Medical Record Protection with Improved GRDE Data Hiding Method on Audio Files

Muhammad Bagus Andra, Tohari Ahmad, *Member, IAENG*, Tsuyoshi Usagawa

***Abstract*— The rapid development of the network technology and internet have changed the way of how files are distributed, including sensitive data such as medical record. Some methods have been applied to ensure its security, including those with cryptography or steganography, along with their advantages and disadvantages. While cryptography may attract suspiciousness of attackers, steganography may not. This is because the file being sent is similar to ordinary files. Therefore, this similarity is one of the key factor in this data hiding problem, besides the capacity of the secret message which can be fully accommodated. In this paper, we propose a data hiding scheme by using audio files as the cover (carrier) to hide the medical record and to secure it. In order to maximize the embedding capacity of the method while maintaining good quality, we combine the Reduced Difference Expansion with Generalized Difference Expansion methods. In addition, we further enhance the method by reducing the difference to achieve a better quality result; and applying a multiple layer embedding scheme, so that same sample can be used for embedding multiple times. For the proposed scheme, the result is quite satisfactory with a 320.0000 bit of payload data can be embedded into 10 second audio clip while still maintaining the quality by 37dB of PSNR value.**

***Index Terms*—Data hiding, data protection, information and network security, medical record**

I. Introduction

THE development of internet and digital transmission has changed the way of the file being distributed across locations. The convenience of sending and receiving files in a real-time regardless the distance has been proven really useful in many fields including the medical environment. Using the established information system, hospitals can exchange information and data almost instantly including a medical record (e.g. [1]) of a patient which leads to better integration between the hospitals, faster medical diagnosis and minimized information error. However, a medical record that contains personal data is confidential which gives a rise to a security issue. To ensure that the data is safe from a hacker attack, a data hiding scheme is used to protect the data being transmitted. This can be done either by cryptography or steganography. Different from cryptography (e.g. [2]), steganography such as [3] [4] produces relatively similar data where public may not be aware of the existence of the secret in it.

Steganography has been used widely for securing data. It undetectably alters a cover object to embed a secret data called payload [5]. In the case of medical data where the payload and the cover data is sensitive, both data must be retrieved without losing any information. This can be achieved using reversible embedding method such as histogram modification of pixel difference that was introduced by Kieu and Chang [6], using integer transform described by Wang et al. [7], and using difference expansion that was invented by Tian [8].

Alattar further improves Tian's method by using triple and quad pair of pixels [9] [10]. He also generalizes the form of the expansion [11] improving the capacity of embeddable data in the pixels. Lou et al. [12] introduced the reduced difference expansion by reducing the difference between the pixels before embedding, resulting in a better quality of images after the embedding process. Further development of difference expansion and its variation for protecting medical data are provided in [13] [14] [15].

Audio files are often chosen as a cover file instead of image file due to generally bigger file size than an image file, providing more space for data to be embedded. Generally, in audio steganography, data embedding is classified into spatial and transform domains [16]. However, most of these methods introduce an irreversible degradation that cannot be tolerated when working with sensitive data such as medical record. Several reversible embedding scheme for audio have been introduced. Wang et al. [17] proposed a reversible embedding method using histogram shift and predicted error expansion, and Choi et al [18] have created a remarkable scheme for applying generalized difference expansion method on the audio file using the intelligent partition.

Using the scheme proposed by Choi et al. [18], we can bring the advantage of difference expansion that has high embedding capacity and low degradation on the result to the audio domain, however, these methods can be further improved in terms of embedding capacity and encoding quality. In this paper we focused on the improvement of the method proposed by Choi et. al. [18] by introducing improved reduced generalized difference expansion which is an improved method from [11] and [12] to be used in embedding process instead of generalized difference expansion. Furthermore, we will use a multiple layer embedding scheme to maximize the embedding capacity while maintaining an acceptable PSNR value.

Manuscript received May 3, 2016; revised October 25, 2016. Part of this works was supported by Hitachi Global Foundation.

Muhammad Bagus Andra was with Department of Informatics, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia. He is now with Kumamoto University, Japan; (email: bagus@hicc.kumamoto-u.ac.jp).

Tohari Ahmad is with Department of Informatics, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia; (corresponding author, phone: +62315939214; email: tohari@if.its.ac.id).

Tsuyoshi Usagawa is with HICC Laboratory, Kumamoto University, Kumamoto, Japan; (email: tuie@cs.kumamoto-u.ac.jp)

This paper is organized as follows. Related work is discussed in Section 2, including a brief review of generalized difference expansion, reduced difference expansion, and intelligent partition. The proposed scheme of embedding data with improved general reduced difference expansion (IGRDE) will be presented in Section 3 along with its application within multiple layer embedding scheme. The experimental result and its comparison with [18] is presented and discussed in Section 4. Conclusion for this paper is presented in Section 5.

II. Related Works

This section reviews some methods, including generalized difference expansion, reduced difference expansion, and intelligent partition. We also discuss other related works.

A. Generalized Difference Expansion

Generalized difference expansion is intended to be an improvement over difference expansion [8] that uses a pair of pixels for embedding a single bit. Instead of using a pair of pixels, this method forms a vector of pixel values by dividing pixels of an image to several blocks. Vector $u = (u_0, u_1, u_2, \dots, u_{N-1})$ is constructed by choosing non-overlapping *N* pixels at certain order [11]. If an image has a dimension of *m* x *n*, the block may be constructed as shown in Fig. 1 where *a* and *b* is one block of pixels and ***u*** is a vector of pixels of the respective block.

The embedding process can be done by calculating the average of pixels and the difference of each pixel in the vector with the first pixel as shown on (1) and (2) respectively, where v_0 is the average of pixels in the vector, u_i is the value of each pixel in the vector and *N* is the size of the block. The difference of every pixel v_i is obtained by subtracting the first pixel on the block u_0 from each pixel in the block u_i.

$$v_0 = \left\lfloor \frac{\sum_{i=0}^{N} u_i}{\sum_{i=0}^{N} i} \right\rfloor \tag{1}$$

$$v_i = u_i - u_0 | 1 \le i \le N-1 \tag{2}$$

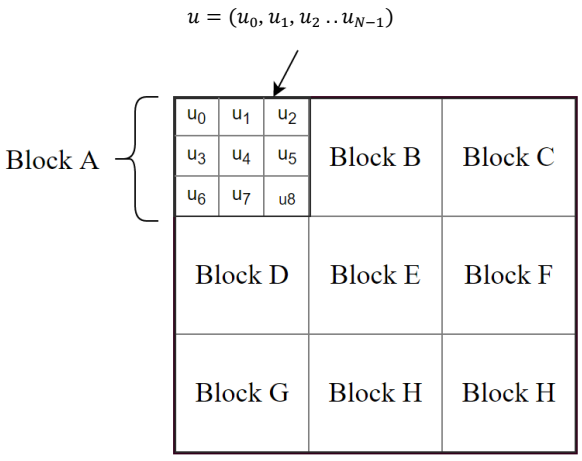

Fig. 1 Constructing Pixel Blocks in GDE (adapted from [8]), non-overlapping pixels are selected to form a vector of pixel values.

Every bit from payload data is then embedded to the difference by either (3) or (4). If the difference value satisfies both (5) and (6) then the block is expandable and (3) is used to embed the bit, otherwise, the block is changeable and (4) is used.

$$\tilde{v}_i = 2 * v_i + b_i | 0 < i < N \tag{3}$$

$$\tilde{v}_i = 2 * \left\lfloor \frac{v_i}{2} \right\rfloor + b_i | 0 < i < N \tag{4}$$

$$0 \le v_0 - \left\lfloor \frac{\sum_{i=0}^{N} \tilde{v}_i}{\sum_{i=0}^{N} i} \right\rfloor \le 255 \tag{5}$$

$$0 \le v_i + v_0 - \left\lfloor \frac{\sum_{i=0}^{N} \tilde{v}_i}{\sum_{i=0}^{N} i} \right\rfloor \le 255 | 0 < i < N \tag{6}$$

Here, $\tilde{v}_i$ is the difference value after embedding, v_i is the original difference and b_i is the payload bit to be embedded, where $b \in (0,1)$. The new pixel after embedding then can be obtained by using (7).

$$u_0' = v_0 + \left\lfloor \frac{\sum_{i=0}^{N} \tilde{v}_i}{\sum_{i=0}^{N} i} \right\rfloor \tag{7}$$
$$u_i' = \tilde{v}_i + u_0$$

To restore the value from the embedded one, we must identify the block whether it is expandable or changeable from the location map. For the expandable block, the original difference is retrieved with (8) otherwise, retrieve it according to (4).

$$v_i = \frac{\tilde{v}_i}{2} | 1 \le i < N \tag{8}$$

The embedded payload is restored by taking it sequentially from the LSB of the pixels value. Besides Alattar [11], Wang [19] have also developed a generalized integer transform that used payload-dependent location map that divides the pixel block into three categories: expandable, changeable and others.

