Indonesian Journal of Computer Science
ISSN 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam No. 1, Padang, Indonesia
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

The Evaluation of Audio Steganography to Embed Image Files Using Encryption and Snappy Compression

Ilham Firman Ashari[1]
Firman.ashari@if.itera.ac.id
[1]Institut Teknologi Sumatera

Article Information

Submitted :17 Jun 2022
Reviewed: 10 Jul 2022
Accepted : 20 Jul 2022

Keywords

Steganography, Cryptography, Image, AES, RC4

Abstract

Images are messages that can be kept secret, so security measures are needed. Techniques that can be used are cryptography and steganography. Steganography can be combined with cryptography to increase security. Images have a relatively large size; Therefore, a compression algorithm is needed. The compression algorithm used is lossless compression. MP3 audio is used as the cover media because it is the most popular audio file. In this study, aspects of imperceptibility, fidelity, recovery, payload, and robustness will be evaluated. The imperceptibility aspect is carried out by observing the RGB Histogram of the image and the audio frequency spectrum, the test results show that there is no significant difference between the audio before and after the image message is inserted. In the fidelity aspect, the PSNR result is above 30 dB. In the payload aspect, the file size after being encrypted with AES and RC4 is larger than just encoded using the base64 encoder. From the recovery aspect, the test results show a BER value of 0. Testing the robustness aspect by manipulating the bitrate, channel mode, and sample frequency, the test results show that the message cannot be extracted.

A. Introduction

The development of information technology is growing and advancing in the current digital era and is felt to have an impact on the convenience of the public in obtaining and accessing information [1]. Information technology has affected all aspects of human life, so humans depend on the development of information technology [2]. We can access digital information anytime and anywhere, digital information is provided via the internet network and we can exchange information easily through the internet network [2].

The ease of exchanging digital information can provide a threat that can be quite serious, a threat that can occur is information leakage, it can even cause financial losses if the data taken theft are important data, for example[3]. Various kinds of techniques can be used by people to be able to access information that does not belong to them, for example with the Man in The Middle Attack (MITM) technique, by bypassing or stealing data while the data is being transmitted [4]. With this MITM attack, attackers can read and divert information that is being transmitted by the sender to the recipient of the message. Therefore, needed a technique that can prevent this from happening, namely cryptography [5]. Cryptography guarantees the confidentiality, integrity, and availability of messages. Data confidentiality, meaning that existing messages cannot be opened or read by unauthorized parties [6][7]. The integrity of the message, the content of the message when it will be sent and after the message is received is the same, in the sense that there is no change in the meaning, shape, and size of the message content and the availability of the message means that confidential messages can be re-accessed at any time by the authorized party [8][9].

Cryptography has provided very good security for data passed in network transmissions, but cryptography can be perceived and arouse suspicion by others and the security factor depends entirely on the algorithms and keys used, because of their randomized or unreadable form. Another technique that can be used to secure information without being able to perceive the existence of the message is steganography. To improve message security, cryptographic techniques can be combined with steganography [10]. In the concept of steganography, several factors are interrelated with one another, namely capacity, robustness, and security. These three factors influence each other, when one factor is increased, the other factors will decrease [11][12]. Steganography techniques that can insert messages in large numbers will cause a decrease in the quality of the resulting stego object. If we want the object from the resulting stego file to be resistant and sturdy against attacks, it will affect the capacity of messages that can be inserted into the stego object [13]. The balance of these factors will determine the resulting stego object.

Digital messages can be text, images, audio, or video. Images can be private and confidential, so they need to be protected. The most frequently used image extensions are (.png, .jpg, .bmp, dan .gif) [14]. Image is one of the digital information that can have a larger size than text messages, especially RGB images. Due to its large size, the image to be inserted must be reduced in size first. The method used is to compress the image, so the image bytes will be reduced in size [15]. The capacity of the picture message greatly affects the final result of the resulting stego audio because the larger the inserted image file, the lower the

quality of the stego object will be, according to the trade-off factor of steganography [16].

Compression methods in image files can be divided into two, namely lossy and lossless compression. The lossy compression algorithm is a compression algorithm that is irreversible, meaning that the compressed image cannot be returned to its original form, because the compressed image pixels are different from the original image, but this difference is not too significant. The lossless compression algorithm is reversible, meaning that the compressed image can be reconstructed like an image file at the beginning of compression, so that the compression ratio in the lossless algorithm is not too large, compared to the lossy algorithm [17]. A lossless compression algorithm can reconstruct the pixel value of the image according to the original image .

Previous research related to steganography has been carried out by Putri, Ashari, and Rahmat. Putri in her research, using LSB encoding method and encryption method using Blowfish and RC4 [18]. Secret text messages will be encrypted with blowfish then after that, text messages encrypted with RC4, the goal is to improve data security. The cover object used in the study is an image file. The results obtained from the research of Putri et al. are PSNR values above 56 dB. Research by Ashari [19], using the LSB technique to insert a text message on the MP3 audio using the LSB technique. Before the message is inserted, the message will be encrypted first using the AES algorithm. The results obtained by the stego object after manipulation, the message cannot be extracted again. The message has met the aspects of recovery, fidelity, and imperceptibility. In a study conducted by Rahmat [20], he proposed a combination technique between the RC4 and AES algorithm and evaluated the encryption algorithm. Messages that can be inserted into the image cover are texts. The method used for insertion is LSB and the message used is a text message that will be inserted in the image. The results of the study show that the AES algorithm has a better speed than RC4 in performing encryption and decryption. The larger the message size, the smaller the PSNR value and vice versa. Another research conducted by Ashari [21] is to evaluate the LSB method by comparing the embedding method with LSB+1, +2, and +3, where the results are that the LSB +3 encoding method is more optimal, but the PSNR value obtained is not as good as the encoding method. LSB+2. For text messages to be returned, an encoding process with a Base64 Encoder is required.

