Improved prediction error expansion and mirroring embedded samples for enhancing reversible audio data hiding

Yoga Samudra, Tohari Ahmad *

Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

A B S T R A C T

Many applications work by processing either small or big data, including sensitive and confidential ones, through computer networks like cloud computing. However, many systems are public and may not provide enough security mechanisms. Meanwhile, once the data are compromised, the security and privacy of the users will suffer from serious problems. Therefore, security protection is much required in various aspects, and one of how it is done is by embedding the data (payload) in another form of data (cover) such as audio. However, the existing methods do not provide enough space to accommodate the payload, so bigger data cannot be taken; the quality of the respective generated data is relatively low, making it much different from its corresponding cover. This research works on these problems by improving a prediction error expansion-based algorithm and designing a mirroring embedded sample scheme. Here, a processed audio sample is forced to be as close as possible to the original one. The experimental results show that this proposed method produces a higher quality of stego data considering the size of the payloads. It achieves more than 100 dB, which is higher than that of the compared algorithms. Additionally, this proposed method is reversible, which means that both the original payload and the audio cover can be fully reconstructed.

1. Introduction

The rapid development of network technology at present has made data transmission not only faster but also more comfortable than ever. Many applications, such as financial transactions, can be carried out anywhere and anytime, as long as an internet connection and supporting devices are available. This growth, however, coincides with the increasing possibility of the sensitive data being compromised [1, 2, 3], including those in a big data environment [4, 5], considering the characteristics of users [6]. This condition has turned security into an essential factor, especially in transmitting and distributing those data. Some security approaches have been introduced, including cryptography and steganography. While the former works by changing the representation of information so that only the receiver can access the real information due to its unreadable format, the latter works by hiding the secret information inside a file called the cover. In its implementation, this cover can be any type of data: image [7][8], video [10], or audio [9][11]. Furthermore, text can also be utilized for the same purpose [12]. Steganography and data hiding refer to the same definition in many research papers, so we use them interchangeably.

Aside from cryptography, the usage of data hiding as a form of a security approach has proven to be reliable and is widely implemented in various environments, such as in the multi-cloud storage [13]. The main reason why this method has become crucial is that the information is sent to the receiver in the same type as that of the corresponding cover, called the stego file. Furthermore, its content is similar to the original medium file; thus, it should not raise too much suspicion towards the illegal parties. At the application level, this data hiding method is often combined with cryptography to make it more difficult for the attackers to exploit the protected data [14, 15]. Besides, if the stego file happened to be leaked, there is just not much information that the attackers can use to retrieve the secret due to many numbers of methods that could be used [16].

Regardless of the medium type being used, several issues have been challenging since its first use, which include payload capacity, stego quality, robustness, embedding efficiency, and detectability [17]. Those two first parameters are the primary requirements to meet, which can be described as follows.

The first issue, the capacity, is the payload size that can be embedded inside a cover file. Some research focuses on how to optimize this
embedding capacity with various methods such as histogram shifting [18, 19], generalized difference expansion [20], and adaptive embedding [21]. The second is that the quality of the stego file generated by the embedding process should be as high as possible. If somehow the quality of the stego-file is low, there is a higher chance that the stego file can be found suspicious because of the big difference in the sample between this file and its corresponding cover file [22]. To prevent this issue, some research specifically deals with how to optimize the stego file quality is introduced, such as [23]. However, it is worth noting that these first and second problems usually are inversely proportional. Therefore, some research works on these two problems at once, such as our previous research [24][25]. In this research, we focus on these problems since the others are derived from them.

The embedding efficiency considers the change of the cover caused by the payload embedding process. A higher number of payloads with a more negligible effect on the cover is better. Overall, this measurement is affected by the previous first and second parameters. The detectability relates to the possibility of distinguishing the stego from its cover. This parameter is derived from the second one, i.e., the quality factor. Robustness relies on the specific environment [17], so it may not be the same for different applications.

Considering the importance of the cover, some research concerns whether it needs to be retrieved or not. In the case of security, it would be ideal to replace the stego file with the cover once the extraction process has been done [26]. Thus, data hiding methods often need to be reversible, such as [27, 28], which are mostly developed based on histogram shifting and difference expansion. One of the disadvantages of these two basic methods is that they are introduced explicitly to the image as the cover file. So, it needs to adopt some changes before being applied to other types of media.

This research considers audio to be the cover. We believe that even though it is less popular than image and video [11], audio has the potential for exploration according to its characteristics. The proposed method is developed based on Prediction Error Expansion (PEE) [20, 29], where their capacity per sample is relatively low, and so is its corresponding stego quality. The proposed improvement is, in general, designed by dynamically assigning and spreading the payload to every sample, and multiple mirror embedding to maintain the capacity and quality, as well as the reversibility.

The following sections are provided in the next parts of this paper. First, the literature study is depicted in Section 2. Then, the proposed method is explained in Section 3, while Section 4 analyzes the experimental results, and Section 5 discusses the characteristics of the technique. To sum up the paper, the conclusion is drawn in Section 6.