B. Reduced Difference Expansion

In order to minimize the degradation caused by large difference value in the embedding process of difference expansion, Lou [12] proposes a new scheme that can improve the quality of the embedded audio by reducing the difference between the pixels before the payload is embedded. The reducing process is done as shown in (9) where h' is a new difference, and h is the original difference.

$$h' = \begin{cases} h & \text{if } h' < 2 \\ h - 2^{\lfloor \log_2 h \rfloor - 1} & \text{if otherwise} \end{cases} \tag{9}$$

The payload bit then can be embedded with the same process as in the difference expansion. Every reduced pixel pair is recorded in a location map. In order to retrieve the original difference of a pixel's pair, first, the location map is read and the original difference can be obtained according to the value in the location map as shown in (10).

$$h = \begin{cases} h' + 2^{\lfloor log_2 h' \rfloor - 1} & \text{if } location\ map = 0 \\ h' + 2^{\lfloor log_2 h' \rfloor} & \text{if } location\ map = 1 \end{cases} \quad (10)$$

After the original difference is retrieved, the embedded bit and the original pixel value can be obtained by using the same process as difference expansion. Kim et al [20] have also proposed a novel expansion technique that can minimize the location map by dividing the block into 4 categories: unambiguously un-expandable set, ambiguously un-expandable set, ambiguously expandable set and, unambiguously expandable set.

C. Intelligent Partition

One main reason why steganography method for an image file such as difference expansion cannot be used in the audio file is that those two file types have different sample size. Because difference expansion was originally designed for 8-bit pixel with square block that is represented with a 2D array, difference expansion cannot be directly applied to audio samples that generally have 16-bit length sample and represented as an 1D array. In order to make the difference expansion compatible with the audio samples, Choi et al [18] have devised a scheme to partition the 16-bit audio samples into two portions that each has a length of 8 bit.

To do this partition, each audio sample is represented in their binary form, forming a 2D array of binaries. Each row represents one sample and each column is called a bigit. The grouping of the bigits is determined by the variance of each bigit.

D. Audio Steganography

The attempt on using an audio file as the cover for steganography has been done by using several techniques. One of them which resides in the spatial domain is low bit encoding or also known as last bit encoding [16] exploits the last bit of samples in an audio file to embed an information by replacing it with payload bits. Many efforts have been done to improve the capability of this method in terms of embedding capacity and quality. Cvejic and Seppanen [21] develop minimum-error replacement (MER) to improve the quality and the capacity of embedded file by distributing the bit used for embedding. Asad et. al. [22] enhance the conventional LSB embedding by devising a random sample selection in order to confuse the attacker, increasing the security of LSB. Kumar and Anuradha [23] use the Fibonacci number to select the sample used for embedding, improving the quality of the embedded file. Using variable bit selection, Banerjee et. al [24] improves the transparency of LSB method by dispersing the used bit for embedding.

The echo hiding technique that also resides in the spatial domain uses short echo on the audio signal to embed the payload bit while still retaining the same statistical characteristic. This creates a masking effect that can trick human auditory system so that no noticeable different is perceived [16]. To improve this embedding technique, Chao and Zhang [25] propose the use of echo distribution by selecting multiple echo kernel rather than one big echo, achieving more natural sound and reducing the detection rate of the embedded payload. With the same objective of raising the imperceptibility of the embedded audio signal, Kim and Choi [26] introduced a novel echo hiding technique by using the combination of forward kernel and backward kernel to reduce the amplitude of the echo. Chen and Wu [27] developed a robust and highly secure echo hiding technique by using analysis-by-synthesis approach in the embedding step; adapting the amplitude of the echoes according to the audio signal and interlaced kernel; providing a more robust result against many attacks. Delforouzi and Pooyan [28] propose a method to increase the capacity of echo hiding by using different echo delay values so that in each section, several bits can be embedded instead of one bit. Furthermore, the dual backward-forward echo kernel is introduced to increase the robustness of the result.

In contrast, the transformed domain hides the payload data in the frequency domain of the audio signal, one of them is by using discrete wave transform (DWT) [16]. It decomposes the audio signal into several sub-bands, by determining the sub-band signal energy. Here, sub-band with less energy can be used for embedding because the change done on this signal does not cause a drastic change in the whole audio signal. Another technique that falls into this domain is spread spectrum. It works by hiding information throughout frequency spectrum of the audio signal. This technique produces redundant copies of the data signal, so that if there is a signal loss in the transmission, the hidden information can still be recovered [9]. Ahani et. al. [29] propose a novel DWT based method by introducing a sparse representation of wavelet coefficient, improving the quality and the embedding rate of the result.

III. Proposed Scheme

The scheme proposed in [18] uses general difference expansion as the embedding method. While this is sufficient in terms of embedding capacity, in some case there are some "click" noises introduced in the embedded audio. Reduced difference expansion can be used to minimize the noise by reducing the difference, but the embedding capacity of RDE is limited at most 0.5 bit per pixel or sample. In order to take advantage of both methods, in our proposed scheme we combine both method by using RDE on the generalized difference expansion. Furthermore, we improve the RDE method by expanding the logarithmic equation of the difference transformation to reduce the difference further.

The whole operation of our proposed scheme is illustrated in Fig. 2 which consists of 4 processes. Those are:

- Audio sample processing
- Sample array partition

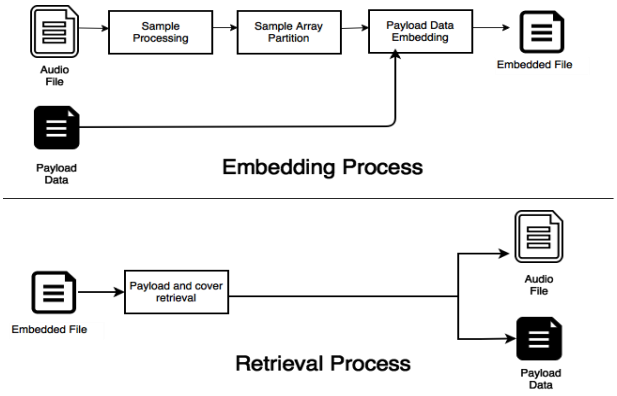

Fig. 2 The Overall Process of the proposed scheme. It has of 4 general steps which comprises 3 steps for encoding and 1 step for decoding processes.

- Payload data embedding
- Payload data and cover retrieval

A. *Audio Sample Processing*

Samples in audio file are generally represented as signed 16-bit integer with PCM (Pulse-code Modulation) whose value ranges from -32767 to 32767. Because difference expansion method is designed for 8-bit signed integer image pixel, it cannot handle a negative number. To make it compatible with audio samples we must first normalize the sample value so it does not contain any negative number. The normalization is done with (11), where u_i' is the sample value after the normalization, u_i is the original sample value and u_{max} is the maximum value for the sample. If the sample length is 16-bit then u_{max} =32767.

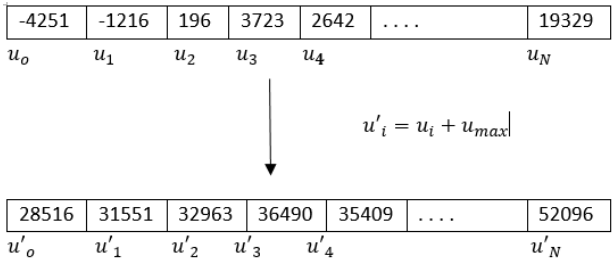

Fig. 3 Normalization of samples value, for the Difference Expansion method to work. It is required to normalize the sample value to positive integer.

$$u'_i = u_i + u_{\max} \tag{11}$$

The normalization process on array of audio sample is shown in Fig. 3.

B. *Sample Array Partition*

Before the payload data is embedded, in order to make the difference expansion method applicable to the audio samples, the 16-bit sample array is partitioned into two array groups, each consisting of 8-bit long samples. The partitioning process is done with the intelligent partitioning process according to [18]. The overall partitioning process is illustrated in Fig. 4 and the detailed operation of each step is explained in each subsection as follows.