In this study, a lossless compression algorithm will be used before the image is inserted into the audio file, this is because the purpose of steganography is that the inserted message must be the same as the message that has been extracted from the stego object. The lossless compression algorithm that will be used is snappy compression. Snappy compression, formerly known as Zippy, is an open-source compression algorithm developed by Google [22]. The snappy algorithm is based on the LZ77 algorithm which is a dictionary compression-based algorithm [23]. Snappy Compression is usually used to be implemented on Big Data. The snappy compression algorithm has two forms, namely basic and streaming. The basic form for each compression must match the memory capacity available on the computer, while the streaming form can adjust and compress large amounts of data. Snappy compression has a faster performance for compression and

decompression than other compression algorithms such as GZip and LZO and is splitable, meaning it can compress data in the form of a byte stream [24].

The steganography method used is based on spatial domains using LSB (Least Significant Bit) technique, hiding the message in a cover object on the LSB bit or the bit that has the least effect on the cover object, but the presence of image bits on the audio cover can be said to be less secure, by Therefore, encryption techniques are needed to secure picture messages [25]. The LSB method is faster and lighter and is suitable for large image sizes, but to optimize the quality of the resulting object stego, before the image is inserted, it must first be compressed with the snappy compression algorithm. In this study, stream encryption algorithms and block ciphers will be used, to compare the most optimal algorithms in terms of performance and secure messages while considering the quality of the resulting stego object. The stream cipher algorithm used is the RC4 algorithm and the block cipher algorithm used is the AES (Advanced Encryption Standard) algorithm. The test carried out in this study was to see and evaluate the effect of the size of the image message using encryption and compression algorithms on the results of the resulting stego object.

B. Research Method

The stages in steganography are divided into two, namely the encryption and encoding stages, and the decoding and decryption stages. These two things are inseparable. The encryption and encoding processes can be seen in Figure 1, and the decoding and decryption processes can be seen in Figure 2.

2.1 Encryption and Embedding Process

The first step is to input the image file as a message and the audio file as the cover media, after that the audio and image files will be validated. The allowed image file extensions are (.jpg, .png, .bmp) and the allowed audio file extensions are (.mp3) otherwise they will be rejected. If the two files match, then both files will be converted into byte streams to compare and measure the capacity of picture messages that can be inserted into audio files. If the image file size is larger than the audio file size, the image file cannot be inserted. After the image file is inputted according to the size of the audio file, then the image encryption process will then be carried out. In this study, there are two choices when it comes to encryption, namely using the AES or RC4 encryption algorithm. Furthermore, encryption is carried out according to the selected algorithm (AES/RC4). The result of the encryption process is in the form of encryption bytes, which will then be compressed using the snappy algorithm. The compressed image bytes cannot be directly inserted into the audio, because when the extraction will be carried out there will be bytes that cannot be returned if the byte is 0, when the decryption process is carried out. Therefore, the image bytes must be first encoded using the Base64 encoder. After encoded by using Base64. Bytes from Base64 Encoder will be compressed again using snappy algorithm. Furthermore, the insertion process using the LSB method is carried out into the audio file, so that the stego audio file is obtained.

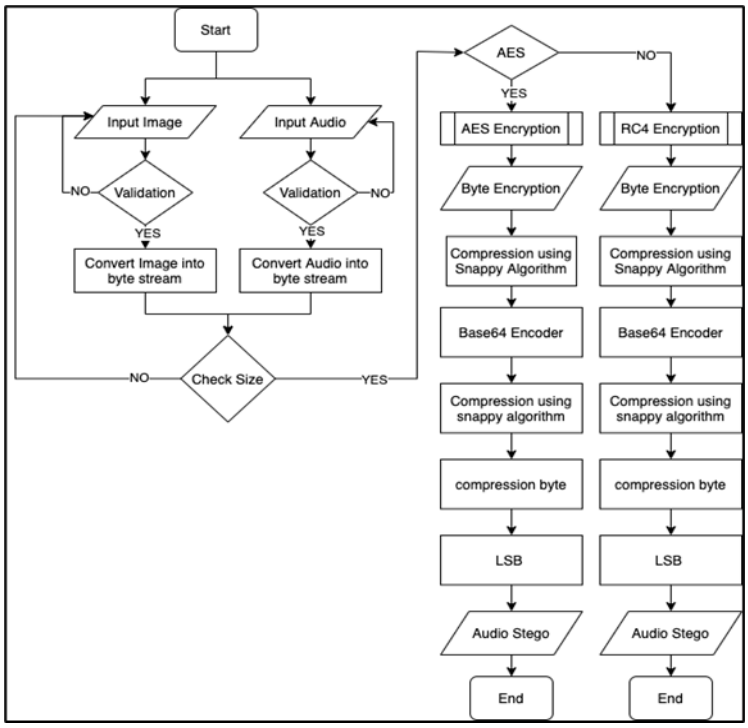


Figure 1. Encryption and Embedding Process

The explanation of the stages of the encryption and encoding process can be seen in subchapter 2.1, an explanation of the extraction and decryption stages can be seen in subchapter 2.2, an explanation related to the input validation process of each file can be seen in subchapter 2.3, and the compression process with the snappy algorithm can be seen in subchapter 2.4.

