it is expected that the execution time for DeRand is shorter than that of [20]. When non-random or un-encrypted (i.e., perceptually meaningful) signals are considered, both [23] and DeRand require two

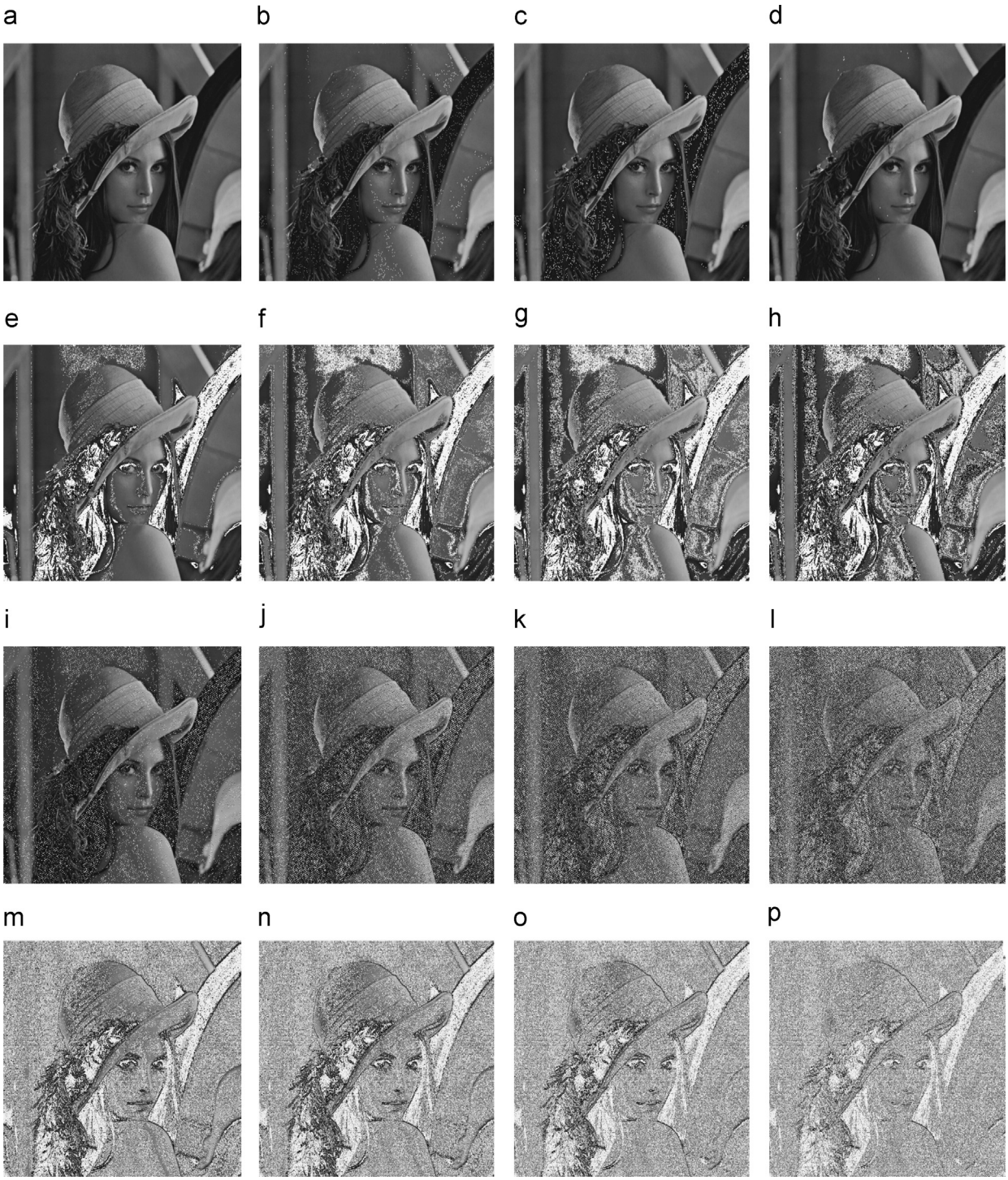

Fig. 7. Output of the test image Lenna processed by DeRand using various combinations of λ and ρ. The value λ for (b), (c) and (d) are 10, 12 and 22, respectively. For the second, third and forth rows, $\lambda = 8, 13$ and 16, respectively. (a) Original. (b) $\rho=1$, SSIM=0.829. (c) $\rho=1$, SSIM=0.850. (d) $\rho=1$, SSIM=0.975. (e) $\rho=10$, SSIM=0.560. (f) $\rho=20$, SSIM=0.292. (g) $\rho=30$, SSIM=0.247. (h) $\rho=40$, SSIM=0.244. (i) $\rho=50$, SSIM=0.404. (j) $\rho=250$, SSIM=0.192. (k) $\rho=500$, SSIM=0.146. (l) $\rho=1100$, SSIM=0.096. (m) $\rho=1500$, SSIM=0.076. (n) $\rho=2000$, SSIM=0.065. (o) $\rho=3000$, SSIM=0.050. (p) $\rho=5000$, SSIM=0.038.

passes only, while [20] still needs to parse the signal for 3 times. Thus, [20] is of the highest complexity, followed by DeRand and [23].

Pertaining to the restoration of host and extraction of payload, the size of side information $|S|$ in DeRand varies, where the minimum and maximum sizes are $|S| = 3\lambda+\rho+N$

and $\lambda \times 2^{\lambda}+\rho+N$ bits, respectively. However, in [20], $|S| = \lambda \times 2^{\lambda}$ bits, which is a constant value. On the other hand, [23] relies on side information where $|S| \in [5\lambda, (2\lambda+|A_{\lambda}|+|B_{\lambda}|) \times \mathcal{B}]$ and $\mathcal{B}$ denotes the number of blocks [23]. Here, as $\mathcal{B}$ increases, $|S|$ increases significantly in [23]. Such dependency of $|S|$ on $\mathcal{B}$ is avoided in DeRand because DeRand operates in global mode, where the input signal is not segmented into blocks for individual processing. All in all, this functional comparison suggests that the proposed DeRand is superior to [20,23].

6. Conclusions

A novel universal histogram mapping method that defines redundancy in any digital signal for data embedding purposes was proposed. Theoretically, Theorem 1 suggests a mean to define redundancy based on the universal parser, and Theorem 2 shows that finding such redundancy is feasible for any digital signal. As such, even for high entropy signal such as random signal, the proposed histogram mapping can still be applied. Practically, these theorems are utilized to design the proposed data embedding method called DeRand, which is universally applicable to any digital signal including random signal. To the best of our knowledge, DeRand is the first universal reversible data embedding method applicable to the random signal. Experimental results show that DeRand achieves scalable embedding capacity (up to 4909 bits when processing random bit sequence of length 256 Kbytes) and scalable distortion with MSE spanning the range of [0.0075, 395.67] with insignificant average file-size increment of 0.42%.

As for future work, we aim to increase the embedding capacity and minimize the side information of DeRand. In addition, we want to verify the robustness of DeRand when operating in noisy environment. Also, we want to improve its carrier capacity by considering various entropy coding schemes.

Acknowledgment

This work was supported by the University of Malaya HIR under Grant UM.C/625/1/HIR/MOHE/FCSIT/01, B000001-22001 (Project title: Unified Scalable Information Hiding).

References

[1] P. Zadrozny, R. Kodali, Big data and splunk, in: Big Data Analytics Using Splunk, Springer, 2013, pp. 1–7.
[2] B.Y. Lei, I.Y. Soon, Z. Li, Blind and robust audio watermarking scheme based on SVD-DCT, Signal Process. 91 (8) (2011) 1973–1984.
[3] A. Cheddad, J. Condell, K. Curran, P.M. Kevitt, Digital image steganography: survey and analysis of current methods, Signal Process. 90 (3) (2010) 727–752.
[4] D. Xu, R. Wang, J. Wang, A novel watermarking scheme for H.264/AVC video authentication, Signal Process.: Image Commun. 26 (6) (2011) 267–279.
[5] Lusson F., Bailey K., Leeney M., Curran K., A novel approach to digital watermarking, exploiting colour spaces, Signal Process. 93 (5) (2013) 1268–1294.
[6] K. Wong, X. Qi, K. Tanaka, A DCT-based Mod4 steganographic method, Signal Process. 87 (6) (2007) 1251–1263.
[7] H. Luo, F.-X. Yu, H. Chen, Z.-L. Huang, H. Li, P.-H. Wang, Reversible data hiding based on block median preservation, Inf. Sci. 181 (2) (2011) 308–328.
[8] C.-C. Chang, T.D. Kieu, A reversible data hiding scheme using complementary embedding strategy, Inf. Sci. 180 (16) (2010) 3045–3058.
[9] W.-J. Yang, K.-L. Chung, H.-Y.M. Liao, Efficient reversible data hiding for color filter array images, Inf. Sci. 190 (2012) 208–226.
[10] C.-C. Chang, C.-C. Lin, C.-S. Tseng, W.-L. Tai, Reversible hiding in dct-based compressed images, Inf. Sci. 177 (13) (2007) 2768–2786.
[11] C.-C. Chang, T.-S. Chen, L.-Z. Chung, A steganographic method based upon JPEG and quantization table modification, Inf. Sci. 141 (1–2) (2002) 123–138.
[12] T.-C. Lin, C.-M. Lin, Wavelet-based copyright-protection scheme for digital images based on local features, Inf. Sci. 179 (19) (2009) 3349–3358.
[13] K. Wong, K. Tanaka, K. Takagi, Y. Nakajima, Complete video quality-preserving data hiding, IEEE Trans. Circuits Syst. Video Technol. 19 (10) (2009) 1499–1512.
[14] H. Malik, R. Ansari, A. Khokhar, Robust data hiding in audio using allpass filters, IEEE Trans. Audio Speech Lang. Process. 15 (4) (2007) 1296–1304.
[15] P. Borges, J. Mayer, E. Izquierdo, Robust and transparent color modulation for text data hiding, IEEE Trans. Multimed. 10 (8) (2008) 1479–1489.
[16] B. Zhao, W. Kou, H. Li, L. Dang, J. Zhang, Effective watermarking scheme in the encrypted domain for buyer-seller watermarking protocol, Inf. Sci. 180 (23) (2010) 4672–4684.
[17] X. Zhang, Separable reversible data hiding in encrypted image, IEEE Trans. Inf. Forensics Secur. 7 (2) (2012) 826–832.
[18] W. Hong, T.-S. Chen, H.-Y. Wu, An improved reversible data hiding in encrypted images using side match, IEEE Signal Process. Lett. 19 (4) (2012) 199–202.
[19] X. Zhang, Reversible data hiding in encrypted image, IEEE Signal Process. Lett. 18 (4) (2011) 255–258.
[20] M.S.A. Karim, K. Wong, Universal data embedding in encrypted domain, Signal Process. 94 (2014) 174–182.
[21] M. Weinberger, G. Seroussi, G. Sapiro, The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS, IEEE Trans. Image Process. 9 (8) (2000) 1309–1324.
[22] S. Golomb, Run-length encodings (corresp.), IEEE Trans. Inf. Theory 12 (3) (1966) 399–401.
[23] S. Ong, K. Wong, K. Tanaka, A scalable reversible data embedding method with progressive quality degradation functionality, Signal Process.: Image Commun. 29 (1) (2014) 135–149.
[24] Z. Ni, Y.-Q. Shi, N. Ansari, W. Su, Reversible data hiding, IEEE Trans. Circuits Syst. Video Technol. 16 (3) (2006) 354–362.
[25] S.-W. Jung, L.T. Ha, S.-J. Ko, A new histogram modification based reversible data hiding algorithm considering the human visual system, IEEE Signal Process. Lett. 18 (2) (2011) 95–98.
[26] G. Marsaglia, W.W. Tsang, The ziggurat method for generating random variables, J. Stat. Softw. 5 (8) (2000) 1–7.