Representing Samples as Array of Bigit

To split the audio samples into two groups we firstly represent each sample in the audio files as an array of bigits. Each sample in the audio files is represented in its binary array, each of which then forms a two-dimensional array. Here, each row is the binary representation of the sample and each column is the bigits. The two dimension array result will have 16 columns and N rows where N is the number of samples in the audio file. Fig. 5 depicts the transformation process of the audio samples into bigits array.

Variance Calculation

After the bigits are formed from the audio samples, we

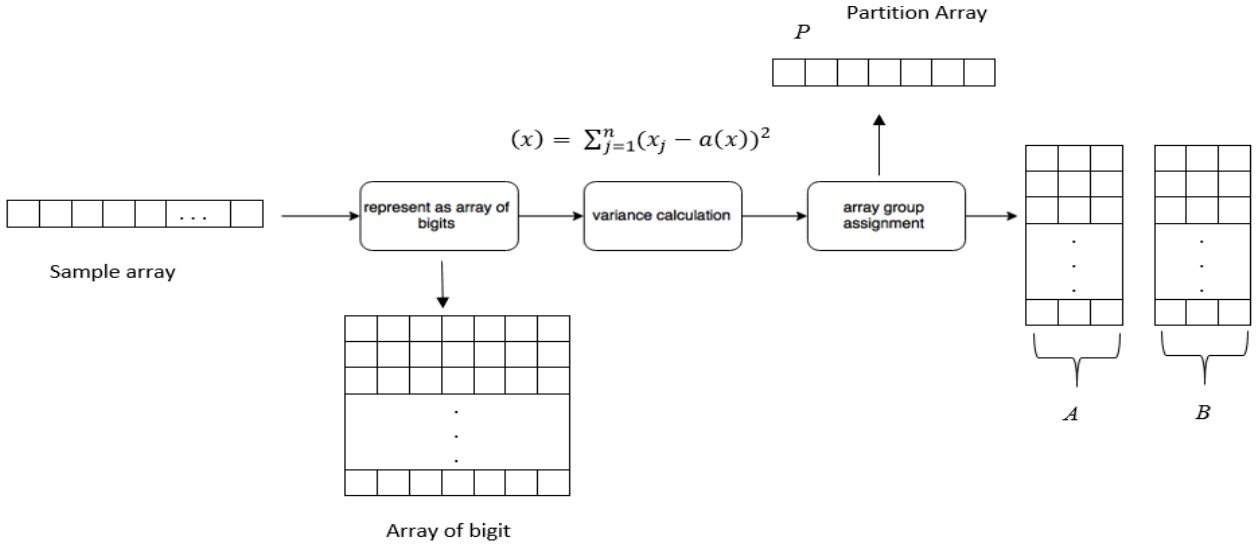

Fig. 4 Sample array partition, the 16-bit sample array is partitioned into two array groups.

.

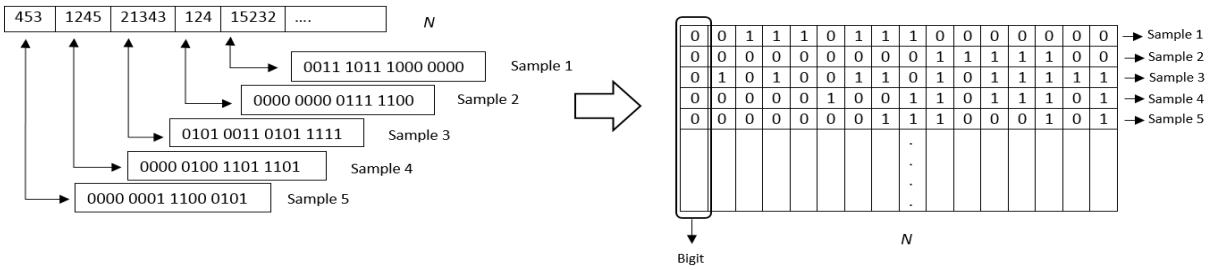

Fig. 5 Representing sample array as bigits,. First, each sample is represented in its binary form. It is then represented as 2D array, that each column of the array is a bigit.

calculate the variance of each bigit. The variance is used to determine the grouping for each bigit in the next step. To calculate the variance of each bigit, we divide the bigit into equal sized segments with length of *n*. For each segment we specify the segment variance v_s as the bitwise variance of the segment defined by (12).

$$v_s = \sum_{i=0}^{n}(s_i - \overline{x}(s))^2 \quad (12)$$

Here, *n* is the length of segments, *s* is a segment of a bigit consisted of several binary values $(s_1, s_2, s_3 \ldots, s_n)$ and $\overline{x}(s)$ is the mean of the segment. The variance of the bigit *v* is then found by summing all the segment variance as shown by (13).

$$v = \sum_{i=0}^{m} v_{si} \quad (13)$$

In this case, m is the number of the segments in the bigit and v_{si} is the variance value of each segment. The whole variance calculation process is illustrated in Fig 6.

Array Group Assignment

Next, we will assign each bigit to two groups A and B by considering the variance value of each bigit. The grouping is done by firstly creating an array *S* from the values of the bigit variance that was calculated in the previous step. The array *S* is then sorted in descending order so that bigit with the biggest variance is in the first index. Two groups A and B are then created to record the index of the original bigit array. The index of the original array is then assigned as follow: First, assign the 6 most significant bigits that have the largest variance into group A and B alternately, that is, the most significant bigit goes to group A, the second goes to group B and so on.

Next, excluding the index that already assigned, distribute the first and the third element from array *S* to group *B*, and assign the second and the fourth element to group *A*. For the next four index, two are put in *B* and the other two to *A*; and finally, the last two is distributed to *A* and *B*, sequentially. The original bigit array *M* is then partitioned into *M1* and *M2* where each group is an 8-bit length bigit array by assigning each bigit which index are in the group *A* to *M1* and group *B* to *M2* after both groups A and B are sorted first. Fig. 7 depicts the ordering and assigning process of each bigit.

To restore the original position of each bigit in the retrieval process, the location of bigits is stored in a partition array *P*. In the partition array P, the bigit that is placed in the same group are marked with the same 1-bit marker value, whether it is 0 or 1. Each index of array *P* represents the index number of the bigit in the original sample array. With the information of these parameters, bigit group can be recreated and the data retrieval process can be carried out without any data change or loss.

C. Payload Data Embedding

Following the nature of generalized difference expansion, the embedding process of the payload data is done segment by segment which size is the same as that used in the

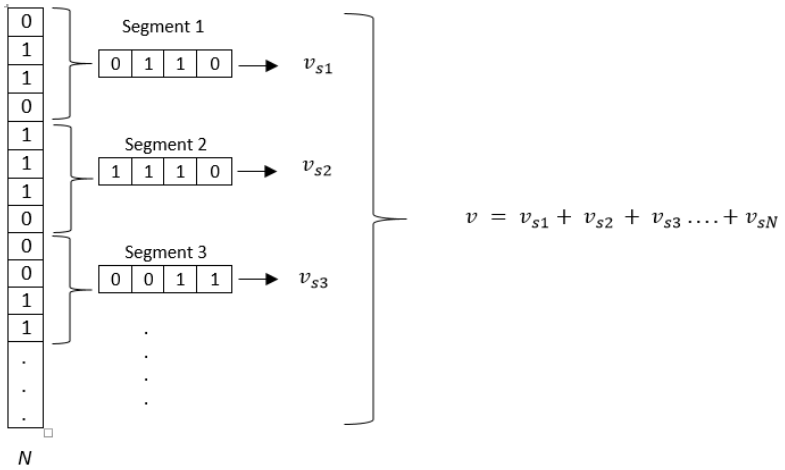

Fig. 6 Bigit variance calculation. It is done in each segment in the bigit.

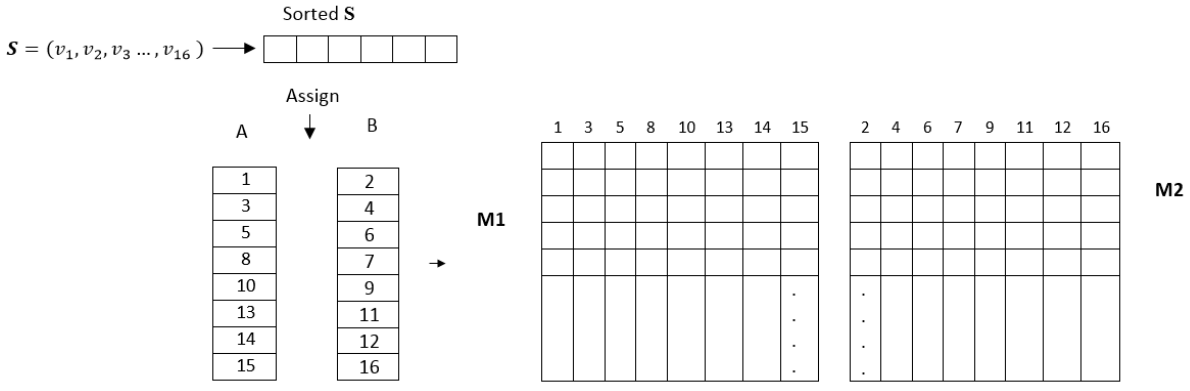

Fig. 7 Assignment of bigit, the order of assignment is done with specific order.

partition process. Moreover, in our proposed scheme, the embedding process is performed in multiple layers to accommodate large payload. This is possible because the difference reduction process of the RDE makes the result sample value does not differ much from the original value. So, the embedding can be done in the same segment multiple times. In each layer, the embedding process is carried out twice, once in partition M1 and once in partition M2. The payload data which is a medical record of the patient are represented as a long string containing all the whitespaces and symbol. This string is then converted to its binary representation resulting in a long one-dimensional binary array. Next, the array is divided according to the capacity of each partition. The whole embedding process is illustrated in Fig. 8.

Capacity Checking and Partition Dividing

Before the payload data is embedded, the capacity of each layer is calculated. The total capacity of each layer is the sum of the capacity of partition M1 and partition M2. The capacity of each partition is determined by the expandable and the changeable segments in the partition. Thus, it can be found with (14), where C_p is the capacity of the partition, S_e is the number of expandable segments in the partition, S_c is the number of changeable segments in the partition and m is the number of samples in one segment. The expandable and changeable segments can be identified with (5) and (6).

$$C_p = \left(\sum_{i=0}^{n} S_e + \sum_{i=0}^{n} S_c\right) m \tag{14}$$

The threshold value t is introduced as a control variable for the sample difference such that a segment that have a sample difference exceeding the threshold will not be used for embedding. The final capacity of each partition is then obtained with (15) where C_f is the final capacity, S_t is the segment whose sample difference exceeds the threshold, n is the number of segment in the partition and m is the number of samples in one segment.

$$C_f = C_p - \left(\sum_{i=0}^{n} S_t\right) m \tag{15}$$

The payload which will be embedded are divided according to the capacity. If the size of the payload is less than or equal to the capacity of M1 then, all the payload can be embedded only to M1 while M2 leaved as is. If the payload sizes are greater than the capacity of M1, then we embed the payload data up to the capacity of M1 and embed the remaining into the M2. If the embedding process is already done in both partition and there are still payload data that have not been embedded, we embed the remaining payload in the next layer with the same procedure. Fig. 9 depicts the payload dividing process.

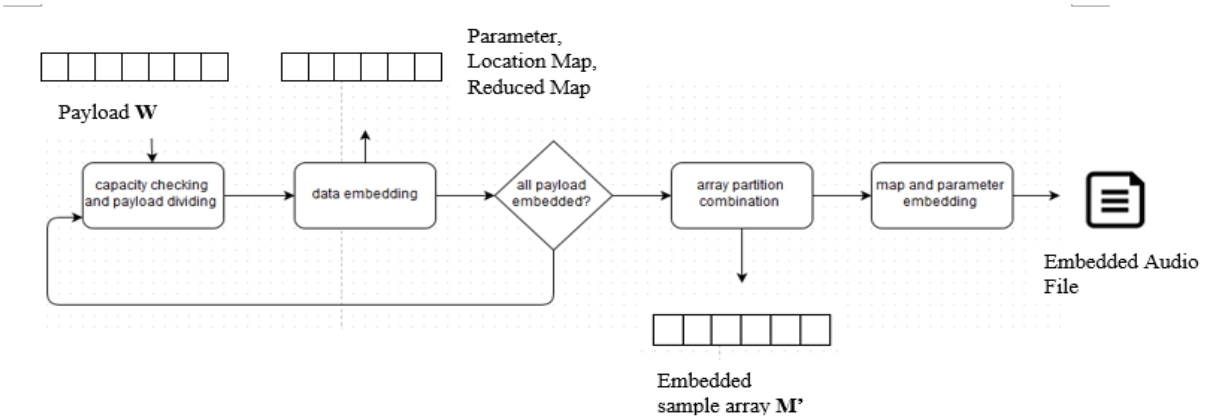

Fig. 8 Payload data embedding process. The payload is firstly divided into two parts and each part is embedded into one group of sample arrays. The embedded sample is then combined to form the new sample array.

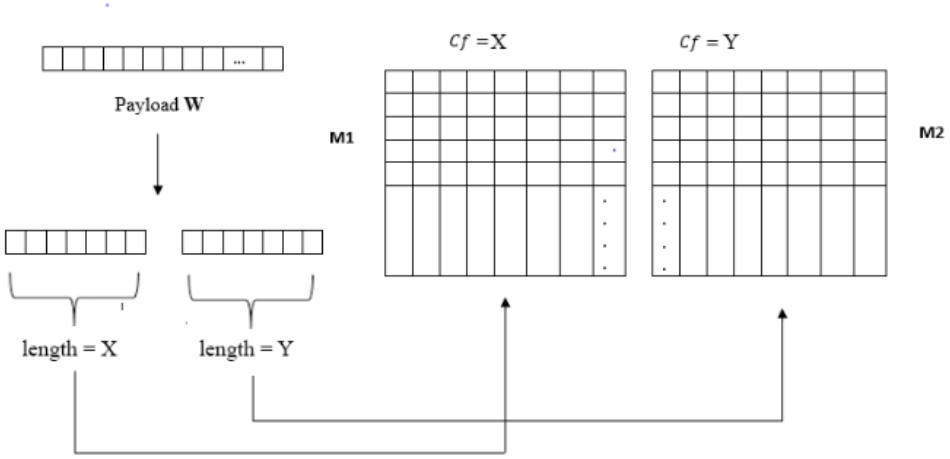

Fig. 9 Payload dividing where the payload data is represented as a string of binary and divided into two parts.

TABLE I
REDUCED MAP VALUES

Condition	Reduced Map Value
$2^{\lfloor log_2 v \rfloor}/2^{\lfloor log_2 \overline{v} \rfloor} = 4$	01
$2^{\lfloor log_2 v \rfloor}/2^{\lfloor log_2 \overline{v} \rfloor} = 1$	10
Not Reduced	11
Other	00

TABLE II
LOCATION MAP

Segment Type	Location Map Value
Expandable	0
Changeable	1

Data Embedding

Every expandable and changeable segment in each partition will be used for embedding, excluding the segments whose sample value exceeds the threshold. In each *n*-length segment, an *n*-1 bit of data are embedded. The first sample of the segment is not used for embedding because it will be operated for a retrieving process. In order to embed the payload bits on the segment, the difference between each sample in the segment is found by the same procedure as generalized DE described in (2).This difference is then reduced so that the resulting sample will be closer to the original sample, reducing the distortion in the audio result. In our proposed method, we improve the RDE [8] by expanding the logarithmic equation to obtain a smaller difference between the samples, as depicted by (16), where $\overline{v}$ is the reduced difference and v is the original difference.

$$\overline{v} = \begin{cases} v, & \text{if } v < 4 \\ v - 2^{\lfloor log_2 v \rfloor - 1} - 2^{\lfloor log_2 v \rfloor - 2}, & \text{if } v \geq 4 \end{cases} \quad (16)$$

Unlike the original RDE, if the sample difference is less than 4, then the difference will not be reduced. This is because the equation will result in a float number when $v = 3$ which causes underflow when $v \leq 3$ and this is not acceptable for the difference expansion. This equation introduces a 4 type of values in the reduced map depending on the result represented by a 2 bit value for each segment. Those values and its condition are shown in the reduced map in Table I.

Reduced Map *R* records the reduction operation that is performed on each segment. This map is later embedded to the sample array for the retrieving process. The payload bit is then embedded by an operation similar to (4), but instead of using the average of the segment as the v_0, we simply use the first sample in the segment as v'_0. This has made the value of the first sample unchanged. This also makes the retrieval easier because the original sample value can be obtained by simply using the first sample. The new embedding operation is defined by (17), where v'_i is the embedded difference, u'_i is the embedded sample, u_0 is the first sample in the segment, $\overline{v_i}$ is the reduced difference and *b* is the payload bit.

$$\left.\begin{array}{c} u'_0 = u_0 \\ v'_i = 2 * \overline{v_i} + b \\ u'_i = u_0 + v'_i \end{array}\right\} \quad (17)$$

The embedding operation is performed on each segment in each partition until all the payload data are embedded. Here, the capacity and the size of the payload embedded in each layer are recorded and will be used later in the retrieval process. For each segment used for embedding, a location map *L* records the type of segment used whether it is an expandable or changeable segments. The segment type is recorded so that the system can determine the operation used in the retrieving process. Table II displays the location map values according to the segment's type.

Partition Combination

Both partition *M1* and *M2* that already embedded with payload data are combined again into one sample array. To combine both partitions, the same partition array *P* from the array partition step are used. For each index in the partition array, take the bigit from the partition with the same value e.g. value 0 for partition *M1* and value 1 for partition *M2*. The bigit is then added into the new sample array *M'*.

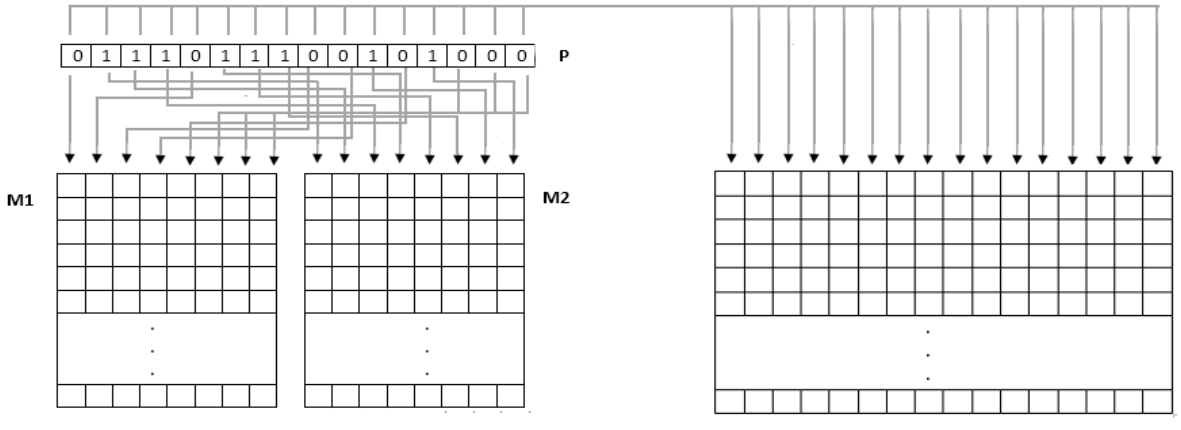

Fig. 10 Sample array reconstruction, using the partition array *P* the sample array is reconstructed to form *M'*

After all the 16 bigits are added to the *M'*. The new sample array is formed with the exact same size as the original sample array having 16-bit length for every sample. Fig. 10 depicts the reconstruction of the sample array.

Parameter and Maps Embedding

The embedding and the partition processes of the original sample array produce a location map and a reduced map, to successfully retrieve the original sample data and payload data. This maps and other parameter information such as segment size, partition array, and layer number are needed. There are several ways to obtain the information of the parameters at the time of retrieval, one of them is to gather all the needed information and parameter, put it into a single file and send it separately from the embedded file through the secure channel. In order to ensure the protection of the file, additional encryption scheme may be performed on the parameter file. However, this method is often considered not optimal because the files are sent separately and the retrieval process becomes dependant on the parameter file.

The other way is to embed the whole information on the embedded file itself, this way the retrieval process can be done immediately and only a single file is needed at the time. Because the maps are represented in bits, we can transform the maps array into an array of integers by taking every 16 bits of the value as an integer. This array of integers is then padded into the last portion of the sample array. In the case of multiple layers used for embedding, each layer will have its own reduced map and location map. To avoid confusion at the retrieving process, the map for each layer will be separated with some unique values or symbols. The process for partition array *P* is same, that we represent the 16-bit partition as a single integer and append it to the end of the sample array. As for the other parameters such as segment size and layer number, which are already in integer values, they can be directly padded in arbitrary order as long as the extraction in the retrieving process is same as the embedding order.

D. Payload and Cover Retrieval

In order to successfully retrieve the original sample array of the cover and the embedded payload, information used in the embedded process is needed: location map, reduced map, partition map and other parameters such as layer number and segment size. After all the information is available, the embedded sample array is partitioned using the partition array, creating a group with the same bigit member as in the embedding process. Then, the payload data bits are retrieved from each embedded sample, and finally, the sample array is reconstructed and the payload bits are combined, producing the original cover and the payload data.

Maps and Parameter Extraction

In order to retrieve the payload and cover data, we need to get the parameters used in the embedding process. First, the parameters such as layer number and segment size are extracted by taking the last two samples from the array. Next, the partition array *P* are obtained by taking the third last sample from the array, and the integer value is then represented as an array of bit. The location map is extracted by taking *n* samples until a unique symbol is found indicating the end of the map. For the multiple layer cases, several location maps are extracted according to the layer number. The extraction procedure of the reduced map is same as the location map. Additionally, the unique symbols for reduced map and location map are set differently to avoid confusion. After all of the required information is obtained, the embedded sample arrays are partitioned with the procedure explained in sample array partition process without the need to calculate the variances, instead, the existing partition array is employed, forming partition *M1'* and *M2'* with same bigit members as in the embedding process. This process is illustrated in Fig 11.

Payload Data Bit Retrieval

Payload data are extracted from each segment of the partitions sample. The partitions *M1'* and *M2'* that are formed from the last step are divided into segments, each of which has *n* numbers of samples according to the parameters obtained from the last step. In order to identify the segment, we check the location map for each segment in each partition. In this case, if there is a value in the location map for the said segment then there are payload bits embedded in that segment. To recover the bits, inverse difference expansion is used, but the difference of the segment must be firstly recovered. The reduced map for the segment is checked and the difference is recovered with an equation shown in (18) according to the value of the reduced map, where v is the embedded difference and $\check{v}$ is the reduced difference.

$$v = \begin{cases} \check{v} + 2^{\lfloor log_2 v \rfloor - 1} + 2^{\lfloor log_2 v \rfloor}, & \text{if } value = 00 \\ \check{v} + 2^{\lfloor log_2 v \rfloor} + 2^{\lfloor log_2 v \rfloor + 1}, & \text{if } value = 01 \\ \check{v} - 2^{\lfloor log_2 v \rfloor - 1} - 2^{\lfloor log_2 v \rfloor - 2}, & \text{if } value = 10 \\ \check{v} = v, & \text{if } value = 11 \end{cases} \quad (18)$$

For the reduced map, 2 bits are taken from each segment at a time because the segment identifier is 2-bit length as explained in the sample array partition process. After the original difference is obtained, the location map value is checked and the original sample and payload bits for that segment is obtained with (19), where v' is the original difference, v is the embedded difference, b is the embedded bit obtained by taking the LSB of the sample, u is the original sample value and u'_0 is the first sample value on the segment.

$$v' = \begin{cases} \frac{v-b}{2}, & \text{if } value = 0 \\ v - b, & \text{if } value = 1 \end{cases} \quad (19)$$
$$u = u'_0 + v'$$

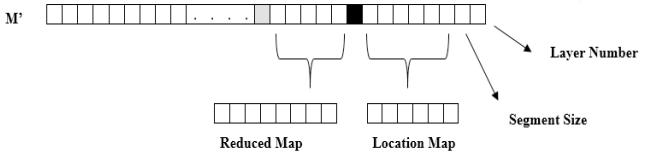

Fig. 11 Maps and parameter extraction, that the parameters and location map is extracted from the array

TABLE III
METHOD COMPARISON

Condition	Choi et al [18]	Proposed Method
Data embedding	Implementing Generalized Difference Expansion [11] which works on images	Improving the capability of method proposed in [12] and combining it with [18]
Layering	Single layer	Multi Layers
Type of embedding map	Location map	Location map and reduced map
Map embedding	Appended to 1 partition before the files being combined	Padded at the end of the audio file

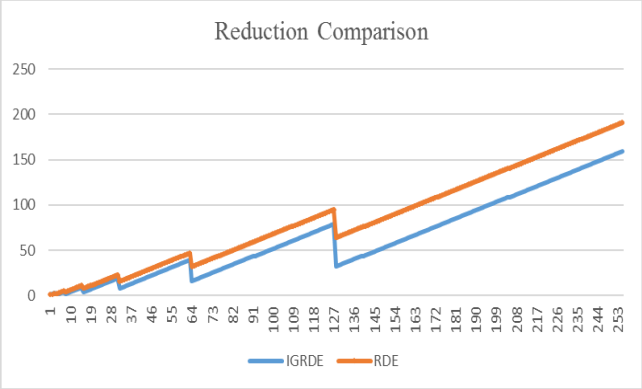

Fig. 12 Difference Reduction Comparison, that the proposed method shows a reduction in the difference between the samples.