2.2 Extraction and Decryption Process

The first step is to input an audio stego file containing confidential information, after which validation and checking will be carried out to ensure that the input file is a file with the extension (.mp3). If the file extension is correct, the audio file will be converted to a byte stream. Furthermore, each LSB byte will be taken from the audio byte stream using the LSB technique. These bytes are then converted with the ISO-8859-1 encoder. Next will be decoded using Base64 Decoder. As a result, this decoder will be decompressed using the Snappy algorithm. The result of byte decompression will be decrypted using AES or RC4 algorithm, according to the selected encryption algorithm. After decryption with the appropriate algorithm, the resulting bytes from the decryption will be converted into an image file.

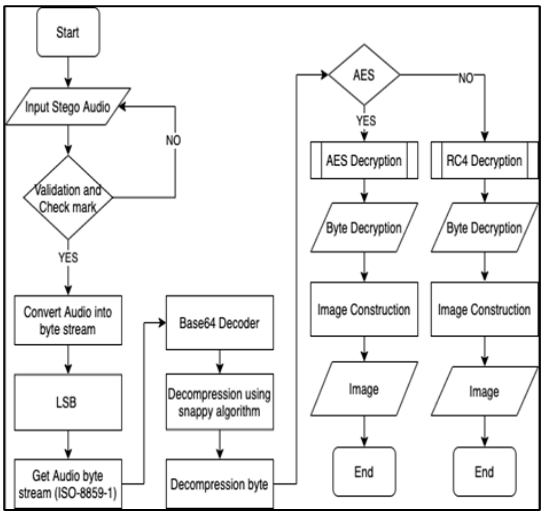


Figure 2. Extraction and Decryption Process

2.3 Validation Process

The stages of the process carried out to determine the maximum image capacity that can be inserted into MP3 audio can be seen in Figure 3, with the following explanation, namely:

1. The application will check whether the mp3 contains ID3v1 or ID3v2 tags
2. If it contains ID3v1 or ID3v2, then check the size and immediately read the MP3 Frame header
3. If it doesn't contain tags, then read the MP3 frame header
4. Count how many frames are MP3
5. Calculate the size of the MP3 frame
6. Read markers ^&*
7. Convert markers to bytes
8. Scan from the first position in each frame after the tag whether it contains a marker

Perform calculations to calculate the maximum capacity as in the last process of figure 3.

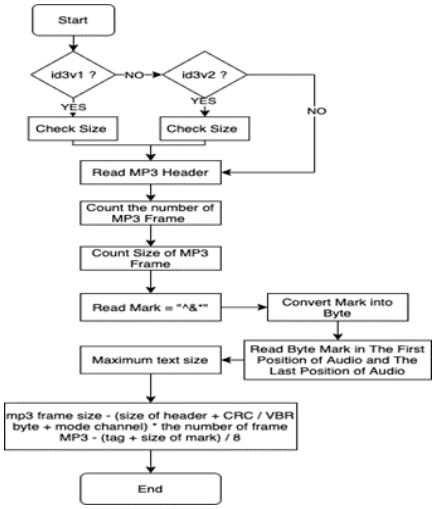


Figure 3. Validation Process

2.4 Snappy Algorithm Compression Process

The snappy algorithm is not bit-based in its compression process, unlike other encoder algorithms such as the snappy algorithm is not bit-based in its compression process, unlike other encoder algorithms such as Huffman Encoding, because it does not use entropy values. The first byte of the byte stream is the length of uncompressed data, which is stored as little-endian then used for variable-length encoding, which can be seen in Figure 4.

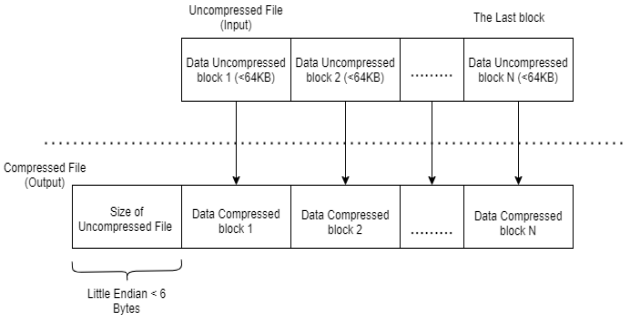


Figure 4. Input and Output Data Using Snappy Algorithm [26]

Based on coding theory, variable-length encoding is a variable code that maps input data or symbols into a variable number of bits. Using variable-length codes allows data to be compressed and decompressed without any difference, meaning that the output is lossless data (there is no difference before and after compression). For the type of element used during the compression process, this snappy algorithm uses 2 bits, unlike the initial 4-bit byte or byte tag which is an uncompressed bit. The data structure of the snappy algorithm can be seen in Figure 5.

The four elements used for encoding are as follows [22]:

1. 00 (literal elements which are uncompressed data, the first 6 bits are used to store the data length. Data lengths exceeding 60 will be stored as integer bytes with a range (1 to 4), where if 60 means 1 byte, 61 means 2 bytes, 62 means 3 bytes, and 63 means 4 bytes).
2. 01 (1 byte after the tag byte will be used as part of the offset. The offset is stored as long as 11 bits, with the length of the compressed data referring to the dictionary is 3 bits of the tag byte).
3. 10 (2 bytes after the tag byte will be stored as part of the offset. The length of the compressed data referring to the dictionary is 6 bits of the tag byte).
4. 11 (offset is stored as 4 bytes of the little-endian integer after the byte tag and Length is stored as 6 bits of the byte tag).

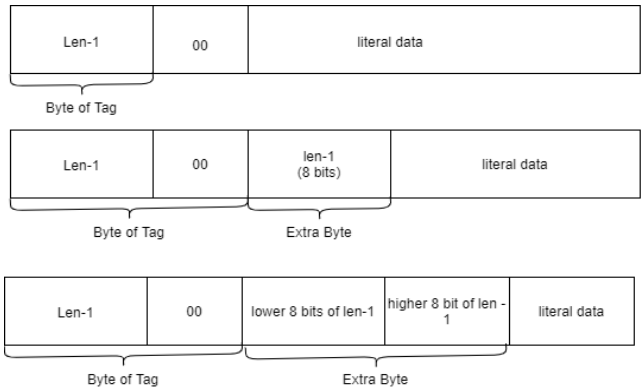


Figure 5. Data structure using snappy algorithm [26]

C. Result and Discussion

The quality of the steganography file is measured using five parameters such as imperceptibility, payload, robustness, recovery, and fidelity. [27]. The measurement of these five steganographic parameters can not only be done subjectively but also can be done objectively. Imperceptibility was measured by subjective observation by looking at the histogram of the image and the audio spectrum of steganography, as well as seeing the difference in the size of the audio file before and after the message was inserted. Objective measurement involves measuring fidelity, recovery, and payload parameters. The technical details of testing these four parameters will be described as follows.

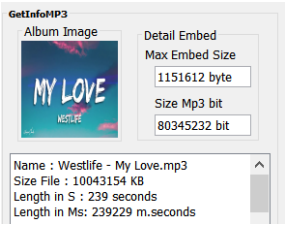
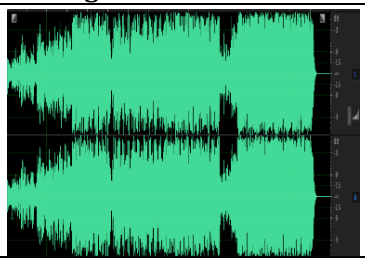
3.1 Imperceptibility

Imperceptibility testing ensures that the audio quality of steganography is not much different from before and after the image message is inserted [28]. In this test, the quality of steganographic audio files will be compared before inserting image messages (JPEG, PNG, and BMP) with after inserting image messages of various sizes, both before being compressed and after being compressed using the snappy compression algorithm and encryption algorithm (AES and RC4). Observation of imperceptibility by looking at the color image histogram before inserting it into the cover audio file and after extracting it from the steganography audio file, besides that, visualization of the spectrum of the initial audio file (before inserting the message) and after inserting the image message will also be seen [29]. The test results on the imperceptibility aspect can be seen in tables 3 to 6.

In table 1 is the audio cover and spectrum of audio that will be used on the experiment. The name of MP3 audio cover is **Westlife – My Love.mp3,** which has a size of 80345232 bits. In table 2 is the original image files before embedding, compression, and encryption is performed. In this study, to insert an image, an encoding process is needed first using the Base64 encoder algorithm, this is so that byte 0 is still readable when the extraction process from a steganographic audio file is carried out. Therefore, in this study, visual observations will also be made using the base64 encoder algorithm. In table 3, the following is a test using the Snappy algorithm and Base64 Encoder. In this test, we will compare the size of the original image after it is compressed and encoded using the Base64 Encoder.

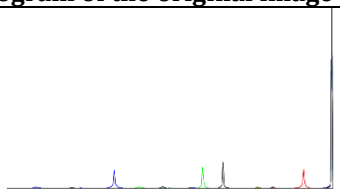
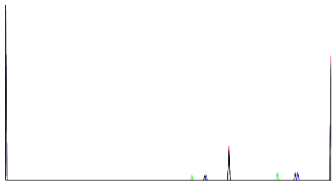
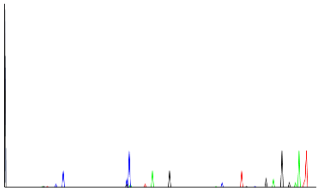
From the test based on table 3, it can be seen in the RGB histogram section between the original image file in table 2 and the image histogram in table 3 after extraction that there is no difference, besides that it can also be seen in the initial audio spectrum with audio steganography after the message is inserted, the difference is not too significant. The next test is to use an encryption algorithm. Image size reduction after compression and encode with base64 encoder is around 15-20 %.

Table 1. Visualization of original Image files and Audio Files

Mp3 audio size before message insert	**Original Audio frequency spectrum before message insert**
	

The audio spectrum is visualized using adobe audition, where the spectrum is visualized from the left and right channels.

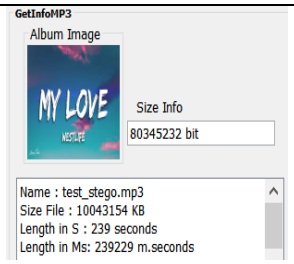
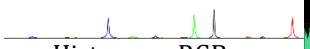
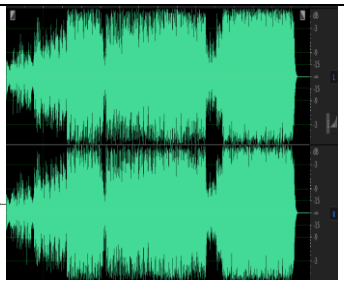
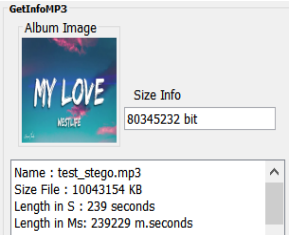
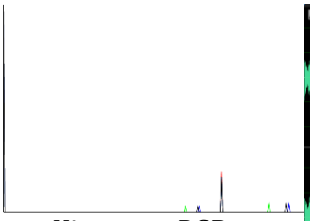
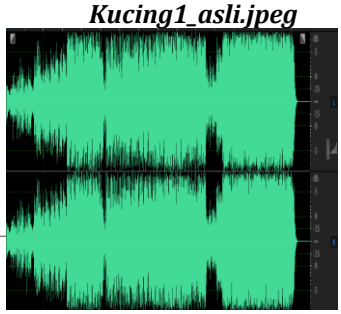
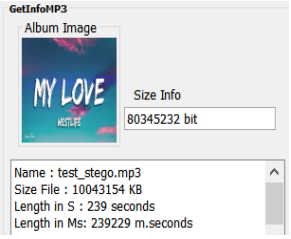
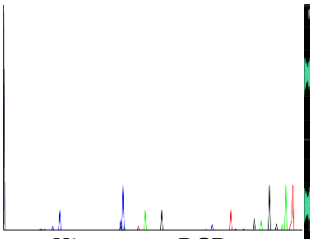
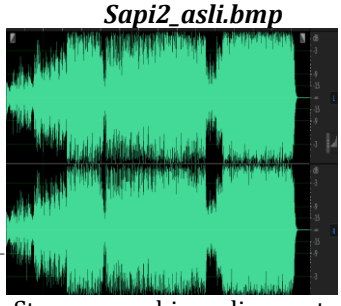
Table 2. Visualization of original Image files and Audio Files

Image size before message insert	**RGB histogram of the original image**
Kucing1_asli.jpeg (15226 bytes) 	 Histogram RGB kucing_1_asli.jpeg
Sapi2_asli.bmp (62056 bytes) 	z  Histogram RGB Sapi2_asli.bmp
Jerapah3_asli.png (77430 bytes) 	 Histogram RGB Jerapah3_asli.png

Tables 4 and 5 are the results of tests using the AES and RC4 encryption algorithms. The process carried out is to encode the image first by using the Base64 Encoder algorithm, then proceed by using the encryption algorithm. On table 4, the following is a test using the Snappy and AES algorithms, in this test a

comparison will be made between the visualization of the original image histogram and after it is compressed and encrypted using AES.

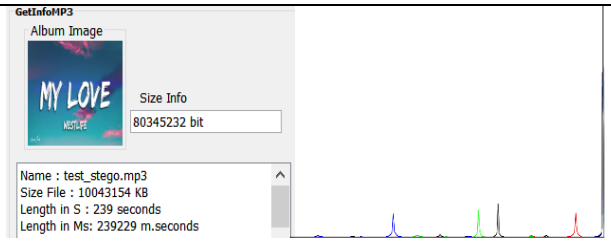
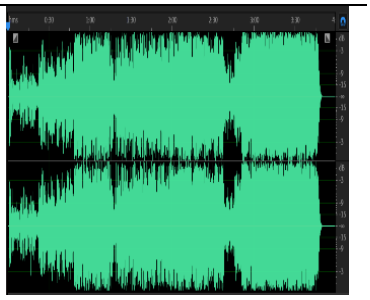
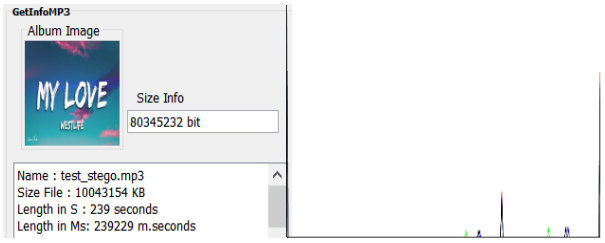
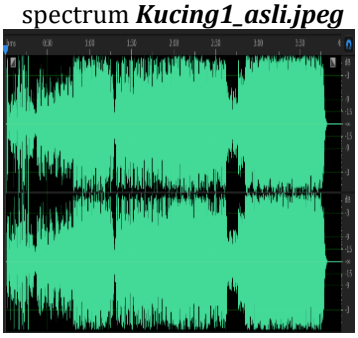
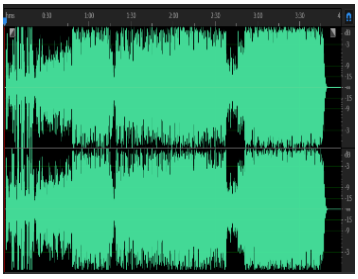
Table 3. Visualization of image and audio files before and after compression and encoding with Base64 Encoder

Image size after being compressed and decoded with Base64 Encoder	**MP3 audio size after inserting picture message using snappy and Base64 Encoder**	**RGB histogram after the image is extracted and decoded with snappy and Base64 Decoder**	**Spectrum steganographic audio files with snappy and Base64 Encoder**
After being compressed and encoded (13809 bytes),  ***Kucing1_asli.jpeg*** After extracted and decoded (15226 byte)		 Histogram RGB Kucing_decode_1.jpg	 Steganographic audio spectrum ***Kucing1_asli.jpeg***
After being compressed and encoded (60187 bytes),  ***Sapi2_asli.bmp*** After extracted and decoded (62056 bytes)		 Histogram RGB Sapi_decode_2.bmp	 Steganographic audio spectrum ***Sapi2_asli.bmp***
After being compressed and encoded (75592 bytes),  ***Jerapah3_asli.png*** After extracted and decoded (77430 bytes)		 Histogram RGB Jerapah_decode_2.png	 Steganographic audio spectrum ***Jerapah3_asli.png***

On the table 4, the following is a test using the Snappy and AES algorithms, in this test a comparison will be made between the visualization of the original image histogram and after it is compressed and encrypted using AES

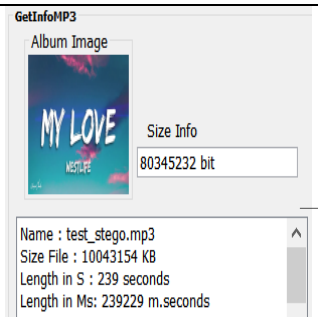
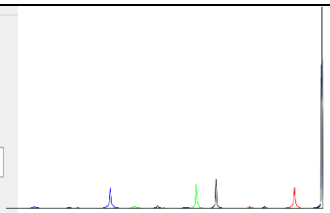
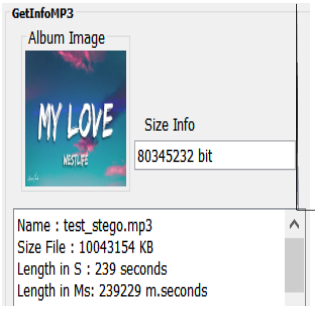
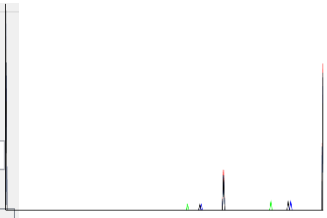
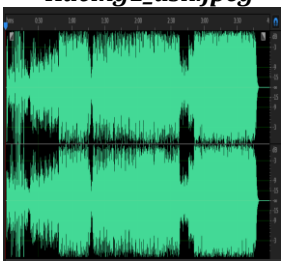
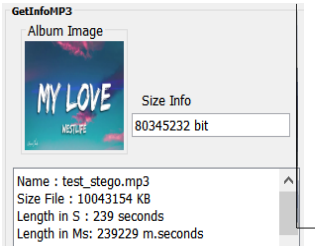
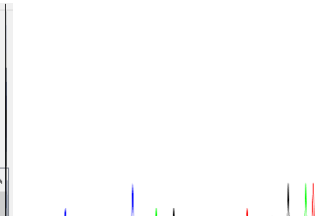
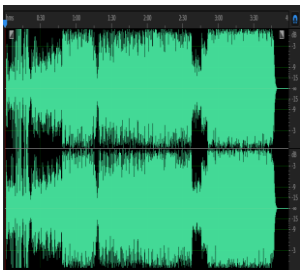
Table 4. Visualization of image and audio files before and after compression and encryption with AES

Image after extracted	**Audio after message insert using snappy and AES**	**RGB histogram after the image is extracted and decrypted with snappy and AES**	**Spectrum audio files after compression and encryption using snappy and AES**

After being compressed and encryption using AES (13978 bytes)  ***Kucing1_asli.jpeg*** After extracted and decrypted AES (15226 bytes)	 Histogram RGB Kucing_AES_1.jpg	 Steganographic audio spectrum ***Kucing1_asli.jpeg***
After being compressed and encryption using AES (60789 bytes)  ***Sapi2_asli.bmp*** After extracted and decrypted AES (62056 bytes)	 Histogram RGB Sapi_AES_2.bmp	 Steganographic audio spectrum ***Sapi2_asli.bmp***
After being compressed and encryption using AES (75904 bytes)  ***Jerapah3_asli.png*** After extracted and decrypted AES (77430 bytes)	 Histogram RGB Jerapah_AES_2.png	 Steganographic audio spectrum ***Jerapah3_asli.png***

Based on table 4, the results show that there is an increase in the image file size after compression and encryption using AES. The increase in image size after compression and encryption of the Base64 encoder algorithm is about (2 - 2.5) %. This is because AES is an encryption algorithm using the concept of block ciphers. So, the message is encrypted in block bytes, not in bytes. In this study, experiments were also carried out using the RC4 encryption algorithm, this is to compare the most efficient algorithms in producing compressed and encrypted image sizes. In the table 5, the following is a test using the Snappy compression algorithm and RC4 encryption, in this test a comparison will be made between the original image size and after it is compressed and encrypted using RC4. The test results can be seen in Table 5.

Table 5. Visualization of image and audio files before and after compression and encryption with RC4

Image After Extracted	Audio after message insert using snappy and RC4	RGB histogram after the image is extracted and decrypted with snappy and RC4	Spectrum audio files after compression and encryption using snappy and RC4
After compressed and encrypted using RC4 (13509 bytes)  ***Kucing1_asli.jpeg*** After extracted and decrypted RC4 (15226 bytes)			 Steganographic audio spectrum ***Kucing1_asli.jpeg***
After compressed and encrypted using RC4 (60560 bytes)  ***Sapi2_asli.bmp*** After extracted and decrypted RC4 (62056 bytes)			 Steganographic audio spectrum ***Sapi2_asli.bmp***
After compressed and encrypted using RC4 (75863 bytes)  ***Jerapah3_asli.png*** After extracted and decrypted RC4 (77430 bytes)			 Steganographic audio spectrum, ***Jerapah3_asli.png***

From the test from table 5, the increase in image size after compression and encryption with RC4 compared to Base64 Encoder is around 1 - 1.5 %. If we look at the increase in image size between RC4 and AES, the biggest increase in image size is in the AES algorithm, this will certainly be significant with a larger image size test case. Regarding the security between the RC4 and AES algorithms, the AES algorithm is safer. The Imperceptibility value is influenced by the number of bytes of the image to be inserted, the more bytes of the image, of course, the quality of the steganographic audio file will decrease, this will certainly have an impact on the clarity of the sound of the steganographic audio file that is produced and listened to.

3.2 Fidelity

Measurement of fidelity aspect using SNR and PSNR. The measurement of the fidelity aspect is to ensure that the quality or quality of the steganography audio file after being inserted can be measured and assessed objectively [28]. The standard to show the quality of a good audio steganography file is with a PSNR value greater than 30 dB, assuming that with a minimum value of 30 dB the message cannot be perceived as being in the steganography file and the steganographic audio quality does not cause much noise. To measure SNR, the following formula (1) can be used [30]:

$$SNR = \frac{XB-YE}{XB} \quad (1)$$

XB is the number of bits of MP3 audio measured and YB is the number of bits that are changed and pasted into the MP3 audio file, as in the formula (2) [31].

$$PSNR = 10\ log_{10} \frac{\sum_{i=1}^{n} {y_1}^2}{\sum_{i=1}^{n} {y_1}^2 - \sum_{i=1}^{n} {y_0}^2} \quad (2)$$

Where y1 is the number of data bits that cover the original audio and y0 is the number of bits of audio steganography. The test results can be seen in Table 6.

Table 6. The results of testing the fidelity aspect by comparing the aspects of the AES and RC4 encryption algorithms

Image Type	Original Image Size	Original Audio Size	Steganography Audio File Size	Encryption Algorithm	Changed Bits	Snappy Algorithm	SNR	Error Rate	PSNR
Jerapah1.png	76 kb	9.57 MB	9.57 MB	AES	50896	Yes	99.51	0.49 (4.9 %)	50.43
Sapi2.bmp	61 KB	9.57 MB	9.57 MB	RC4	41532	Yes	99.54	0.46 (4.6 %)	52.87
Jerapah1.png	76 kb	9.57 MB	9.57 MB	RC4	51220	Yes	99.49	0.51 (5.1 %)	50.43
Sapi2.bmp	61 KB	9.57 MB	9.57 MB	AES	41546	Yes	99.58	0.42 (4.2 %)	53.14

From the fidelity test based on table 6, it can be concluded that the image with the same size using the RC4 algorithm has the largest PSNR value, meaning that the audio steganography has better quality than using the AES algorithm. In terms of changed bits, it can be said that images encrypted with the AES algorithm produce more bits changed than using the RC4 algorithm, this will result in the PSNR value getting smaller and the error rate getting bigger.

3.3 Payload

From the fidelity test, it can be concluded that the image with the same size using the RC4 algorithm has the largest PSNR value, meaning that the audio steganography has better quality than using the AES algorithm. In terms of changed bits, it can be said that images encrypted with the AES algorithm produce more bits changed than using the RC4 algorithm, this will result in the PSNR value getting smaller and the error rate getting bigger. The payload test results can be seen in Table 7.

Table 7. Payload test results by comparing image sizes after being compressed and encrypted

Message Type	File Size after Encoded with Base64 Encoder	File size after compression with snappy (bit)	File Size after compression with snappy and AES (bit)	File size after compression with snappy and RC4 (bit)
(Jerapah1.png)	(117,030 bytes)	(99,780 bytes)	(102,114 bytes)	(101,717 bytes)
(Sapi2.png)	(65,926 bytes)	(59,200 bytes)	(61,262 bytes)	(61,222 bytes)

From the payload test, it can be concluded that the file size after being encrypted with AES and RC4 which is added with compression is smaller than being encoded using a base64 encoder. If we compare which encryption algorithm is more efficient, it can be said that the RC4 algorithm is more efficient than the resulting file size. The smaller the size of the inserted image, the better the quality of the resulting steganography audio file will be.

3.4 Recovery

Recovery testing is used to ensure that the message file inserted is no different from the message file after extraction [28]. The parameters compared are the size of the message before it is inserted into the audio cover file and the size of the message after it is extracted from the audio steganography file. In the recovery test using measurements with bit error rate (BER). So by using this measurement it can be seen that if there is a byte error from the message character that is not the same, it will be added up and the percentage of the total bit calculated using the BER formula (3) [30].

$$BER = \frac{ER}{NK} \tag{3}$$

ER is the number of bytes of unequal message characters and NK is the total number of message bytes to be inserted in the MP3 audio file. Below is pseudocode to calculate the BER value in an audio steganography file that is used to compare the message content after extraction in terms of the number of characters and message characters.

```
Start -> 0
End -> 0
diff_char -> 0
if (originalteks.length()>extractteks.length())
       end->extractteks.length()
else if(originalteks.length()<extractteks.length())
       end->originalteks.length()
If(originalteks.length()!=extractteks.length())
       diff_char->Math.abs(originalteks.length()-extractteks.length())
while(start<end)
       if(originalteks.charAt(start)!=originalteks.charAt(start++))
         diff_char++;
```

BER test results can be seen in Table 8.

Table 8. BER test results using compression and encryption algorithms

Image Type	Bytes of messages inserted with snappy and AES	Message bytes with snappy and RC4	Message bytes with snappy and AES	Message bytes extracted with snappy and RC4	BER
 (Sapi2.bmp)	83,197	83,189	83,197	83,189	0
 (Kucing1.jpeg)	18,290	18,482	18,290	18,482	0

The test results show a BER value of 0, meaning that there is no difference between the message bits before insertion and after insertion into the steganography audio file. The recovery aspect has been 100% successful.

3.5 Robustness

Robustness aspect testing is used to assess whether the inserted steganography audio file is resistant to manipulation attacks [19]. Testing is done by changing the frequency of the sample rate of the audio file, manipulating the bit rate and sample frequency of the audio file. The test results on the robustness aspect can be seen in table 9 to table 11.

Table 9. MP3 Cover Manipulation Test Results on Bit Rate

Audio Steganography	Initial Bit Rate (kbps)	Max Capacity Before Bitrate Modification (bytes)	Final Bit Rate (kbps)	Max Capacity After Bit Rate Modification	Extraction
Westlife.mp3	128	307890	250	532021	Failed
Pupus.mp3	128	389330	290	610132	Failed

The results of the mp3 cover manipulation test on channel mode can be seen in table 10.

Table 10. MP3 Cover Manipulation Test Results on Channel Mode

Audio Steganography	Initial Channel Mode	Max Capacity Before Channel Mode Modification (bytes)	Final Channel Mode	Max Capacity After Channel Mode Modification (byte)	Extraction
Westlife.mp3	Joint-Stereo	307890	Stereo	298202	Failed
Pupus.mp3	Joint-Stereo	389330	Dual-Channel	361263	Failed

The test results on the Sample frequency in audio files can be seen in table 11.

Table 11. MP3 Cover Manipulation Test Results on Sample Frequency

Audio Steganography	Initial Frequency (Hz)	Max Capacity Before Modification of sample frequency (byte)	Final Frequency (Hz)	Max Capacity After Modification of sample frequency (byte)	Extraction
Westlife.mp3	44100	307890	48000	304530	Failed
Pupus.mp3	44100	389330	32000	382793	Failed

Based on the test results on the robustness aspect (bit rate, channel mode, and sample frequency), it can be concluded that audio steganography cannot extract the message.

D. Conclusion

In this study, image messages with extensions (PNG, JPG, and BMP) have been successfully inserted into MP3 audio media. Evaluation of the quality of the inserted and extracted images and steganography audio files were tested based on five aspects, namely imperceptibility, fidelity, payload, recovery, and robustness aspects. Testing the imperceptibility aspect by observing the histogram of the RGB image message from the image file before the message is inserted and after the message is inserted. In addition, observations were also made on the frequency spectrum of the original audio file with audio steganography. The results of the imperceptibility test of the audio file can be concluded that there is no significant difference between the frequency spectrum of the audio file before and after the message is inserted. From the extracted image to the RGB histogram between the image file before it was inserted into the audio and after it was extracted there was no change, meaning that the extracted image message was the same as the inserted image message. The test results on the fidelity aspect obtained PSNR results from the image message on average above 30 dB for image sizes > 50 kB, this shows that for the fidelity aspect the audio quality of steganography can be said to be good. In the payload aspect, the file size after being encrypted with AES and RC4 is larger than just encoded using a base64 encoder. If we compare which one is more effective and efficient, it can be said that the RC4 algorithm is more efficient in terms of the resulting file size. The smaller the size of the inserted image, the better the quality of the resulting steganographic audio file will be. From the recovery

aspect, the test results show a BER value of 0, meaning that there is no difference between the message bits before and after being inserted into the steganography audio file.

E. References

[1] I. F. Ashari, "Graph Steganography Based On Multimedia Cover To Improve Security and Capacity," in *2018 International Conference on Applied Information Technology and Innovation (ICAITI)*, 2018, no. April 2019, pp. 194–201.

[2] I. F. Ashari, "Implementation of Cyber-Physical-Social System Based on Service Oriented Architecture in Smart Tourism," *J. Appl. Informatics Comput.*, vol. 4, no. 1, pp. 66–73, 2020, doi: 10.30871/jaic.v4i1.2077.

[3] N. I. Munawar, Zen and Putri, "Keamanan Jaringan Komputer Pada Era Big Data," *J-SIKA| J. Sist. Inf. Karya Anak Bangsa*, vol. 02, no. 01, pp. 14–20, 2020.

[4] A. Sebbar, K. Zkik, Y. Baddi, M. Boulmalf, and M. D. E. C. El Kettani, "MitM detection and defense mechanism CBNA-RF based on machine learning for large-scale SDN context," *J. Ambient Intell. Humaniz. Comput.*, vol. 11, no. 12, pp. 5875–5894, 2020, doi: 10.1007/s12652-020-02099-4.

[5] A. Mallik, A. Ahsan, M. M. Z. Shahadat, and J. C. Tsou, "Man-in-the-middle-attack: Understanding in simple words," *Int. J. Data Netw. