ISSN 2278-3091
Volume 8, No.1.5, 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse5381.53019.pdf
https://doi.org/10.30534/ijatcse/2019/5381.52019

Reformat the File Uncompressed into Lossy Based on Audio Compression Method using Huffman Shift Coding Scheme

Tonny Hidayat[1], Mohd Hafiz Zakaria[2], Ahmad Naim Che Pee[3]
[1]Universitas AMIKOM Yogyakarta, Indonesia, tonny@amikom.ac.id
[2]Universiti Teknikal Malaysia Melaka, Malaysia, hafiz@utem.edu.my
[3] Universiti Teknikal Malaysia Melaka, Malaysia, naim@.edu.my

ABSTRACT
One form of sound is music that has become a requirement for people in many ways. The impact of technological development and the ability of processing units to process data can be felt in a product we call music, comparable to analog-produced music in the era before the 2000s and in the digital age, there is now a clear distinction of the quality and accuracy of the frequency instruments enjoyed by hearing. Increasing the quality of an Audio data will make the file size increase also applies to image or video data, so it needs a compression for storage problems and real-time access needs. Huffman algorithm can be used for file compression, to reformat from Uncompressed file to Lossy file can use scheme of Huffman Shift Coding algorithm. This research provides a novelty in changing the format of Uncompressed (WAV) to Lossy (Mp3) format to respond to a real-time storage and access issues. In this paper providing changes in the Rate / Distortion Control Process includes Scale Factor, Quantizer and Noiseless, this process will change every symbol that is owned by WAV audio data which is then reconstructed to Mp3 format. From the results of compression obtained differences with the standard Mp3 compression.

Key words : Lossy, Shift Coding, Huffman, Compression, audio, lossless.

1. INTRODUCTION

Satisfaction in listening to an audio with good quality and apparently is the entertainment needs for music lovers who have been accommodated by the development of technology in the current generation. Included in the audio field. For music lovers or quality control a sound, it becomes a satisfaction to enjoy the sound quality is detailed and precise. Many options can be used Lossless or Lossy audio format. One master format that is commonly used is WAV to maintain all data without being lost [1]. In general terms for digital music lovers are still more likely to use Mp3 format (MPEG-1 Audio Layer 3). Starting from the 90s Mp3 format began to be known and used because the scheme and audio results can be captured quite satisfactory for music lovers but not recommended for the needs of mastering, archive, and developers. The comparison of file sizes of these two file types is very different, WAV has a larger file size, whereas Mp3 is much smaller to 80% of the source because it is Lossy so a lot of data is omitted and cannot be recovered into the original data. If CD audio quality data use a sample rate of 44.1 kHz, 16-bit depth, stereo (2 channels) with a duration of 100 seconds. In a simple calculation can be calculated the amount of data samples generated byte count is then the total storage of audio data per second is 44100 * 100 * 16/8 * 2 = 17.640.000 bytes so for the duration of 60 seconds (1 minute) required 10,584 MB [2][3]. So it can be predictive if the music with a length of 3 minutes, with the same sample rate conditions as above, will produce 30 MB file size in storage in storage, this will certainly spend a lot of space on the hard disk storage [4].

Compression is needed to address the issue of storage issues with specific schemes, methods, and algorithms [5]. There are many algorithms that can be used in terms of compression one Huffman shift coding algorithm that can be used for the compression method Lossy. The Huffman algorithm works by encoding in bits to represent character data. As in some literature that the Huffman Algorithm is not so maximized if there are many variations of symbols on a data [6]. To condition the Huffman algorithm to convert the Uncompressed format into Lossy format using the compression scheme of the Huffman Shift Coding algorithm that will group the symbol from the original data or the source into multiple blocks. Changes made in this study are in the Formation process, specifically the Rate / Distortion Control Process to achieve a higher ratio with different conditions for WAV file specifications, so that compression can be done at various sample rates.

2. THEORETICAL ANALYSIS

A. Digital Audio

Sound or Music in the digital world now is an analog wave that is now beginning to be abandoned but still maintained in some mastering purposes. The waves generated from the vibrations that propagate from the air pressure that is around, then the sense of hearing catch it with the help of eardrum which then we think is sound. The process of digitizing from

analog is to convert each wave sample into numbers which can then be processed and generated by the computer [7][8]. The change of audio data from analog to digital is done by measuring a wave across multiple nodes in the unit time range, computing each wave node into a numerical form and then writing the numbers to a data into a file (sampling)[9]. The advantage gained from digital audio rather than analog is the perfect sound reproduction quality. Perfection in the improvement or addition of an instrument and sampling will result in the ability to double the audio signal simultaneously without decreasing the sensitivity of the resulting sample rate. One of the advantages or other benefits that can be achieved through digital audio is the resistance to the signal that should not be stored (noise)[10]. Analog signal is elementary to experience noise (noise) in the event of transmission, the form of noise that will sound very often found when playing with a cassette recording with electrical components [5].

B. The Format WAV

As it is already known that the multimedia application needs a mechanism in the data storage format such as image, audio and video. RIFF is a means to store any type of data, while audio WAV format is part of the Microsoft RIFF is used to store digital audio data [11]. This file format is one of the audio file formats on the PC. Along with the popularity of Windows then many applications that support this file format. Due to its simple structure, many developers develop different specifications and standards to maintain a sound quality or to convert it into other formats[12].

WAVE files use standard RIFF structures that group file contents (sample formats, digital audio samples, etc.) into separate "chunks," each having its own header and byte of data. The chunk header specifies the type and size of the chunk data byte. With this method of setting up a program that does not recognize a special chunk type can easily pass through this chunk section and continue the process of processing the known chunk. Certain chunk types may consist of sub-chunk. For example, in Figure 1 can be seen chunk "fmt" and "data" is actually a sub-chunk of chunk "RIFF"[13][14].

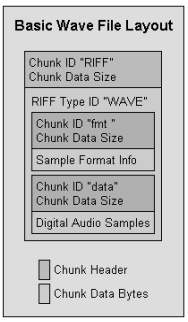


Figure 1: Scheme WAV Layout

Chunk in the RIFF file is a string that must be set for each word. This means the total size of the chunk must be a multiple of 2 bytes (like 2, 4, 6, 8 and so on). If a chunk consists of an odd number of bytes, then a byte (extra padding byte) must be added by adding a value of zero to the last byte of data. This extra padding byte is not counted on the chunk size. Therefore a program must always make word settings to determine the size of the header value of a chunk to calculate the offset of the next chunk[3].

C. Format MP3

Figure 2: Description of the Data header frame

MP3 is a format that is interesting because it could retain the sound quality while having a size not too large. The technology was developed by the Fraunhofer Institute engineer in Germany, Karlheinz Brandenburg. MP3 is composed of lots of frames, where each frame contains some seconds of audio data which is useful, which is ready constructed by the decoder. The initial part of each frame of data is the "frame header", which contains 32 bits of meta-data that is associated with the incoming data frames Though MP3 looks so good, it turns out that this format also has its limitations of its own [15]:

a. Bitrate is limited, maximum of 320 kbit/s (some encoder can produce a higher bit rate, but very little support for mp3-mp3 which have high bit rate.
b. Resolution time used mp3 can be too low for voice signals are highly transient, so that can cause noise [16].
c. Frequency Resolution is limited by the size of the window that the small length, reducing the efficiency of coding.
d. There is no scale factor band for frequencies above 15.5 or 15.8 kHz [16][17].
e. A joint stereo mode is done on a per frame basis.
f. Delay for the encoder/decoder is not defined, so there is no encouragement for gapless playback (audio playback without gap)[16]. However, some encoder like LAME can add additional metadata that provides information to an MP3 player to overcome it.

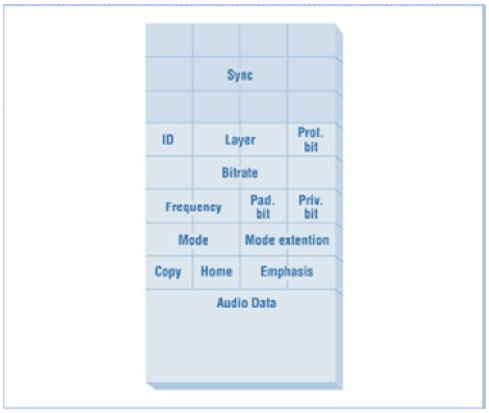


Figure 3: Visually MP3 Frame Header

Table 1:. Characteristics Of The Header Files

The position of the	The purpose of	Bit
A	Frame synchronization	11
B	MPEG Version	2
C	MPEG Layer	2
D	Protection	1
E	Bitrate index	4
F	Sample rate frequency	2
G	Bit padding	1
H	Private bit	1
I	Fashion channel	2
J	Advanced mode	2
K	Copyright	1
L	Originality	1
M	Emphasis	2

D. General Huffman Code

The main thing about the understanding of the process i.e. compression coding data symbols use the bit then merged or deleted. There are two types of compression: lossless and Lossy. In lossless compression, first data will be broken down into smaller sizes and data eventually reunified. Whereas, in Lossy compression, there are bits of information that are eliminated after compression. This type of compression is often done for the compression of pictures. The general principle in the process of compression is reducing duplication of data so as to represent the memory becomes less than the original digital data representation. Some comparisons between Lossy and lossless:

The advantage of lossy methods over lossless is in several cases a Lossy method can produce smaller compressed file compared to the lossless method, while still meeting the requirements of the application.

Lossy methods are often used for dynamically compressing the sound, pictures, and video. Because the data is intended to be human readable interpretation in order for it to be in formulation, ideally Lossy compression is transparent, which can be verified with tests ABX. While lossless is used for dynamically compressing data to be accepted in original condition are geared as a text document.

Lossy will experience generation loss on the data while in lossless does not occur because the data results of decompression are equal to the original data.

Huffman code is one method of data compression that was created by David a. Huffman. Huffman code using specific techniques to represent each of the symbols in which the expression appears most often gets the size of the smallest bits and expression that rarely comes up gets a bit more size [18] [19]. The process of formation of Huffman code is:

a. Select 2 symbol with the most opportunities. Both symbols of yesteryear combined as the parent node and summed up his chances. This new symbol treated as new nodes and taken into account in the search for the next symbol that has a very small chance.
b. Next, select the two symbols, including a new symbol, which has the smallest chance. Do the same thing as the previous step.
c. Repeat until the entire writings of encoding.

For example, a large number of tables will be given the appearance of a "ABBACADABA":

Table 2: The Emergence Of Writing Table

Symbol	Frequency	Opportunities
A	5	3/10
B	3	1/10
C	1	2/10
D	1	1/10

At the time performed the steps above like Huffman coding, then the result is:

A code: 0
B code: 10
C code: 110
D code: 111

3. METHODOLOGY

Compression is the conversion of data into forms that require fewer bits, and it is usually done so that the data can be stored or transmitted more efficiently. The reverse process, namely, compression and decompression. Decompression is itself a process to return the new data that has been generated by the process of compression be preliminary data [20], [21]. Data compression has done can be used a lower chain that corresponds to the capacity of the data compression process has been performed.

Compression technique consists of 3 categories, among others: the source, the entropy, and hybrid. For source categories, namely the type of compression audio lossy, occur some part component of lost data resulting from the process of compression [22]. For the group of entropy is a type of lossless audio, which means that no data is lost during compression process is running. As for the category of hybrid, a combination of the kinds of lossless and lossy audio.

A. Algorithm Of Huffman Coding Shift

In the Huffman Shift Coding scheme, all the resulting symbols from the audio data generated will be grouped into blocks with the condition that each block gets the same proportion. General size blocks are generally $2k - 1$ symbols, that k is a positive integer. This is done to calculate the audio data because If $k = 1$ the result will be the same as Huffman Shift Coding Standard which is likely to result in the schema of this method remains weak against many variations of the resulting data symbol[23].

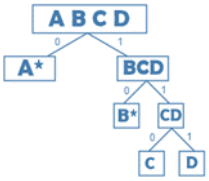
Starting on the first block, the symbol is encoded with the standard Huffman Coding. At the time of the first block in the

encoding process, performed also the coding symbol hypothesis with a condition that is the frequency number of appearances equal the sum of the frequencies of symbols from other blocks, referring to the fundamental principle, i.e., the proportion of each block must be the same. That becomes the Differentiator between blocks to the next block is the addition of a prefix, then the result of encoding symbols hypotheses obtained will be used to identify any blocks as a differentiator. The primary objective is achieved can increase processing time less with the ratio of the average code length is optimum, with Huffman compression algorithm Shift scheme as follows:

1. Source Symbol is composed of possible symbols that appear from the largest to the smallest.
2. The accumulation of all the number of symbols derived from the source will be grouped into blocks by the number of servings of the same symbols as much. The blocks are in order in the first encoded using Huffman Coding Standard. When the process of encoding symbols on the first block took place, accompanied the process encoding the encode hypothetical occurrence frequency is equal to the number of occurrences of a symbol from another block. Distinguishing between the blocks with the block next is + 1 or > = 1 prefix codes generated calculations of symbols coding hypothesis as a marker per block.
3. Symbol C as a representation of the hypothetical symbol of Huffman Coding.
4. The I-symbol code of block k is $C\ k$-1 coupled with the I-th symbol of the first Huffman Coding block.

Changes can be seen when shaping it, Huffman tree is at the moment in value right as shown in the following example:

Value :
A = 0,4
B = 0,3
C = 0,2
D = 0,1



B. FFMPEG

Another thing of concern in compressing and changing the format of an audio data (lossless to Lossy) is when compress results from the compression process can be replayed, the necessity of adding a plugin function codec such as FFMPEG as the computer program to record, and then convert and save it like mind my extensions compatible format. Fabrice Bellard who first started this formula which then manage by Michael Niedermayer a line program or set of procedures to module file format for digital audio and video [7].

Role in the process of formation of the FFMPEG plugins that is instrumental in the process of compression of audio data which results of decompressing audio data can still Play and be heard again. As it is already known that the standards of the flow layer 3 is owned by the Mp3 format that is Each byte, bit and metadata will be modified with FFMPEG which supports the layer.

C. Process Scheme

1. Fundamental

In general, there are two Lossy and Lossless compression methods, it is also impact on algorithms that work is also divided for two purposes. Terms and facts from lossy compression is a compression results cannot be restored as the original due to data being removed according perceptual procedure is needed. While lossless compression has a different principle, i.e., the data is in compression, after Decompression will be the same as the original data, which means that no data is lost after the process of compression and data can be restored.

The main purpose of these schemes this process is changing a little bit the procedure of Huffman Coding in Shift do audio data compression with the achievement of the desired end is answering the problem of storage media to reduce the size data, and other things are in terms of facilitating access in transferring real time or via the internet. The data will be processed by a special compression such as output compression MP3 (lossy) with provision 2 channel (mono and stereo) as the limit. Audio files are generally the same as the other types of digital data that is a representation of the number of bits ' 0 ' and ' 1 '. This provision is always done as a standard on all methods of compression that is processing a recurring data compaction.

2. Standard Audio Compression Rate

Huffman encoding algorithm or is actually a compression algorithm that can be applied to any type of good for a binary file or a text file. The algorithm is useful with a low compression ratio if there is a lot of redundancy data or data on the same looping audio file that is used [24].

The WAV format file will be very large in size as long as the duration. Of audio data that are already collected by type of 48,000 kHz sample rate with the number of the channel 2 or stereo and bits per sample of his 24 bits for the duration for 1 second only requires a capacity of 48,000 x 2 x 24 = 2,304,000 bits per second = 288,000 bytes per second. If we take the average duration of the audio data that has a duration of about 4 minutes, then storage media required is 288,000 x 4 x 60 = 69,120,000 byte or the equivalent of 69 megabytes. When we have 500 megabytes of storage media, then the disk space can be calculated once saved one audio data WAV format is = 500-69 = 431 megabytes [25]–[29]. Then if we have a storage media that you want to transfer audio data WAV, then only can accommodate approximately 7 files only.

3. Compression Algorithm and Implementation

provisions in performing audio compression process, there are several methods that can be used, among others: psychoacoustic model, auditory masking, critical band, and joint stereo. In performing MP3 compression, the method is a psychoacoustic model. The standard that has been experimenting on psychoacoustic is MPEG Layer-1, followed

by MPEG Layer-2 [30]. In this model, the first one is to drastically reduce the level of data required to regenerate the sound equivalent of CD quality. MPEG Layer-3 further reduces it (Layer-1: 384 kbps, Layer-2: 192 kbps, Layer-3: 112 kbps). Difference between MPEG Layer 1,2 and 3 regarding design:

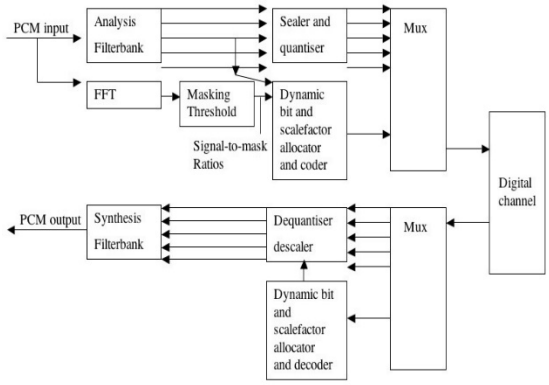


Figure 4: MPEG Layer 1 and 2

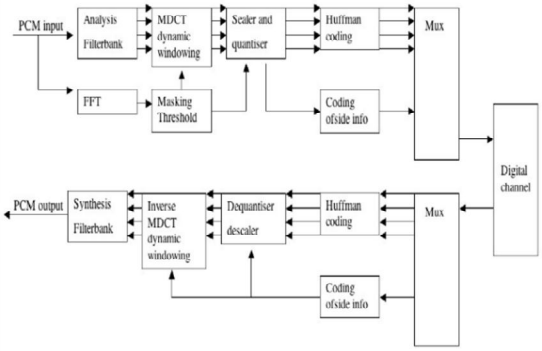


Figure 5: MPEG Layer 3

The process of formation of the MP3:

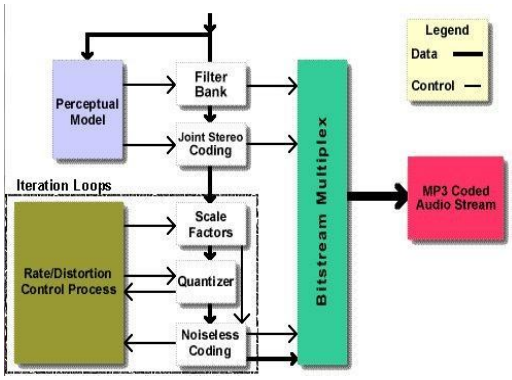


Figure 6: The Process Of Formation Of MP3 Files

The first block seen is the Bank-Filter block. the solution used by Fraunhofer to combine a bank filter polyphase and a Modified Discrete Cosine Transform (MDCT) is to use a hybrid solution for MP3. Bank-filter polyphase is the initial separation of audio streams into sub-band frequencies spaced equally [31]. With this sample, core frequencies are selected and then used for compression. One problem that arises from this election is the core frequency overlapping with the subband, which causes quality problems and voice clarity. To respond to this problem, then on the 32 sub-band the unambiguous MDCT filter bank. In order for sub-band made up-down with more then do expansion at any distance. To resolve the error on MDCT, inverse functions combined with new samples in the process via the MDCT, this activity is called domain aliasing cancellation time. In this process simply takes time, but the calculations and reduced the time it takes using a recursive factorization[31], [24]. The audio data is made from samples that are then passed to the encoder joint stereo coding, in the process of this is information overload and don't matter is removed from the flow of bits.

To count as the boundary of the frequency of perceptual models are either in the determination of the critical frequency. Other functions as well as to determine the initial masking on each sub band. the term masking is when there is a voice that barely covered or overwritten with other sounds so doesn't sound. Based on the model psychoacoustic, the algorithms working on an encoded can create a rule to eliminate or reduce the number of bits allocated for each segment of a sound level which is under at the start of masking for each sub-band [25]. Another factor that determines whether that can be omitted is the bit rate of the encoding. The lower bit rate, the more data is eliminated.

The entire data stream is divided into several parts, as well as sub-brands then passed to a pair of loops, namely: a. the control loop (control of noise/distortion loop) Examine each section to determine the noise in each sub-band which is outside sound level or masking limits allowed. If sub-band sound is permitted, exceeded the scale factor for the band to be adjusted. Scale factor gives an advantage gained for each sub band. The gain is the ratio of output and input signal to change provisions of the bit rate is increased then the loop rate is initialized.

Huffman code is a scheme for analyzing data sets and generates a set of bits to represent the data. Rate loop checks on code Huffman. condition that occurs is the more frequently used input sample is then given a bit string that is smaller, so do lossless compression Lossy data from the psychoacoustic encoding [21][33], and encoding different Huffman tables used for the spectrum of different frequencies to get maximum results for size reduction. If the number of Huffman code bit higher bitrate is allowed, then the number of pieces of the time will be reduced [34][35][36][37].

4. RESULT AND DISCUSSION

A. The Role of Huffman Code

The encoded data consists of 576 rows of frequencies per channel and a small section stored as 16 bits of a signed integer. For example, it will give a picture of the frequency spectrum in MPEG file Layer 3:

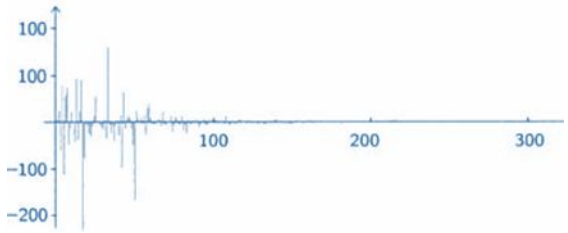


Figure 7: The frequency of the layer 3 quantized

4, 24, −63, 79, −114, 55, 75,
−51, −13, 21, −1, −43, 94, −41, 22, 92, −231, −77, −4, −7, −26, −31, −11, 12, 52,
0, 2, −6, −12, −18, 6, −36, 159, −6, −25, −6, −43, −6, 3, −29, 14, −99, 64, −15, 7,
14, 9, −15, −52, −168, 25, 6,
1, 10, 3, 15, −28, 30, 38, 8, −3, −6, −5, 2, 5, 4, −9, 24, −6, −6, −1, 2, 13, −22,
−8, 12, 5, −5, −6, 12, −20, −6, −24, 7, −5, −5, 3, 0, 5, 3, −11, −5, −1, 6, −7, −6,
5, −2, 3, 5, 4, 2, 2, −2, 1, 3, −5, 14, 0, −8, −5, 0, 1, −4, 2, 3, 5, −1, 1, −1, 2, 0, 4,
−2, 1, 0, 0, −3, −3, −1, 0, −6, −5, −2, −1, 0, −1, −1, 3, 8, 2, 0, 2, 1, 1, 1, 0, 2, −4,
−3, −1, 2, 2, −1, −1, 0, 0, 0, 1, 1, 2, 2,
0, −2, 1, 0, −1, 2, 1, −1, 0, −1, 0, −1, −2, −1, 0, 0, 1, 1, −2, −2, 0, 0, 0, −1, 0, 0,
0, 1, −1, −1, 0, 0, 1, 0, 1, 1, −1, −1, 1, −1, 0, 2, 0, 0, −1, 0, 1, 2, 1, 2, −1, 2, 2, 3,
−1, 2, 2, 0, 0, −1, 1, 1, 1, 0, −2, 1, 0, 2, 1, 0, 0, −1, 1, 0, −1, 1, 0, −1, 0, 0, 1, 0, −1,
−1, 1, 1, 0, 1, 0, −1, 0, 1, 1, −1, 0, 0, 0, −1, −1, −2,
−1, 0, 0, −1, 0, 0, 0, 0, 0, 0, 0, 1, −1, 0, 0, 0, 0, 0, −1, 1, −1, 0, 0, −1, 0, 1, 1, 0,
0, 1, 0, −1, 1, 0, 1, 0, 0, −1, 1, 0, 0, 1, −1, −1, 0, 0, 0, 1, 1, 0, 0, 0, −1, 0, 0, 0, −1,
1, 0, 0, 1, −1, −1, 1, 0, 0, −1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, −1, 1, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, −1, 0, 0, 0, 0, 0, 0, 0, 1, 1, −1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, −1, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
0, 0, 0, 0, −1, 0, −1, 0, 0, −1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, −1,
0, 0, 0, 0, 0, 0, 0, −1, 0, 1, 0, 0, 0, −1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, −1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, −1, 0.
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Please note the numerical value of 576:

It can be seen from the top end of the spectrum, and all zero values preceded by a lot of "little value". Part of the spectrum of left zero value, where the spectrum is completed by the start of values 1 and-1 is part "of little value". This value can not hear this sound, so the encoder eliminates typically it.

B. Do the Decoding on Small Value

To do the decoding on small value, assign the value to 4 (u0, u1, u2, u3) and do encode on the group. Actually, just the absolute value, 0 or 1, which encodes, and code for groups followed until 4 sign bit. For any non-zero value in the group have 1 sign bit. If the bit is 0, the value of it positive, if 1, the value is negative.

The encoder can choose 2 different table to encode the value of the small, htabA and htabB. htabA is a table are indexed by byte value of 6 bits. Any byte contained the absolute value of (u0, u1, u2, u3) on the left and the actual value of Huffman is used on the right. Suppose there is a variable named huffman_cache, which contains huffman_cache_size bit left-align. Here we assume that we fill huffman_cache that ensure there are at least 16 bits are valid from huff_cache.

code = htabA [((unsigned int) huff_cache) >> (HUFF_CACHE_SIZE-6)];
huff_cache = huff_cache << (code & #F);
huff_cache_size = huff_cache_size-(code & #F);

After we load code from htabA, 4 bit far right contains the necessary bits. They are needed to set up the huff_cache. The value of the variable codes need to be arranged with the appropriate sign before put into the array of raw frequency u. To store a value into the correct position, we put a pointer to u. u To the value that's already coded zero, Huffman code followed by the bit marked. Bit-1-1, value of the marked and marked with zero bits + 1. Rather than test each bit separately, we combine a bit of huff_cache and 4 bits of code to 1 8-bit value that we use as an index into a table named signed_small_values that contains the value of the u0, u1, u2 and u3 and n the number of bits used

Code = (code & #F0) | (((unsigned int) huff_cache >> (HUFF_CACHE_SIZE-4));
* u + = signed_small_values + [code] u0;
*u++ = signed_small_values[code].u1;
*u++ = signed_small_values[code].u2;
*u++ = signed_small_values[code].u3;

C. Do Decoding at a great value

The first part of the frequency spectrum contains "great value". Here the values can be developed into 213 = 8192 and put it on a table of Huffman code most of which are never used. To be more effective, the encoding scheme used for the "great value", i.e., a value from 0 to 15, using a table of Huffman code, and was followed in the flow of bits by unsigned, added to the 15 for a very large value. To find out the number of bits used in coding the unsigned, each table there is a value named "linbit" most 13. It delivers maximum value to a large value of 213 + 15 = 8207. 0-15 values can be encoded with 4-bit code because Huffman has a length of at least 1 bit, it gives the maximum compression value of 25%. For Improvisation, the great value in encoding in the pair (u0, u1) which makes the value of compression can be 12.5%.

If the htab is a Huffman table, this is a table of short integers are indexed by bit HWIDTH from huff_cache. Each integer there is a short in the left-most 8 bit, 4bit next value u0 and u1 far right. Huffman tables use the extension because, we cannot conclude that each short int in the table have shapes as you describe, it could be a pointer. In this case, he has a negative value, and 12 bit far left gives the difference between the initial index and table extension. The difference is negative, because the extension table is the primary table after 4 bit number giving the rest of the index used to access the table extension. How to read two great value with the htab:

int width = HWIDTH; short int * h = htab; code = h [((unsigned int) huff_cache) >> (HUFF_CACHE_SIZE-width)];
While (code < 0)
{huff_cache = huff_cache << = huff_cache_size huff_cache_size; width-width; h = h-(code >> 4); width =

code & #0F; code = h [((unsigned int) huff_cache) >>
(HUFF_CACHE_SIZE-width)];
}

After you get the code right, we change the code mask and huff_cache to obtain the value u0 and u1.

huff_cache = huff_cache << (code >> 8);
huff_cache_size = huff_cache_size-(code >>
8);
code = code & #FF;

To read the bit marked, we simply test the leftmost bits of the huff_cache, which is the sign bit, and test it like: if (huff_cache < 0) ... else .... But, in fact, more use of the branch as it takes a rather long time. Determine the sign of the value

{int tmp = ((signed int) huff_cache) >>
(HUFF_CACHE_SIZE-1);
value = (value Å tmp) the tmp; huff_cache huff_cache << = 1;
huff_cache_size--;
}

The above code uses the property of the value represented in the binary format of 2 's complement. Change huff_cache from the far left to channel produce an integer value which is all bit is set to one if the far left 1, or 0 if the far left 0. We save the result to a tmp and use 2 times: as a mask from operations decrease it, and Å. In the case of sign bits already set, this value is the complement of a binary operation and an addition of 1, which is against the definition of 2's complement representation.

D. Quantization and Scaling

On a short window cases and in one frame there are 576 MDCT coefficients, three consecutive group of 192 coefficient accumulated to form a single frame representing the MDCT coefficients 576 in the form of a vector comprising the element *xrm*: where each element assumes real values ranging from -1.0 to 1.0..

$\mathbf{xr} = [xr0 \cdot \cdot \cdot xrm \cdot \cdot \cdot xr575]T$

When executing, the component in vector quantization **xr**, grouped into 22 scale factor band, depicted on the figure 8, i.e. There is a horizontal axis on the interval for 22, and each interval corresponds to one band scale factor, $\mathbf{xr}^T\{s\}$. then there was the following conditions..

$\mathbf{xr} = [\mathbf{xr}^T\{0\}...\mathbf{xr}^T\{s\}...\mathbf{xr}^T\{21\}]^T$

Each band has its own quantization factor, then the variation of scale Division of each pita factor based on the frequency of sampling ... If *XR m* the property of the band s, then quantize becomes as follows:

$ix_m = Q(xr_m, Z_{\{s\}}) = sign(xr_m) \cdot [(2^q |xrm|)^{3/4}]$

where ix_m MDCT coefficients are quantized integer of the value and its mark is equal to *XR m*, $Z_{\{s\}}$ is a value scale factor relating to the band s, q is a positive integer which is the quantization factor which corresponds to the $Z_{\{s\}}$, and [.] means rounding to the nearest integer.

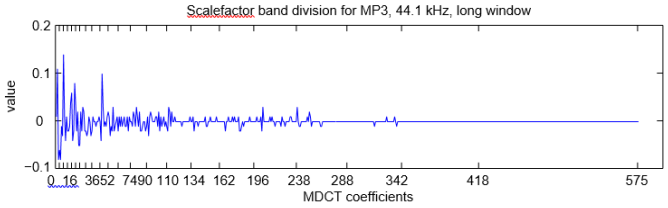


Figure 8: Scale-factor band division for MP3, 44.1 kHz, long window

E. Working Mechanism Quantization Factor and Bit Rate

The iteration loop system there are two nested to quantize to find optimal quantization factor *q*. The larger the number of bits of the target *TB* (shown at targ_bits in Figure 9), the more iterations to be performed, which will lead to greater quantization factor.

```
/* inner_loop starts with the initial quantization step computed above
 * and slowly increases until the bits < huff_bits.
 * Thus it is important not to start with too large of an inital
 * quantization step.  Too small is ok, but inner_loop will take longer
 */
huff_bits = targ_bits - cod_info_w.part2_length;
if (huff_bits <= 0)
    break;

/*  increase quantizer stepsize until needed bits are below maximum
 */
while ((cod_info_w.part2_3_length
    = count_bits(gfc, xrpow, &cod_info_w, &prev_noise)) > huff_bits
       && cod_info_w.global_gain <= maxggain)
    cod_info_w.global_gain++;
```

Figure 9. The inner iteration loop of the MP3 encoder (simplified) from LAME v3.97

In the condition of MP3 signal transcoding from a bit rate of *BR* 1 to bit rate *BR* 2 , where *BR* 1 <*BR* 2 . The number of target bits corresponding to one frame each is denoted as *TB* 1 and *TB* 2 . Because one frame is divided into 22 band factor scales in quantization (see Figure 8), and each factor scale has its own quantization factor , we represent the quantization factor corresponding to $q_{1\{s\}}$ and $q_{2\{s\}}$ $(0 \le s \le 21)$. Because the bit rate means the total bits per second, we have *BR* 1 < *BR*2 => *TB* 1 < *TB* 2 , under the assumption of a constant bit rate (CBR) mode. When transcoding from *BR* 1 to *BR* 2 , for each factor-scale band in one frame it is naturally expected to have $q_{1\{s\}} < q_{2\{s\}}$. However, if for all $q_{1\{s\}} < q_{2\{s\}}$, the number of target bits for TB_2 can sometimes not be achieved correctly. Then for some *s* $(0 \le s \le 21)$, encoder allows $q_{1\{s\}} = q_{2\{s\}}$. So when $BR_1 < BR_2$, we have $q_{1\{s\}} \le q_{2\{s\}}$ for all s. On the other hand, if if $q_{1\{s\}} = q_{2\{s\}}$ for all s, there is no transcoding. Therefore we have $0 \le P\{ q_{1\{s\}} = q_{2\{s\}} \} < 1$ in each MP3 frame transcribed.

F. Probability Values

The determinant for Lossy quality that contains less small values is that of the Lossy format compression can be done twice and the second bit rate of BR2 is greater than the first bit rate of BR1. Because the MDCT coefficients quantized to normal MP3 single compressed while it

goes doubly compressed, in this case the states respectively as *ix S ix D* with. Now it will You can use *IX S* with *IX D* , where the double compression quantization factor is $q1$ and $q2$, respectively. As a limitation, analysis will be carried out on the condition of the MDCT *xr* coefficient is non-negative in the following because the analysis is the same when *xr is* negative.

Double compression consists of three steps. First, the MDCT coefficient is quantized as $Qq1\ (xr) = [(2^{q}1 \cdot xr)^{3/4}]$ with the first quantization factor $q1$; second, the coefficient value become $Q_{q1}{}^{-1}(Q_{q1}(xr)) = [(2^{q}1 \cdot xr)^{3/4}]^{3/4}. 2^{-q}1$ when quantized with $q1$; third, the coefficient value becomes $[[(2^{q}1 \cdot xr)^{3/4}]^{3/4}. 2^{3/4}(^{q}2-^{q}1)]$ during the second quantization by factor $q2$. It can be said that the MDCT coefficient for Lossy Mp3 phases is compressed multiple, and can be stated as follows..

$ixD = Qq2\ (Q^{-}{}_{q1}{}^{1}(Qq1\ (xr))) = [(2^{q}2 \cdot [(2^{q1} \cdot xr)\ 3/4\ ]\ 3/4 \cdot 2^{-q}1\ )\ 3/4\ ]$

$=[(2^{q1} \cdot xr)\ 3/4\ ] .\ 2\ ¾\ (^{q}2-^{q}1)]$

For a meaningful comparison, we use identical quantization step $q2$ for a single compressed coefficient, *ixS*. Is.

$ixS = Qq2\ (xr) = [(2^{q2} \cdot xr)\ 3/4\ ]$

It can be noted that $PD = P\ \{ixD = 1\}$, $PS = P\ \{ixS = 1\}$ the next one is made

$PD < PS$

$PS = P\ \{ixS = 1\} = P\ \{[(2^{q2} \cdot xr)\ 3/4] = 1\}$

$== P\ \{0.5 \leq (2^{q2} \cdot xr)\ 3/4 < 1.5\}$

Because $BR1 < BR2$ applies to MP3s with Lossy quality , from the conclusions of the previous subsection we know that $q1$ dan $q2$ is a positive integer, $q1 \leq q2$ for all factor-scale bands, that is, actually for each MDCT coefficient,, $P\ \{\ q1 = q2\ \} <1$ for each frame. Therefore, it is clear that $2\ 3/4\ (^{q}1-^{2}1) \geq 1$. The next time it can be shown

$PD = P\ \{ixD = 1\}$

$= P\ \{[[(2q1\ \cdot xr)3/4\ ] \cdot 2\ 3/4\ (q2 - q1\ )\ ] = 1\}$

$= P\ \{0.5 \leq [(2q1 \cdot xr)\ 3/4\ ] \cdot 2\ 3/4\ (q2 - q1\ ) < 1.5\}$

$= P\ \{[(2q1\ \cdot xr)\ 3/4\ ] = 1 \cap 1 \leq 2\ 3/4\ (q2 - q1\ ) < 1.5\}$

$= P\ \{0.5 \leq (2q1 \cdot xr)\ 3/4 < 1.5 \cap 0 \leq q2 - q1\ < 0.80\}$

$= P\ \{0.5 \leq (2q1 \cdot xr)\ 3/4 < 1.5 \cap q2\ = q1\ \}$

$= P\ \{0.5 \leq (2q\ 2 \cdot xr)\ 3/4 < 1.5 \cap q2\ = q1\ \}$

$= PS \cdot P\ \{q2\ = q1\ \}$

$< PS$

In the example above, we need to note that the event is $0 \leq q2 - q1\ < 0.80$ is equivalent to the occurrence of $q2\ = q1$, because $q1$ and $q2$ both are non-negative integers. Because $0.5 \leq (2q2 \cdot xr)\ < 1.5$ describes events from a single compression case, while $q2\ = q1$ describes the events of a multiple compression case, these two events are independent. Therefore, we have proven that $PD < PS$ for the MDCT +1 coefficient value. In the same way, we can prove that $P\ \{ixD = -1\} = P\ \{ixS = -1\} \cdot P\ \{q2\ = q1\ \}$ for value −1.

G. Test Result

The scheme has been designed the process undertaken, and the results of testing the audio file compressed with Huffman Coding Algorithm Shift can be seen in table 4 and table 5, with testing done on some file types WAV format obtained from data collection on http://www.music.helsinki.fi"J.S. Bach; The E major Partita, Gavotte en rondeau (excerpt)-Sirkka Väisänen, violin "File which is in the process of having the size and type of complexity varies as the following table 3.

The checking is done by the application that provides the numerical data of the resulting compression ratio results.

Table 3: File WAV specification

File Name	Detail
a2002011001-e02.wav	Original recording (PCM encoded 16 bits per sample, sampling rate 44100 Hertz, stereo)
a2002011001-e02-16kHz.wav a2002011001-e02-8kHz.wav	PCM encoded sound files with 16 kHz and 8 kHz sampling rate
a2002011001-e02-ulaw.wav	U-LAW encoded (8 bits per sample) sound file with 44.1 kHz

Table 4: Results WAV file compression process to Mp3

Parameter	a2002011001-e02.wav	a2002011001-e02-16kHz.wav	a2002011001-e02-8kHz.wav	a2002011001-e02-ulaw.wav
Original Size (KB	9,357	3,395	1,698	9,357
Compression Size Mp3/KB	2,465	1,711	1,072	1,289
Ratio Compression %	26,34	50,40	63,13	13,78
Compression Speed Second	9	4	3	12

* In the U-law type audio file to WAV, it cannot be processed and analyzed correctly because the original size is 4,679 KB.

Table 5: Comparison of Compression Process repeatedly

Audio File WAV	Compression to Mp3		
	Once	Twice	Three Times
a2002011001-e02.wav	2,465	2,465	2,465
a2002011001-e02-16kHz.wav	1,711	1,711	1,711
a2002011001-e02-8kHz.wav	1,072	1,072	1,072
a2002011001-e02-ulaw.wav	1,289	1,289	1,289

5. CONCLUSION

Based on the research conducted on compression, it can be seen that the process rate/distortion control process can be done in the way format specified by 16-bit samples, but there are differences in the ratios obtained at each sample rate and unsuccessful on the U-Law type. The most optimal compression ratio is found in the kind of WAV format 16bit / 44.1 kHz of 26.34%, and some fragments have been obtained

is 20% from standard Mp3 compression. Results Process speed depends not only on the complexity of the recorded audio data but also depends on the bit sample and sampling rate. Doing repeated compression does not affect the results because the schematic process is static, so the compression process is done several times with the same data still getting the same results.

ACKNOWLEDGEMENT

This paper is an ongoing research on more in-depth lossless compression technique for music files. Thank you Universitas AMIKOM Yogyakarta and Universiti Teknikal Malaysia Melaka, and everyone who provided substantial support in this study.

REFERENCES

[1] B. Singh, A. Kaur, and J. Singh, "A Review of ECG Data Compression Techniques," *Int. J. Comput. Appl.*, vol. 116, no. 11, pp. 39–44, 2015. https://doi.org/10.5120/20384-2644

[2] I. R. E. Oyarzabai, "(12) Unlted States Patent (10) Patent N0 .: Primary Examiner * Srirama Channavajjala," vol. 2, no. 12, 2013.

[3] R. R. Devi and D. Pugazhenthi, "Ideal Sampling Rate to Reduce Distortion in Audio Steganography," *Procedia Comput. Sci.*, vol. 85, no. Cms, pp. 418–424, 2016.

[4] S. Hicsonmez, E. Uzun, and H. T. Sencar, "Methods for identifying traces of compression in audio," *2013 1st Int. Conf. Commun. Signal Process. Their Appl. ICCSPA 2013*, no. May, 2013. https://doi.org/10.1109/ICCSPA.2013.6487284

[5] M. Hans and R. W. Schafer, "Lossless compression of digital audio," *IEEE Signal Process. Mag.*, vol. 18, no. 4, pp. 21–32, 2001. https://doi.org/10.1109/79.939834

[6] G. S. Sandeep, B. S. S. Kumar, and D. J. Deepak, "An efficient lossless compression using double Huffman minimum variance encoding technique," *Proc. 2015 Int. Conf. Appl. Theor. Comput. Commun. Technol. iCATccT 2015*, no. C, pp. 534–537, 2016.

[7] R. Yu, X. Lin, S. Rahardja, and H. Huang, "MPEG-4 Scalable to Lossless audio coding - Emerging international standard for digital audio compression," *2005 IEEE 7th Work. Multimed. Signal Process.*, pp. 2–5, 2006. https://doi.org/10.1109/MMSP.2005.248562

[8] S. Sukode, I. J. Volume, S. Sukode, P. S. Gite, and H. Agrawal, "International Journal of Advanced Trends in Computer Science and Engineering Available Online at http://warse.org/pdfs/2015/ijatcse01412015.pdf CONTEXT AWARE FRAMEWORK IN IOT : A SURVEY," vol. 4, no. 1, pp. 1–9, 2015.

[9] O. R. Devi, "International Journal of Advanced Trends in Computer Science and Engineering Available Online at http://www.warse.org/ijatcse/static/pdf/file/ijatcse02422015.pdf," vol. 4, no. 2, pp. 15–21, 2015.

[10] K. Brandenburg, C. Faller, J. Herre, J. D. Johnston, and W. B. Kleijn, "Perceptual coding of high-quality digital audio," *Proc. IEEE*, vol. 101, no. 9, pp. 1905–1919, 2013. https://doi.org/10.1109/JPROC.2013.2263371

[11] M. A. Austin, B. Muralikrishnan, M. H. Supriya, and P. R. Saseendran Pillai, "Development of a hardware based underwater target identification system," *2009 Int. Symp. Ocean Electron. SYMPOL 2009*, pp. 79–84, 2009.

[12] S. Whibley *et al.*, "WAV Format Preservation Assessment," pp. 1–11, 2016.

[13] D. Luo, W. Luo, R. Yang, and J. Huang, "Identifying compression history of wave audio and its applications," *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 10, no. 3, pp. 30:1--30:19, 2014. https://doi.org/10.1145/2575978

[14] M. Zou and Z. Li, "A wav-audio steganography algorithm based on amplitude modifying," *Proc. - 2014 10th Int. Conf. Comput. Intell. Secur. CIS 2014*, pp. 489–493, 2015.

[15] J. J. Thiagarajan and A. Spanias, *Analysis of the MPEG-1 Layer III (MP3) Algorithm Using MATLAB*, vol. 3, no. 3. 2011.

[16] "MP3." [Online]. Available: http://www.absoluteastronomy.com/topics/MP3. [Accessed: 30-Jun-2018].

[17] "wikipedia Mp3," 2018. [Online]. Available: https://en.wikipedia.org/wiki/MP3. [Accessed: 30-Jun-2018].

[18] "13. Compression and Decompression."

[19] S. Karlsson *et al.*, "Lossless Message Compression Bachelor Thesis in Computer Science," *16th IASTED Int. Conf. Parallel Distrib. Comput. Syst. (PDCS 2004)*, vol. 2006, no. May, pp. 559–564, 2013.

[20] T. Hidayat, M. H. Zakaria, and N. Che Pee, "Comparison of Lossless Compression Schemes for WAV Audio Data 16-Bit Between Huffman and Coding Arithmetic," *Int. J. Simul. Syst. Sci. Technol.*, vol. 19, no. 6, pp. 36.1-36.7, Feb. 2019. https://doi.org/10.5013/IJSSST.a.19.06.36

[21] T. Hidayat, M. H. Zakaria, and A. N. C. Pee, "Lossless coding scheme for data audio 2 channel using huffman and shannon-fano," *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 11, pp. 3467–3477, 2018.

[22] A. R. Of, S. For, C. Mode, I. N. Lossless, and C. On, "Jurnal Teknologi A REVIEW OF STANDARD FOR ADVANCED CODING MODE IN LOSSLESS COMPRESSION ON," vol. 1, pp. 1–5, 2016.

[23] K. Skretting, "IMPROVED HUFFMAN CODING USING RECURSIVE SPLITTING . Høgskolen i Stavanger , Department of Electrical and Computer Engineering P . O . Box 2557 Ullandhaug , N-4004

Stavanger , Norway," *Comput. Eng.*

[24] S. Ashida, H. Kakemizu, M. Nagahara, and Y. Yamamoto, "Sampled-data audio signal compression with huffman coding," *Proc. SICE Annu. Conf.*, pp. 1047–1051, 2004.

[25] M. Johnson, J. Partan, and T. Hurst, "Low complexity lossless compression of underwater sound recordings," *J. Acoust. Soc. Am.*, vol. 133, no. 3, pp. 1387–1398, 2013. https://doi.org/10.1121/1.4776206

[26] M. Hans and R. W. Schafer, "Lossless compression of digital audio," *IEEE Signal Process. Mag.*, vol. 18, no. 4, pp. 21–32, 2001.

[27] M. Hosseini, "A Survey of Data Compression Algorithms and their Applications," *Appl. Adv. Algorithms*, no. April, pp. 1–14, 2012.

[28] H. M. Kasem, M. Elsabrouty, O. Muta, and H. Furukawa, "Performance of perceptual 1-bit compressed sensing for audio compression," *Proc. - IEEE Symp. Comput. Commun.*, vol. 2016-Febru, pp. 477–482, 2016.

[29] A. File and S. Calculations, "AudioMountain.com," pp. 1–2, 2016.

[30] G. Brzuchalski, "Quantization and psychoacoustic model in audio coding in advanced audio coding," *Photonics Appl. Astron. Commun. Ind. High-Energy Phys. Exp. 2011*, vol. 8008, p. 80081J, 2011. https://doi.org/10.1117/12.905576

[31] M. Qiao, A. H. Sung, and Q. Liu, "Revealing real quality of double compressed MP3 audio," *MM'10 - Proc. ACM Multimed. 2010 Int. Conf.*, no. October, pp. 1011–1014, 2010.

[32] R. Yang, Y. Q. Shi, and J. Huang, "Defeating fake-quality MP3," *MMandSec'09 - Proc. 11th ACM Multimed. Secur. Work.*, pp. 117–124, 2009.

[33] T. Hidayat, M. H. Zakaria, and A. N. C. Pee, "A critical assessment of advanced coding standards for lossless audio compression," *Int. J. Simul. Syst. Sci. Technol.*, vol. 19, no. 5, pp. 31.1-31.10, 2018.

[34] I. Conference and O. N. Electronics, "to Assess the Quality of Audio Codecs," no. Icecs, pp. 252–258, 2015.

[35] U. Rahardja, T. Hariguna, & Q. Aini, "Understanding the Impact of Determinants in Game Learning Acceptance: An Empirical Study", International Journal of Education and Practice, 7(3), 136–145, 2019. http://doi.org/10.18488/journal.61.2019.73.136.145

[36] J. V. Rueda-galofre, R. E. Cantillo-carrillo, S. Palomares-garcía, A. Castellano-suárez, and G. W. Ibañez-prada, "An Integrated Framework for Genetic Improvement and Artificial Breeding of Beef Cattle : a Case Study in Livestock Industry of Colombia," Int. J. Adv. Trends Comput. Sci. Eng., vol. 8, no. 1, pp. 62–70, 2019. https://doi.org/10.30534/ijatcse/2019/11812019

[37] Z. R. Al Ashhab, M. Anbar, M. M. Singh, K. Alieyan, and W. I. A. Ghazaleh, "Detection of HTTP Flooding DDoS Attack using Hadoop with MapReduce : A Survey," Int. J. Adv. Trends Comput. Sci. Eng., vol. 8, no. 1, pp. 71–77, 2019.