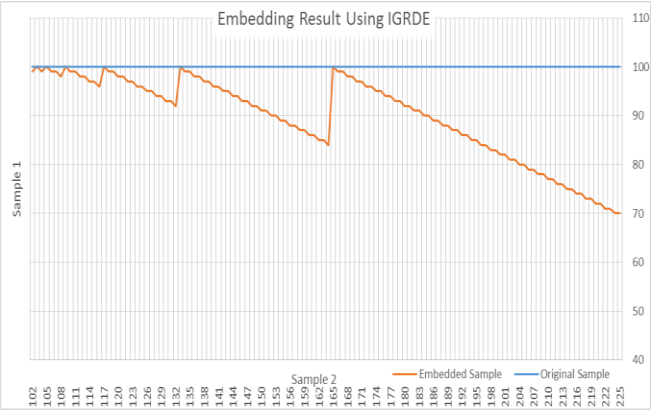

Fig. 13 Difference Reduction Comparison, the proposed method shows a reduction in the difference between the samples.

In the case multiple layers are used for embedding, the retrieval is done from the deepest layer. The sample result of the current layer is used to retrieve the sample value in the next layer, that is the layer above the current layer. In addition, the payload bits are also stored in the reverse order, where the bits that are retrieved from the deepest layer are the latest bits of the payload data.

Cover and Payload Data Reconstruction

Using the retrieved original sample on each partition *M1'* and *M2'*, the original sample array *M* is formed using partition combination process. The sample array *M* will contain exactly same samples as the original audio file. The cover file is then recreated by writing an audio file with *M* which sampling rate is same as the original file. The payload is reconstructed by taking the retrieved bit from the last step and forming one-dimensional bit array. This bit array is then represented as a string by taking 8 bit at a time to form a single character for character encoding with 8-bit length like UTF-8. As for the multiple layer case, the *M* is the sample array that retrieved last which is in the top layer. The partition array used for combining the partition in each layer is identical, and the payload bit are constructed by appending the bits retrieved in the deepest layer at the end of the payload array. This process is continued by the layer above it and so on until all the payload bits are retrieved. The retrieved payload is then written to a text file, resulting a medical record of the patient without any information loss.

The overall comparison between the proposed method and the research of Choi et al. [18] can be summarized in Table III, whose experimental results are provided in Section IV. Furthermore, there are more evaluation scenarios in this paper.

IV. EXPERIMENT AND ANALYSIS

We first evaluate our proposed scheme by comparing the effectiveness of the reduction process with the original RDE method. In our proposed scheme, the difference between the first sample in the segment is reduced by further expanding the logarithmic operation. The evaluation is done within the 8-bit sample difference value domain ranging from 0 to 255. Fig. 12 shows the comparison between reduced difference of the original RDE and our proposed method.

From the evaluation, we can see that the IGRDE method can give a smaller difference between the sample values. On average, the improved reduced difference expansion gives about 26.96% smaller difference compared to the original RDE. In addition, IGRDE also gives a smaller difference for every value in the range except for 3 and 2 because, in the IGRDE, the difference whose value is less than 4 is not reduced. IGRDE still preserves RDE logarithmic nature where the reduced difference increases incrementally until reached a certain point and decreases significantly, that is when the value reach 2^n. We then evaluate the effect of the reduced difference on the embedded sample. In this case, we try to embed a bit 1 into two samples where one of the sample is static and the other one is diynamically increased. The evaluation results of this experiment are shown in Fig. 13.

From the experimental result, we find that the embedding result is corresponding to the reduction evaluation, with increasing difference between the two original samples, the differences from the resulting sample will also increase. Although the difference is growing, we find it does not differ too much from the original sample values. The overall degradation of the sample can still maintain the quality of the audio file. From our evaluation, the highest difference occurs in the first sample by 30% and 11.76% in sample 2 where the smallest difference occurs in sample 1 by 0% and sample 2 by 0%. This means that the sample value does not change at all. As shown in Fig. 13, this condition happens periodically, specifically when the difference is 2^n.

To test the whole process of our proposed scheme, we perform three other evaluation. In the first evaluation, we embed the payload data in the audio file without using a

threshold, which means that the threshold value is set to 255 and the whole audio sample array is treated as one segment. This evaluation is used to measure the quality of the embedded audio result in a case where all the samples are used for embedding.

In the second evaluation, we embed the medical record of a patient as payload data to the audio files. The medical record is a text file with patient information including personal data, medical history, perception that given to the patient, and other medical data on it. The payload is embedded with various threshold values and segment size parameters to see the effect of the parameter on the result. From this evaluation, we can measure some aspects of the proposed scheme such as embedding capacity, the reversibility of both cover and payload data, and the quality of the embedded audio.

In the third evaluation, we attempted to embed the payload data into shorter audio clip in every genre. We tried to analyze the correlation between the embedding capacity and the length of audio clip used for each genre of musics.

The audio data used as cover in evaluation 1 and 2 are 10 second audio clip files that have 44.1 kHz sampling rate with 16-bit length sample. For evaluation 3 we used the same audio clip but with a shorter duration. The reason why we use 44.1 kHz is because it is the most common sampling rate used in the audio files nowadays. To ease the embedding process, the stereo track is mixed into a mono track, so that the resulting sample array is only one, instead of two in the stereo where the samples for the left and right channel are different.

A. *Evaluation I*

In this evaluation we compare the embedding capacity of the proposed method and the original method used by Choi et al [18]. The data used for payload is identical text file and the embedding is done in a single layer. The audio file used is 10-second audio clip.

In the case of no threshold is used in the embedding process, the payload data that can be embedded is the n-1 bit where n is the number of samples in the audio file. With 10-second audio file which sampling rate is 44.1 kHz. The total amount of sample is 441.000. Here, our scheme is able to embed 482.346 bit in the audio file except the first sample which is unchanged. The PSNR of the result is 26.97 dB. With the same evaluation scheme, Choi et al [18] are able to hide 441.000 bit with PSNR value of around 15 dB. We find drastic improvement of the quality because of the reduced difference used in our proposed method. We can also find that more than one bit can be embedded to the sample. This is because the 16-bit sample is partitioned into two groups where each group is 8-bit length. This means that the maximum embedding capacity that can be reached is 2 bit per sample.

B. *Evaluation II*

We choose four audio files as a cover which represents different genre of musics. These audio files are chosen so we can see the relation between the music genre and the embedding result. In this evaluation, we also try to vary the parameters used in the embedding process including threshold values and segment size to analyze the effect of each parameter.

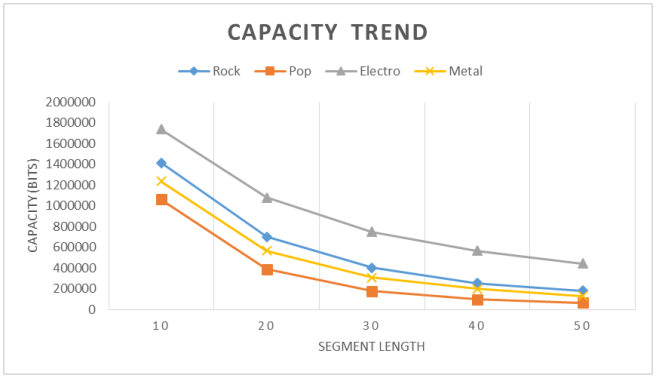

Fig. 14 Embedding capacity trend with segment length variation

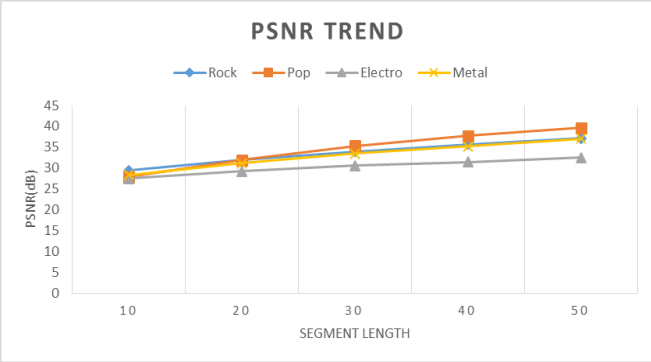

Fig. 15 PSNR trend with segment length variation.

To objectively measure the quality of embedding result we used PSNR value as a measurement. The PSNR value indicates the ratio between the power of a signal and the noise that distort it. Greater PSNR means that the reconstructed signal is closer to the original signal hence having a better quality. Figs. 14-19 show the trend of embedding result for each music genre pop, rock, electro, and metal. This test uses 5 embedding layers with various threshold values and segment lengths. Figs. 14-19 show the trend of the capacity and PSNR of the result.

In this evaluation using four different audio files, we initially set the threshold value to 50, about 1/5 of the maximum threshold. This value is considered quite strict, so the result is expected to have a good quality. From the result we can see that although the threshold limit is quite strict the proposed scheme can still embed a satisfactory amount of payload about 320.000 bit while still maintaining PSNR value as high as 37dB. With the PSNR value above 40, the clicking noise does not present and no artifact is noticed when the audio is heard, improving the transparency of the result and decreasing the detection rate by attacker.

From the experiment result, we also found that the embedding capacity and quality depends on the parameter used. For every 10 value increase in segment length, the embedding capacity decreases by 49.375% in average. This is because the longer the segment used for embedding, more sample will be checked for the threshold and embedding condition so there is a higher chance that some samples do not pass the check because of the overflow or because the sample value exceeds the threshold causing the whole segment to be ignored and not used for embedding. In exchange, the quality of the result increases by 8.24% in average because samples used for embedding are ensured to pass the condition checking.

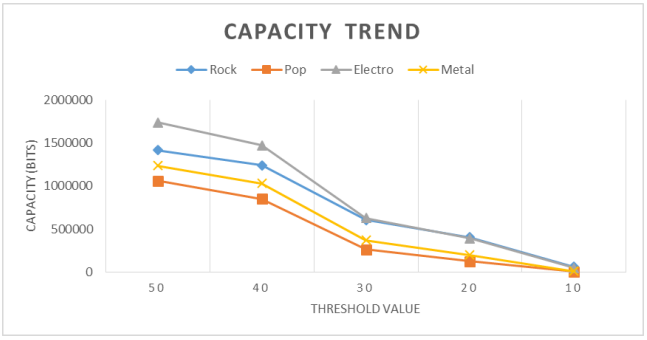

Fig. 16 Embedding capacity trend with threshold variations.

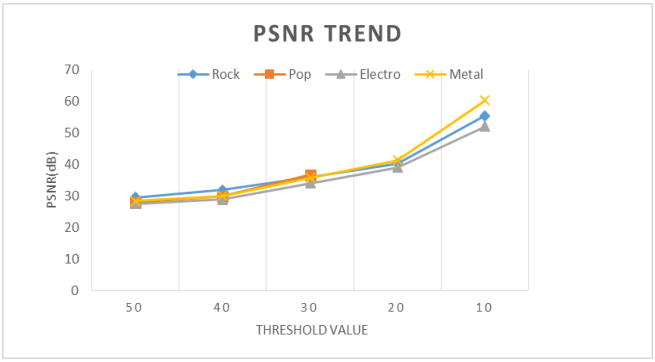

Fig. 17 PSNR trend with threshold variations.

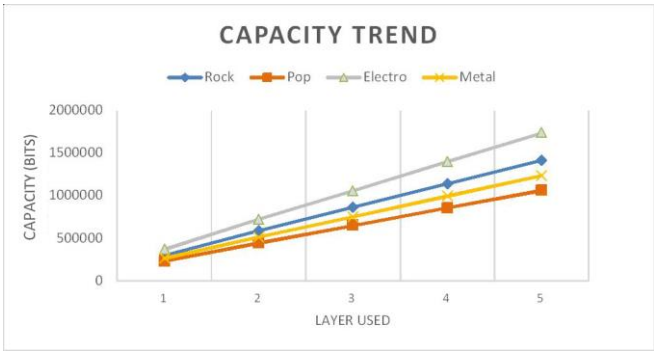

Fig. 18 Embedding capacity trend with multi-layer scheme.

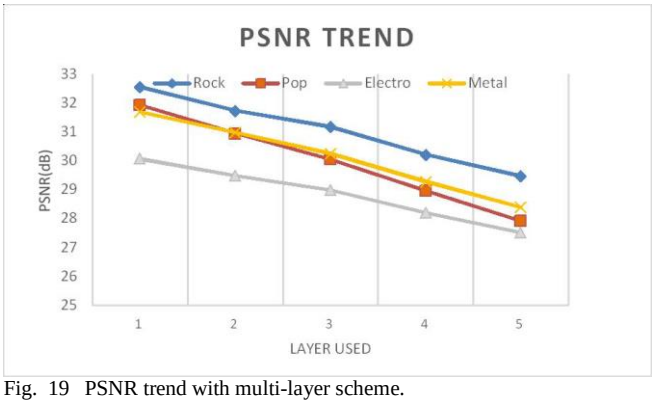

Fig. 19 PSNR trend with multi-layer scheme.

We can also see that every 10 value increase in the threshold increases the embedding capacity by 20,177%. This is because the higher the threshold value, the more samples in the segment will pass the threshold checking and more segment will be used for embedding. This will also cause the quality to decrease by 5.74% in average.

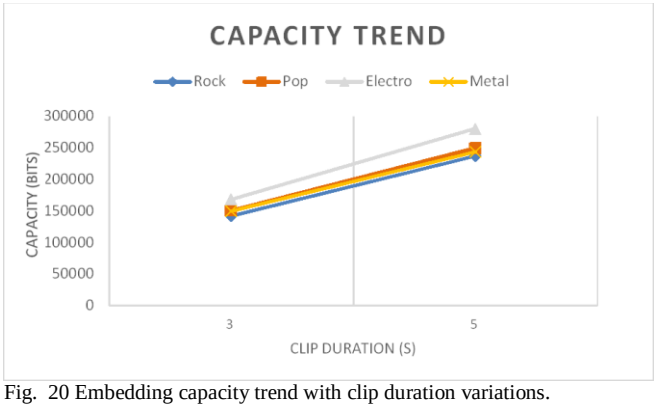

Fig. 20 Embedding capacity trend with clip duration variations.

Embedding using multi-layer scheme increases the embedding capacity significantly. For each layer added for embedding, we can add 82.47% of the original embedding capacity, in average. While the quality only decreases by 2.71% for each layer, we can see that the multi-layer scheme provide a good result when combined with IGRDE because the embedded value does not differ too far from the original value, so multiple bits can be embedded in the same sample. The maximum capacity for the multi-layer scheme is reached when no more data can be embedded in the sample, which is when all segments do not pass the threshold check and overflow check.

From the audio file used for the test, we can also see that a genre contributes to the result of the embedding process. Generally, the music with electro genre has the largest embedding capacity reaching 367578 bit followed by rock genre and metal genre, while the pop genre has the smallest embedding capacity. The capacity of the electro music is, however, reduced drastically with the change of the threshold; value while rock genre tends to decrease gradually. In the terms of PSNR, the pop genre has the best quality at 65.87 dB when tested with threshold parameter variation. The metal genre however, can match the quality of pop genre with a really subtle difference in the PSNR value.

The execution time of the process is quite the same albeit the parameter used is different. The execution time of 5 layer process requires about 1000 seconds in our testing environment with each layer takes about 200 seconds to process. This shows that the embedding process in each layer is same and does not require extra time.

The recovery test on our scheme succeeds to retrieve the cover file and the payload file without any information loss by comparing the samples values of original audio file with the retrieved audio file. The comparison of the payload file also shows that the payload is retrieved without any data or formatting loss. This result proves that the difference expansion method used on audio files still has its lossless or reversible trait.

C. Evaluation III

In this evaluation, the audio clip used as a cover is an identical audio clip used in the evaluation II but with a shorter duration. We cut the duration into 3 second and 5 second for each audio clip representing each genre and embed the same payload data to each audio clip. The embedding

process is done only in single layer. The embedding capacity result for each case is illustrated in Fig. 20.

In this evaluation we found that the difference in embedding capacity for each genre is less obvious than the result from the evaluation II where the audio clip used as a cover has a duration of 10 seconds. This result, in other way, can also shows that with shorter duration the difference between music genre is blurring; especially with the 3 seconds audio clip the embedding capacity is almost the same for pop, metal and rock genres. Electro genre still maintain the highest embedding capacity among all the genres used in testing.

Because the embedding capacity reflects how much changeable and expandable segment are there in the sample array of the audio file, we could say that in the short duration clip, the variation of the changeable and expandable segment in the each genre is subtle compared to longer duration audio clip where the sample amount is much larger.

From this result, we can also guess that the reason every genre delivers a different embedding capacity is because there are different variation of expandable and changeable segment. Form the result of the Evaluation II, the electro genre that gives most embedding capacity has the most expandable and changeable segment in its sample array while the pop genre has the least changeable and expandable segment.

V. Conclusion

In this paper, we propose a new audio steganography scheme that is based on generalized difference expansion and reduced difference expansion. We then improve the reduced difference expansion process by expanding the logarithm equation to get a smaller difference and better quality. To make the audio file compatible for difference expansion method we use an intelligent partition to change the sample size of the audio files into 8 bit which is later divided into two groups. The embedding is then done in both partitions.

The result of the proposed scheme is satisfying. With the capacity as high as 32000 bit, The PSNR value can still be maintained at 37db. Furthermore, the multiple layer scheme also contributes a lot in the terms of the capacity increase. By using the RDE we can maximize the multiple layer embedding ability in exchange for more computing time. We also find that the parameter used for embedding affects the overall embedding result. The larger threshold value is used then the capacity of the embedding will increase and the PSNR value will decrease. On the other hand, when the larger segment size is used, the lower capacity of the embedding and higher the PSNR value of the embedding result we obtain.

Acknowledgment

The authors would like to thanks Hitachi Global Foundation who has supported the research that was conducted at Kumamoto University, Japan.

References

[1] M. Kushima, K. Araki, M. Suzuki, S. Araki and T. Nikama, "Text Data Mining of In-patient Nursing Records Within Electronic Medical Records Using KeyGraph," *IAENG International Journal of Computer Science*, vol. 38, no. 3, pp. 215-224, 2011.

[2] Z. Kartit and M. El Marraki, "Applying Encryption Algorithm to Enhance Data Security in Cloud Storage," *Engineering Letters*, vol. 23, no. 4, pp. 277-282, 2015.

[3] J. M. Blackledge and A. I. Al-Rawi, "Steganography using Stochastic Diffusion for the Covert Communication of Digital Images," *IAENG International Journal of Applied Mathematics*, vol. 41, no. 4, pp. 270-298, 2011.

[4] M. H. A. Al Huti, T. Ahmad and S. Djanali, "Increasing the Capacity of the Secret Data using DE pixels Blocks and Adjusted RDE-based on Grayscale Images," in *International Conference on Information and Communication Technology and Systems*, Surabaya, Indonesia, 2015.

[5] W.-L. Tai, C.-M. Yeh and C.-C. Chang, "Reversible Data Hiding Based on Histogram Modification of Pixel Differences," *IEEE Trans. Circuits Syst. Video Technol*., vol. 19, no. 6, pp. 906-910, 2009.

[6] T. D. Kieu and C.-C. Chang, "A High Stego-image Quality Steganographic Scheme with Reversibility and High Payload using Multiple Embedding Strategy," *Journal of Systems and Software*, vol. 82, no. 10, pp. 1743-1752, 2009.

[7] G. Wang, X. Li and B. Yang, "High Capacity Reversible Watermarking Scheme based on an Integer Transform," in *Proc. 10th Pacific Rim Conf.*, Bangkok, 2009.

[8] J.Tian, "Reversible Data Embedding using a Difference Expansion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol.13, No 8, pp. 890-896, 2003.

[9] A. M. Alattar, "Reversible Watermark using Difference Expansion of Triplets," in *Proc. of the IEEE International Conference on Image Processing (ICIP 2003)*, 2003.

[10] A. M. Alattar, "Reversible Watermark using Difference Expansion of Quads," in *Proc. of the IEEE International Conference Acoustics, Speech, and Signal Processing*, 2004.

[11] A. M. Alattar, "Reversible Watermark using the Difference Expansion of a Generalized Integer Transform," *IEEE Transactions on Image Processing*, vol. 13, no. 8, pp. 1147-1156, 2004.

[12] D.C.Lou, M. Hu and J. Liu, "Multiple layer Data Hiding Scheme for Medical Images," *Computer Standards & Interfaces*, vol.31, no.2, pp. 329-335, 2009.

[13] T. Ahmad, M. Holil, W. Wibisono and R. M. Ijtihadie, "An Improved Quad and RDE-based Medical Data Hiding Method," in *The IEEE International Conference on Computational Intelligence and Cybernetics (CyberneticsCom)*, Yogyakarta, Indonesia, 2013.

[14] Y. Yang, W. Zhang, D. Liang, N. Yu, "Reversible Data Hiding in Medical Images with Enhanced Contrast," *Digital Signal Processing*, vol. 52, pp. 13-24, 2016.

[15] C.V. Kumar and V. Natarajan, "Hybrid Local Prediction Error-based Difference Expansion Reversible Watermarking for Medical Images," *Computers and Electrical Engineering*, vol. 53, pp. 333–345, 2016.

[16] I. Bilal, M. Roj, R. Kumar and P. Mishra, "Recent Advancement in Audio Steganography," in *International Conference on Parallel, Distributed and Grid Computing* (*PDGC*), Solan, 2014.

[17] F. Wang, Z. Xie and Z. Chen, "High Capacity Reversible Watermarking for Audio by Histogram Shifting and Predicted Error Expansion," *The Scientific World Journal*, p. 7, 2014.

[18] K.-C. Choi, C.-M. Pun and C. L. Chen, "Application of a Generalized Difference Expansion based Reversible Audio Data Hiding Algorithm," *Multimedia Tools and Applications*, vol. 74 Issue 6, March 2015, pp. 1961-1982, 2015.

[19] X. Wang, X. Li, B. Yang and Z. Guo, "Efficient Generalized Integer Transform for Reversible Watermarking," *Signal Processing Letters*, vol. 17 , issue 6) , pp. 567-570, 2010.

[20] H. J. Kim, V. Sachnev, Y. Q. Shi, J. Nam and H.-G. Choo, "A Novel Difference Expansion Transform for Reversible Data

Embedding," *IEEE Transactions on Information Forensics and Security*, vol.3, issue: 3 , pp. 456 - 465 , 2008.

[21] N. Cvejic and T. Seppanen, "Increasing the Capacity of LSB-based Audio Steganography," *IEEE Workshop on Multimedia Signal Processing*, pp. 336-338, 2002.

[22] M. Asad, J. Gilani and A. Khalid, "An Enhanced Least Significant Bit Modification Technique for Audio Steganography," *International Conference on Computer Networks and Information Technology* (*ICCNIT*), pp. 143-147, 2011.

[23] H. Kumar and Anuradha, "Enhanced LSB Technique for Audio Steganography," *Third International Conference on Computing Communication & Networking Technologies* (*ICCCNT*), pp. 1-4, 2012.

[24] S. Banerjee, S. Roy, M. Chakraborty and S. Das, "A Variable Higher Bit Approach to Audio," in *International Conference on Recent Trends in Information Technology* (*ICRTIT*), Chennai, 2013.

[25] X. Cao and L. Zhang, "Researches on Echo Kernels of Audio Digital Watermarking Technology Based on Echo Hiding," in *International Conference on Wireless Communications and Signal Processing* (*WCSP*), Nanjing, 2011.

[26] H. J. Kim and Y. H. Choi, "A Novel Echo-Hiding Scheme with Backward and Forward Kernels," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3 issues 8, pp. 885-889, 2003.

[27] O.-C. Chen and W.-C. Wu, "Highly Robust, Secure, and Perceptual-Quality Echo Hiding Scheme," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, issue 3, pp. 629-638, 2008.

[28] A. Delforouzi and M. Pooyan, "Increasing Payload of Echo Hiding Scheme Using Dual Backward and Forward Delay Kernels," *IEEE International Symposium on Signal Processing and Information Technology*, pp. 375-378, 2007.

[29] S. Ahani, S. Ghaemmaghami and Z. Wang, "A Sparse Representation based Wavelet Domain Speech Steganography Method," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, issue 1, pp. 80-91, 2014.