2. Data hiding techniques

In the recent study of data hiding, there have been various trends from how it started to focus solely on optimizing the embedding capacity and quality of the stego file and then utilize computational intelligence for optimization into data hiding techniques or even as the technique itself. Some of those techniques such as Least Significant Bit (LSB) [30, 31], bit-plane index manipulation [32], Fibonacci-like employing bit-plane(s) mapping [33], and the early introduced Difference Expansion (DE) [34] have become fundamental for the development of new techniques such as FCM Clustering [35], Generalized Difference Expansion (GDE) [20], Least Significant Digit (LSD) [36], Pixel Value Modification (PVM) [37] and Prediction Error Expansion (PEE) [29, 38]. Some of these techniques can also be further enhanced with optimization methods; some also proposed a data hiding scheme with an encrypted image [39], watermarking encryption [40] or fragile watermarking for the authentication purpose [41].

Some research proposes an audio file as a cover; its primary purpose is to obtain a reasonable embedding capacity. Using audio instead of an image, the capacity of data embedded inside the cover significantly increases.

Jung and Yoo [30] present a reversible data hiding method by using an audio file using linear interpolation. This interpolation is needed to create some sets of new samples based on the original ones. The confidential data are then embedded into the interpolating samples using Reduced Difference Expansion. However, the constructed stego audio has different characteristics from its original audio, which makes them incomparable. It is because the number of samples is twice as much as the original space. Consequently, the frequency needs to be adjusted to twice its original number.

Still using an audio file as the cover, Andra et al. [28] present a reversible data hiding method while maintaining the quality of stego audio as well as its embedding capacity. Medical records are employed as secret information. The Intelligent Partition is implemented by changing the structure of audio samples from an array of samples into a 2D array of big. By performing this change, Generalized Difference Expansion can optimize its embedding capacity. Furthermore, its capacity is enhanced by using multi-layered embedding so that a sample might have a chance to be embedded more than once.

Unfortunately, this embedding method produces lower PSNR values. To compensate for this downside, Reduced Difference Expansion is applied. It is also to close the big difference between stego and original sample values. Hence, the scheme can satisfy both the embedding capacity and quality of stego audio but with the disadvantage of requiring additional information besides the stego audio itself, such as Reduced Map, Location Map, and other parameters. The experimental result shows that an audio cover with the electro genre is prone to have more spaces for embedding but has a lower quality of stego audio after being embedded.

On the other hand, Bobeica et al. [29] propose reversible audio-based data hiding by using Prediction Error Expansion [42]. Although not as complicated as [28], this method has proven to be reversible. Nevertheless, this scheme limits the embedding capacity, which can only hold 1-bit secret data per sample. This research presents a reversible method using an audio file's capacity control for Prediction Error Expansion. Initially, audio samples are divided into two groups: one reserved for additional information and the other selected samples for embedding. Then, these selected samples would be further divided into two sets, namely cross and dot. The embedding process starts with determining each prediction error value from both sets. Prediction context is formed around a determined sample using neighboring samples, which is one sample before and after the determined sample.

Xiang and Li [42] use a much more complex prediction context that utilizes 60 neighboring samples, which is 20 samples before and 40 future samples after. These prediction error values are then processed for embedding along with the initial threshold value, which is determined after sampling the audio. Its value is specified by some basic information such as bit size of payload data, the number of stored LSB in the reserved sample area, and the estimated number of samples at risk of overflow or underflow. After that, the prediction error histogram can be calculated. Hence, the matched threshold value that satisfies the calculation can be found. Unfortunately, the number of bit data that can be embedded is limited to only 1 bit per sample. Overall, this scheme outperforms that of Andra et al. [28], concerning the PSNR value. On the other hand, some data hiding methods can be implemented for different purposes, such as in [43, 44, 45, 46].

3. Audio-based improved PEE and mirroring approach

This research is intended to overcome the limitations of the previous scheme by optimizing its performance. We propose an improved reversible data hiding method that is based on [29], where it has a critical downside. That is, the limited and relatively small capacity of each sample, for which the quality of the stego file is considered not good enough. This proposed method revolves around the flexibility of the neighboring samples that comprise the prediction context.
This improvement will optimize the capacity by combining the Prediction Error Expansion (PEE) with an equal-sample distribution approach. This approach makes the sample capacity dynamic while still using all existing sample spaces. Additionally, this considers the sample’s quality by spreading hidden data to every sample. Then, to further optimize quality, the embedding approach uses mirror embedding. It works by repeatedly reflecting the embedded value with a mirror point until it is considered close enough to the original sample value and can be extracted later on. The general process of the proposed method is provided in Fig. 1 and Fig. 2, which describe the embedding and extraction processes, respectively. In detail, the proposed method can be described as follows.

### 3.1. Pre-processing stage

As shown in Fig. 3, initially, the obtained audio samples are pre-processed before being embedded by the payload. It is to prepare and optimize the next embedding results. The detailed steps can be described as follows.

1. Get the audio samples.
2. Normalize every value of audio samples by adding it with 32768 to change the domain values from 16-bit integer to unsigned 16-bit integer (all positive integer).
3. Separate these samples into three different areas, which are: the first block sample consisting of 1-100 first samples, the selected area, and the last block sample comprising 1-100 last samples like in Fig. 4. The total samples between the first block and the last block (both blocks called reserved area) must be equal, whose number is determined at the beginning. The selected area comprises samples located between these two reserved blocks.
4. Once the samples have been allocated, calculate the bit rate from both the payload’s and the cover’s perspective sides by using (1) - (5). Here, \( R_p \) and \( R_c \) are the bit rate of the payload and the cover respectively, \( n \) represents the number of total samples of the cover audio file; \( P_{size} \) and \( C_{size} \) depict the size of the payload and the cover, whose value is defined in (3). Next, \( r_1 \) and \( r_2 \) are respectively the number of total reversed samples at first and the last blocks; \( e \) represents the prediction error value, and \( P_x \) denotes the average neighboring sample value \( x \).
5. Determine a threshold value \( T \) to be used for comparison in the next step. It is ideal for initializing this threshold value to 0, which can be incremented if needed later.

6. Compare \( R_p \) and \( R_c \) values as in (6) to determine the possibility whether a payload can be embedded into cover audio or not.
7. If the result of the previous step is 0, then the payload cannot be embedded to the cover audio file. If the result is 1, then continue to the embedding stage.
8. If the embedding stage fails to embed all payload data, then increment the threshold value \( T \) by 1 and repeat step 6.

\[
R_p = \frac{8 \times P_{size}}{n} \tag{1}
\]
\[
R_c = \frac{C_{size}}{n} \tag{2}
\]
\[
C_{size} = \sum_{i=1}^{n-2} \left[ \log_2(e_i) \right] \tag{3}
\]
\[
e_i = |x_i - P_{x_i}| \tag{4}
\]
\[
P_{x_i} = \frac{\left( \sum_{j=r_1}^{r_2} x_j \right) - x_i}{r_1 + r_2} \tag{5}
\]

These two-bit rate values in (1) and (2) are acquired by using PEE not as embedding means but as pre-process embedding. Instead of providing the actual value, it gives the average neighboring values approaching its actual sample value. An example of this construction is demonstrated in Fig. 5.

After both of these \( R_p \) and \( R_c \) values have been found, they are compared to determine whether the payload data can embed the cover audio or not by using (6). Here, \( T \) represents the tolerance value, which is initiated with value 0. If the status of the bit rate (\( S_t \)) is equal to 1, then the cover audio can be embedded; thus, we continue to the embedding stage. Otherwise, increment the \( T \) value by 1 and repeat the process in (6) again. This process stops when the combination of \( R_p \) and \( T \) produces a value higher than \( R_c \), implying that the cover audio cannot be embedded with the payload data.

\[
S_t = \begin{cases} 
1, & \text{if } (R_c \geq [R_p] + T) \\
0, & \text{if } (R_c < [R_p] + T)
\end{cases} \tag{6}
\]

### 3.2. Embedding stage

This stage is performed after the audio samples have been pre-processed. Bit rate values from both perspectives are necessary to calculate the amount of bit data that has to be embedded later. The PEE
The approach is designed to find the prediction error by using neighboring sample values before and after the sample is processed. Different from the PEE approach in the previous research, where the prediction error value is obtained and used later in the embedding process, the prediction error value in this proposed approach is only to calculate the number of bits that a sample can hold.

The general flow of this process is shown in Fig. 6; meanwhile, the detailed explanation of this process can be presented in the algorithm below.

1. Start from the first sample in the selected area.
2. Find the average of the neighboring sample value of the determined sample by using the PEE approach (for an illustration, see the previous Fig. 5). The number of the neighboring samples used is determined from the pre-process embedding.
3. Determine the prediction error value, which is obtained by calculating the difference between the average neighboring sample and original sample values, as shown in (4).
4. Calculate the total bits that can be embedded in the determined sample $i$, which is represented as $S_i$ in (7).

$$S_i = \lfloor \log_2 |e_i| \rfloor$$  \hspace{1cm} (7)

5. Compute how many bits ($E_i$) should be embedded in the sample. It is carried out by comparing the result of (7) with the bit rate value from the payload’s perspective (1). The comparison and calculation processes are provided in (8) and (9), respectively, where $T$ represents the tolerance value as mentioned before in the pre-processing.
step, and $R_p$ denotes the number of bits per sample from the payload's perspective.

$$E_{\text{status}} = \begin{cases} 1, & \text{if } (S_i \geq R_p + T) \\ 0, & \text{if } S_i < R_p + T \end{cases}$$

(8)

$$E_i = \begin{cases} [R_p + T], & \text{if } E_{\text{status}} = 1 \\ S_i, & \text{if } E_{\text{status}} = 0 \end{cases}$$

(9)

6. Get the amount of bit data ($E_i$) from the payload based on the previous step's result.
7. Determine every value in the prediction error value range as the mirror point. It is illustrated in Fig. 7.
8. Convert payload’s bit data into decimal value using (10), where $Pd$ is the decimal value obtained from $Pb(E_i)$ representing the payload data with $E_i$ bit is taken.

$$Pd_i = \text{bit2dec}(Pb(E_i))$$

(10)

9. Calculate the embedded value from the average neighboring sample to approach the original sample value by using every mirror point to process it. So, this value comes as close as possible to the original sample. This can be done with (11), then (12). The notation $C(M_i)$ represents the number of mirror points existing within the $i$th sample, while $Pd_i$ depicts the new value to be embedded in the audio sample. The illustration of this step is shown in Fig. 5.

$$C(M_i) = \left[ \frac{|e_i|}{2^{E_i}} \right]$$

(11)

$$Pd'_i = \begin{cases} x_i + |e_i| + Pd_i, & \text{if } C(M_i) \text{ is even} \\ x_i + |e_i| + 2^{E_i} - Pd_i, & \text{if } C(M_i) \text{ is odd} \end{cases}$$

(12)

10. Embed the decimal value taken from the average neighboring sample value ($P_{x_i}$) (see (5)) of the determined sample into the value of the original sample. The process is carried out using (13). Here, $P_{x_i} \neq x_i$; so, the value of $P_{x_i}$ must be either more or less than $x_i$. The set of $N_{x_i}$ is used as stego-audio sample values.

$$N_{x_i} = \begin{cases} x_i - Pd'_i, & \text{if } P_{x_i} > x_i \\ x_i + Pd'_i, & \text{if } P_{x_i} < x_i \end{cases}$$

(13)

11. In case the new sample ($N_{x_i}$) value is either underflow or overflow, we record the index sample and note it temporarily as out of bound sample. Then, these values are reflected with their respective last mirror points.
12. Repeat the process from step 2 to step 11 until all the payload data have been processed.
13. Construct a new audio file with the same specification as the original audio but filled with new samples as the embedding result.

Some additional information also needs to be processed so that the extraction process can be carried out. This information includes the necessary information, the original sample value, and the index sample, which may suffer from overflow or underflow.

The necessary information needed for the extraction process is listed below:

- The number of samples in the reserved area ($R$).
- The number of neighboring samples used for the PEE approach.
- The last index sample being processed that contains the last chunk of bit data of payload.
- The amount of bit data embedded in the last processed sample.
- Bit rate per sample from payload’s perspective.
- Tolerance limit ($T$).
- A flag whether the audio sample has an existing overflow/underflow sample (Boolean value)

These values will be embedded in the reserved area ($R$) by using the simple LSB technique, as depicted in (14) with $b$ as the bit representing the information. The slightest bit of information of the original sample value still contains as $I$ by expanding the information with either ‘1’ or ‘0’ based on the LSB value comparison between the previous original sample value (before being processed) and that of after being processed as in (15).

$$N_{x_i} = 2 \times \left( \left\lfloor \frac{x_i}{2} \right\rfloor + b, \quad i \in Z^+ \right)$$

(14)

$$I_i = \begin{cases} 0, & \text{if } x_i \mod 2 = 0 \\ 1, & \text{if } x_i \mod 2 \neq 0 \end{cases}$$

(15)

The information of the original sample value can be contained as the absolute difference between the original sample value ($x_i$) with the last mirror point formed on the same sample (see (16)). This value is then converted to its binary value with total bits ($E_i$) following the chosen bit as in (9).

$$I'_i = |x_i - (C(M_i) \times 2^{E_i})|$$

(16)

The last information that would be needed is whether the overflow or underflow samples exist or not. If there is no sample suffering from this condition, then this information can be ignored. However, if some samples exist, only then should this information be created.

### 3.3. Extraction stage

The extraction process is performed in a similar way to the embedding, but it is done in the reverse order, as shown in Fig. 8. Furthermore, the PEE approach is also being used to determine the prediction error value in each sample. The basic information in the reserved area ($R$) is essential, as it also explains where the last processed sample is located. Before this process is done, it also needs the pre-processing stage, but it is only limited to translating the additional information obtained from the previous embedding process. The explanation of the extraction process is presented in the algorithm below.

1. Start from the specified last processed index sample.
2. Find the average neighboring sample value of the determined sample by using the previous variation of the PEE approach (see Fig. 5 for an example). The number of neighboring samples used is determined at the beginning.
3. Calculate the prediction error value based on the average neighboring sample ($P_{x_i}$) and stego sample ($x'_i$), as in (17).

$$e'_i = |x'_i - P_{x_i}|$$

(17)
4. Compute the total bit data that can be embedded in the determined sample, as in \((18)\).

\[
S'_i = \lfloor \log_2 |e'_i| \rfloor \tag{18}
\]

5. Further calculate the number of bits that have been embedded by comparing the result between \((18)\) and the sum of bit rate value from the payload’s perspective \((R_p)\) and tolerance \((T)\), which are obtained from the additional information reserved in the reserved area \((R)\). The comparison and calculation process can be seen in \((19)\) and \((20)\).

\[
E_{status} = \begin{cases} 
1, & \text{if } S'_i \geq (R_p + T) \\
0, & \text{if } S'_i < \text{mod}(R_p + T)
\end{cases} \tag{19}
\]

\[
E_i = \begin{cases} 
R_p + T, & \text{if } E_{status} = 1 \\
S'_i, & \text{if } E_{status} = 0
\end{cases} \tag{20}
\]

6. Determine every value in the range of the prediction error as a mirror point, as in Fig. 7.

7. In case the sample is recorded as either an overflow or underflow sample, remove the furthest mirror point from prediction error values.

8. Extract the hidden decimal value by using \((21)\) and \((22)\).

\[
C(M_i) = \left\lfloor \frac{|e'_i|}{2^{S'_i}} \right\rfloor \tag{21}
\]

\[
Pd'_i = \begin{cases} 
|e'_i| - (2^{S'_i} \times C(M_i)), & \text{if } C(M_i) \text{ is even} \\
|e'_i| - (2^{S'_i} \times C(M_i) + 1), & \text{if } C(M_i) \text{ is odd}
\end{cases} \tag{22}
\]

9. Revert the stego sample into its original one, from the information of the processed original sample value of embedding \((x'_i)\) using \((23)\). Equivalent to that in the embedding process step 10 in Section 3.2, is that \(P_x'_i \neq x'_i\).

\[
x_i = \begin{cases} 
P_x'_i + (2^{S'_i} \times C(M_i) + x'_i), & \text{if } P_x'_i < x'_i \\
P_x'_i - (2^{S'_i} \times C(M_i) + x'_i), & \text{if } P_x'_i > x'_i
\end{cases} \tag{23}
\]

10. Convert the decimal value to its binary representation \((b)\) by using \((24)\).

\[
b_i = \text{dec}2\text{bit}(d'_i) \tag{24}
\]

11. Repeat the process from step 2 to step 10 until the first sample of Selected area \((S)\) has been processed.

12. Construct the original audio file.

13. Construct the payload data.

4. Experimental analysis

The proposed method is evaluated by using some scenarios, whose design can be described as follows.

4.1. Experimental design

The experiment takes input consisting of two file types. The first is audio files to be the cover, taken from a public data set [47], which comprises 15 audio files, each of which plays 3 seconds. In more detail, this data set comprises three genres: Classical, Country-Polk, and Pop-Rock; each of them plays five instruments: Acoustic Guitar, Voice, Saxophone, Piano, and Cello, as depicted in Table 1. In this research, the audio covers are in the form of 16-bit depth mono audio of *.wav files, whose frequency is 44100 Hz. This construction is the same as other research, such as [23]. The second is payload files in a binary format. This payload is constructed by various sizes of data, from 1 kb to 100 kb, representing different types of secret data to protect, as shown in Table 2.

Controlled variables in this experiment are threshold value \((T)\), total reserved sample on the first sample \((r_1)\), and the last sample \((r_2)\). So, we conduct 2 types of experiments. The first is to observe the effect of neighboring samples or reserved samples value on the quality of stego-audio by using 6 random audio files (from the previous 15) and 1 payload data with a size of 300 kb. The second is to compare this method with previous ones by using 15 cover audio files and 11 text-based payload data.

In the experiment, the range of total reserved samples \((r_1\) and \(r_2)\) to use is 1-100, with a threshold value starts from 0. This experiment serves 2 purposes: to observe the effect of total reserved samples used on the quality of the stego audio and to find the best total reserved samples to be taken throughout the second experiment. Each payload in Table 2 is embedded in all audio covers in Table 1. Therefore, we have data representing the quality of each corresponding stego audio resulting from each combination of them. The quality factor closely relates to the capacity because a higher quality stego file can carry more payloads before its value falls to the minimum threshold (see Section 5).

Similar to other research, the quality is represented by Peak Signal to Noise Ratio (PSNR) in dB, whose value is inversely proportional to the Mean Square Error (MSE). Their formula is depicted in \((25)\) and \((26)\). Here, \(x(i,j)\) and \(x'(i,j)\) depict respectively the cover and the stego samples located in \((i,j)\). It is worth noting that the audio samples are in 1D, instead of 2D, unlike images. So, the value of both \(m\) and \(n\) follow this characteristic. Here, the higher the PSNR value, the better the quality. It means that the stego is closer to its original cover.

Before starting the experiment, the variable values need to be initialized, such as the number of neighboring samples and the tolerance value. Initially, the tolerance value is set to 0 so that all samples can be embedded with a few bits as possible.

\[
\text{MSE} = \frac{1}{m \times n} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} [x(i,j) - x'(i,j)]^2 \tag{25}
\]

\[
\text{PSNR} = 10 \times \log_{10} \left( \frac{\text{MAX}^2}{\text{MSE}} \right) \tag{26}
\]

4.2. Number of neighboring samples

To determine the neighboring samples, we conduct a separate experiment specifically to find the best number of neighboring samples that will be carried out through the whole experiment. It can be done...
Table 1. Audio Data Set

| Cover No | Audio cover | Genre, Instrument       |
|----------|-------------|-------------------------|
| 1        | Audio 1     | Cello, Country-Folk     |
| 2        | Audio 2     | Cello, Classical        |
| 3        | Audio 3     | Cello, Pop-Rock         |
| 4        | Audio 4     | Acoustic guitar, Country-Folk |
| 5        | Audio 5     | Acoustic guitar, Classical |
| 6        | Audio 6     | Acoustic guitar, Pop-Rock |
| 7        | Audio 7     | Piano, Country-Folk     |
| 8        | Audio 8     | Piano, Classical        |
| 9        | Audio 9     | Piano, Pop-Rock         |
| 10       | Audio 10    | Saxophone, Country-Folk |
| 11       | Audio 11    | Saxophone, Classical    |
| 12       | Audio 12    | Saxophone, Pop-Rock     |
| 13       | Audio 13    | Voice, Country-Folk     |
| 14       | Audio 14    | Voice, Classical        |
| 15       | Audio 15    | Voice, Pop-Rock         |

Table 2. Various Sizes of Payload.

| Payload no | Size (kb) |
|------------|-----------|
| 1          | 1         |
| 2          | 10        |
| 3          | 20        |
| 4          | 30        |
| 5          | 40        |
| 6          | 50        |
| 7          | 60        |
| 8          | 70        |
| 9          | 80        |
| 10         | 90        |
| 11         | 100       |

Fig. 9. Comparison of the average PSNR values generated by various numbers of neighboring samples.

by comparing the average PSNR value from some values in the range of the determined neighboring samples. For this purpose, we use six random cover audios and one payload file containing 300,000-bit data, and then compare the value between 1 and 100; thus, we generate 600 stego audio files. Based on the generated data, then we calculate the average PSNR and compare it with each other. The result is presented in Fig. 9, which shows that their performance is not significantly different. We find that the best PSNR value is obtained when the number of neighboring samples is 3. In this number, 87.59 dB is obtained. Furthermore, this number of samples is relatively small, which is good. So, it is used throughout the entire experiment.

4.3. Quality of the stego audio and its corresponding capacity

After embedding all sizes of payloads in Table 2 to each audio cover in Table 1, we obtain the PNSR values, which can be plotted in Fig. 10. Firstly, this result shows that the payload capacity reduces the quality of the stego file, which has been predicted before. For example, by taking 1 kb, the quality of the resulted stego files is the best for all types of covers, which is, on average, 120.22 dB, while 100 kb results in 100.55 dB.

Nevertheless, the distance of the PSNR values being generated by each capacity of payloads is not similar, even though this payload capacity increases uniformly (see Fig. 11). That is, the higher the number of embedded bits, the lower the decrease of PSNR values. For example, embedding 1 kb, 10 kb, and 20 kb produce 120.22 dB, 110.53 dB, and 107.53 dB, which means that the drop is 9.69 and 3.00, respectively. Finally, by using 90 kb and 100 kb, the obtained PSNR values are 101.01 dB and 100.55 dB, respectively, and their difference is 0.46. It is predicted that this pattern also applies to larger payload sizes. It can be inferred that in a certain number of bits, the quality of the generated stego file is relatively stable.

4.4. Audio genres

In this scenario, we evaluate the relation between the performance of the proposed method and the genre of the audio covers for various sizes of the payload, whose result has been depicted in Fig. 10. According to these data, the stego files of Audio 4 (Acoustic guitar, Country-Folk) are often slightly better than others. However, we see no specific pattern of this trend; we see that audio genres may not significantly affect the quality. In more detail, the difference between the maximum and the minimum PSNR values of each payload size for various genres is less than 0.4 dB; the standard deviation calculated based on 1 kb is 0.13, while that of other payload sizes is even less than 0.05. It is known that those stego files are relatively uniform, regardless of the audio genres.

It is different from that of [23] in that the quality relies on the genre. Nevertheless, they may be incomparable because the fundamental theorem of [23] is quite different. Furthermore, that research applies an interpolation scheme, which makes the number of samples two times bigger, while in this proposed method, it is constant. Therefore, the size of the stego file of this proposed method is relatively the same as that of its corresponding cover, which means that it looks like a conventional audio file.

4.5. Other research

Further evaluation is performed by comparing the performance of the proposed method with other research [24, 25, 28, 29]. For this purpose, we implement them whose experimental results, along with that of the proposed method, are provided in Fig. 12, which shows the average of their PSNR values. As shown in Fig. 12, the proposed method can maintain the audio quality in PSNR to at least 100 dB. With the same amount of payload data embedded, the research in [29] achieves about 43 dB of PSNR value, while that in [28] is around 35 dB. Similar to that of [24, 25], the PSNR value generated by the proposed method slightly decreases along with the increase of the payload size. However, its value is still higher than that of [24, 25].
It is found that this significant difference in quality is mainly caused by the design of the mirroring process in the proposed method. This process has also made almost no difference whether the prediction error value is relatively small or big. To enhance its quality further, we also distribute the payload data to be embedded almost evenly on every sample. Also, both of these methods [28, 29] maintain a stable PSNR value as the payload size increases. According to the average of the PSNR values in Fig. 12, it can be inferred that the proposed method can hold any size of payloads better than others especially for a smaller payload it is much better.

Next, we take the highest PSNR value that [28, 29] and [24, 25] can achieve, which is done respectively by using Audio 2 and Audio 13 as the cover. Figs. 13 and 14 depict this performance along with that of the proposed method. In Fig. 13, it is shown that by using this cover only, the PSNR value of both [28] and [29] increases to about 47 dB and 68 dB, respectively, while that of the proposed method is relatively stable, similar to its average value. It means that the proposed method generates the stego files whose quality is the highest among them for all audio covers. Similar to this pattern, the proposed method is also better than others, as depicted in Fig. 14.

5. Discussion

Unlike in [29], some changes are made in this proposed method, such as the sampling method, determining threshold value, determining sample distribution, and the embedding method itself. The result of sampling is not processed, as its range values vary from a negative to a positive integer, but instead, it is normalized into all positive integers so that the calculation can be made simpler. Furthermore, the threshold value is not determined by basic information from previous research, but instead, it uses the information that has been processed, such as the average bit per sample that can be embedded from both perspectives (either payload or cover side). Thus, this scheme does not form a prediction error histogram for further calculation on the threshold. Samples are processed almost the same as in PEE without further dividing into cross and dot sets.

The proposed embedding method also utilizes the concept of continuous mirror embedding so that the fewer bit data being embedded in a sample, the less of a difference for the stego-audio. Moreover, this method also evenly allocates the bit rate per sample, resulting in the maximum usage of every sample in an audio file. Meanwhile, some samples that are considered overflow or underflow are processed further.

Overall, capacity and quality are two main focuses in this research, which relate to each other. If the stego file has a better quality for the specified level, then it means that it can carry more data (higher capacity) for that respective quality level and vice versa. Let \( Q_c \) and \( C_c \) be respectively the quality and the corresponding capacity of a stego file, and \( Q_{\text{min}} \) is the minimum allowed quality level. A better RDH method can take more data before the stego file quality \( Q_c \) decreases to \( Q_{\text{min}} \); while another RDH method may only take small data, but its \( Q_c \) drop to \( Q_{\text{min}} \) quickly. Therefore, this proposed method works on these two factors as described in the previous sections.

6. Conclusions

In this paper, we proposed a Prediction Error Expansion-based data hiding scheme. Furthermore, we improved the embedding method by twisting Reduced Difference Expansion into a new method that we called Mirror Embedding in order to maintain a better quality of the
The embedding capacity in this method depends on the average bit per sample on the cover’s perspective side, which can be either relatively big or small, depending on the cover audio itself. The performance of the proposed method is satisfying. By embedding 100,000-bit data inside a three-second audio clip, the PSNR value can still be maintained above 100 dB. The usage of mirror embedding has proven effective in maintaining and increasing the PSNR value by embedding the bit data as close to the actual sample as possible. Furthermore, the concept of distributing bit data evenly among the audio samples has made the embedding capacity looks dynamic, thus raising the PSNR value even more. Besides, the bigger the payload data size, the lower the quality of the stego-audio will be. Moreover, as long as the bit per sample value from the payload perspective side is not higher than that of the cover perspective side, the payload data can be embedded completely.

Declarations

Author contribution statement

Yoga Samudra: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.
Tohari Ahmad: Conceived and designed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.
Funding statement
Tohari Ahmad was supported by Kementerian Riset dan Teknologi/Badan Riset dan Inovasi Nasional, Indonesia (901/PK5/ITS/2021).

Data availability statement
Data will be made available on request.

Declaration of interests statement
The authors declare no conflict of interest.

Additional information
No additional information is available for this paper.

References
[1] I.H. Sarker, A.S.M. Kayes, S. Badsha, Cybersecurity data science: an overview from machine learning perspective, J. Big Data 7 (2020) 41.
[2] H.-Y. Tran, J. Hu, Privacy-preserving big data analytics a comprehensive survey, J. Parallel Distrib. Comput. 134 (2019) 207-218.
[3] M. Hirano, N. Tsuzuki, S. Ikeda, R. Kobayashi, LogDrive: a proactive data collection and analysis framework for time-traveling forensic investigation in IaaS cloud environments, J. Cloud Comput. 7 (2018) 18.
[4] C.L. Calvert, T.M. Khojaghofaa, Impact of class distribution on the detection of slow HTTP DoS attacks using Big Data, J. Big Data 6 (2019) 67.
[5] K.A. Jadall, M. Aljinni, M.S. Dosouqi, Big data analysis and distributed deep learning for next-generation intrusion detection system optimization, J. Big Data 6 (2019) 88.
[6] A.L. Hadlington, Human factors in cybersecurity; examining the link between internet addiction, impulsivity, attitudes towards cybersecurity, and risky cybersecurity behaviours, Helvion 3 (7) (2017).
[7] I.J. Radhim, P. Premaratne, P.J. Vial, A. Halloran, Comprehensive survey of image steganography: techniques, evaluations, and trends in future research, Neurocomputing 335 (2019) 299-326.
[8] P. Puteaux, S.Y. Ong, R.S. Wong, W. Puech, A survey of reversible data hiding in encrypted images – the first 12 years, J. Vis. Commun. Image Represent. 77 (o) (September 2020) 103085.
[9] A. Aslabbany, A.H. Ali, F. Ridznan, A.H. Arzni, M.R. Mokhtar, Digital audio steganography: systematic review, classification, and analysis of the current state of the art, Comput. Sci. Rev. 38 (2020) 100316.
[10] Y. Liu, S. Liu, Y. Wang, H. Zhao, S. Liu, Video steganography: a review, Neurocomputing 335 (2019) 238-250.
[11] Y. Shi, X. Li, X. Zhang, H. Wu, B. Ma, Reversible data hiding: advances in the past two decades, IEEE Access 4 (2016) 3210-3237.
[12] A.A. Gurbah, K.A. Alaseri, Refining Arabic text stego-techniques for shares memorization of counting-based secret sharing, J. King Saud Univ. Comput. Inf. Sci. 33 (9) (2021) 1108-1120.
[13] L.M. Metcheka, N. Ndoundam, Distributed data hiding in multi-cloud storage environment, J. Cloud Comput. 9 (2020) 68.
[14] K. Chen, C. Chang, High-capacity reversible data hiding in encrypted images based on extended run-length coding and block-based MSB plane rearrangement, J. Vis. Commun. Image Represent. 58 (2019) 334-344.
[15] C. Qin, Z. He, X. Luo, J. Dong, Reversible data hiding in encrypted image with separable capability and high embedding capacity, Inf. Sci. 465 (2018) 285-304.
[16] P. Manirho, T. Ahmad, Information hiding scheme for digital images using difference expansion and modulus function, J. King Saud Univ. Comput. Inf. Sci. 31 (3) (2019) 325-347.
[17] A.A. Abdulla, Exploiting similarities between secret and cover images for improved embedding efficiency and security in digital steganography, Doctoral thesis, University of Buckingham, 2015.
[18] P. Feng, Y. Zhao, X. Zhang, M. Long, W. Pan, Reversible data hiding based on RS-BEMD coding and adaptive multi-segment left and right histogram shifting, Signal. Process. Image Commun. 81 (2020).
[19] R. Zhang, C. Lu, J. Liu, A high capacity reversible data scheme for encrypted covers based on histogram shifting, J. Inf. Secur. Appl. 47 (2019) 199-207.
[20] K.-C. Choi, C.-M. Pan, C.L. Chen, Application of a generalized difference expansion based reversible audio data hiding algorithm, Multimed. Tools Appl. 74 (6) (2019) 1961-1982.
[21] D. Wang, X. Zhang, C. Yu, Z. Tang, Reversible data hiding by using adaptive pixel value prediction and adaptive embedding bin selection, IEEE Signal Process. Lett. 26 (11) (2019) 1713-1717.
[22] Y. Yang, X. Xiao, X. Cai, W. Zhang, A secure and high visual-quality framework for medical images by contrast-enhancement reversible data hiding and homomorphic encryption, IEEE Access 7 (2019) 96900-96911.
[23] T. Ahmad, T.P. Figar, Enhancing the performance of audio data hiding method by smoothing interpolated samples, Int. J. Innov. Comput. Inf. Control 14 (3) (2018) 767-779.
[24] T. Ahmad, Y. Samudra, Reversible data hiding with segmented secrets and smoothed samples in various audio genres, J. Big Data 7 (2020) 80.
[25] T. Ahmad, M.H. Amirzal, W. Wibisono, R.M. Jiharie, Hiding data in audio files: a smoothing-based approach to improve the quality of the stego audio, Helion 6 (3) (2020) e03464.
[26] H. Yao, F. Mao, C. Qin, Z. Tang, Dual-JPEG-image reversible data hiding, Inf. Sci. 563 (2021) 130-149.
[27] Y. Fu, P. Kong, H. Yao, Z. Tang, C. Qin, Effective reversible data hiding in encrypted image with adaptive encoding strategy, Inf. Sci. 494 (2019) 21-36.
[28] M.B. Andra, T. Ahmad, T. Usagawa, Medical record protection with improved GRDE data hiding method on audio files, Erg. Lett. 25 (2) (2017) 112-124.
[29] A. Bobeica, I.G. Dragoi, I. Caciula, D. Colnuc, F. Alb, Y. Fang, Capacity control for prediction error expansion based audio reversible data hiding, in: Proc. of 22nd International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 2018, pp. 810-815.
[30] K.-H. Jung, K. Yoo, Steganographic method based on interpolation and LSB substitution of digital images, Multimed. Tools Appl. 74 (6) (2015) 2143-2155.
[31] V.I. Kustov, D.K. Pronsko, Modeling hidden data by combined LSB&DCT algorithm, in: Proc. of the 2017 XX IEEE International Conference on Soft Computing and Measurements (SCM), St. Petersburg, Russia, 2017.
[32] A.A. Abdulla, S.A. Jassim, H. Sellahewa, Secure steganography technique based onbitplane indexes, in: Proc. of IEEE International Symposium on Multimedia, Anaheim, USA, 2013.
[33] A.A. Abdulla, S.A. Jassim, H. Sellahewa, Efficient high-capacity steganography technique, in: Proc. of Mobile Multimedia/Image Processing, Security, and Applications, vol. 8755, 2013.
[34] A.M. Alataar, Reversible watermark using difference expansion of quads, in: IEEE Int. Conf. Acoust. Speech, and Signal Process, Montreal, Que., Canada, 2004, pp. 377-380.
[35] J. Wang, N. Mao, X. Chen, J. Ni, C. Wang, Y. Shi, Multiple histograms based reversible data hiding by using FCM clustering, Signal Process. 159 (2019) 193-203.
[36] M. Hussain, A.W.A. Wahab, Y.B. Idris, A.T.S. Ho, K.-H. Jung, Image steganography in spatial domain: a survey, Signal Process. Image Commun. 65 (2018) 46-66.
[37] M. Hussain, A.W.A. Wahab, A.T.S. Ho, N. Javed, K.-H. Jung, A data hiding scheme using parity-bit pixel value differentencing and improved righthost digit replacement, Signal Process. Image Commun. 50 (2017) 44-57.
[38] L. Xiong, D. Dong, Reversible data hiding in encrypted images with somewhat homomorphic encryption based on sorting block-level prediction-error expansion, J. Inf. Secur. Appl. 47 (2019) 78-85.
[39] X. Wu, T. Qiao, M. Xu, N. Zheng, Secure reversible data hiding in encrypted images based on adaptive prediction-error labeling, Signal Process. 169 (2021) 108200.
[40] R. Schmitz, Use of SHDM in commutative watermarking encryption, EURASIP J. Inf. Secur. (2021).
[41] Rahkmawati, S. Suwadi, W. Wirawan, Blind robust and self-embedding fragile image watermarking for image authentication and copyright protection with recovery capability, Int. J. Intell. Engg. Syst. 13 (5) (2020) 197-210.
[42] S. Xiang, Z. Li, Reversible audio data hiding algorithm using noncausal prediction of alterable orders, EURASIP J. Audio Speech Music Process. 2017 (1) (2017) 4.
[43] D. Renza, D.M. Ballisteros L., C. Lemos, Authenticity verification of audio signals based on fragile watermarking for audio forensics, Expert Syst. Appl. 91 (2018) 211-222.
[44] A. Gurbah, N. Al-Juaid, E. Khan, Counting-based secret sharing technique for multimedia applications, Multimed. Tools Appl. 78 (5) (2019) 5591–5619.
[45] S. Xiang, X. Luo, Reversible data hiding in homomorphic encrypted domain by mirroring ciphertext group, IEEE Trans. Circuits Syst. Video Technol. 28 (11) (2018) 3099-3110.
[46] L.K. Ketshabetswe, A.M. Zungeru, M. Mangwala, J.M. Chuma, B. Sigweni, Communication protocols for wireless sensor networks: a survey and comparison, Helion 5 (6) (2019).
[47] IRMAS: a dataset for instrument recognition in musical audio signals, accessed on August 2017. [Online]. Available: https://www.upsf.edu/web/mtg/irmas/.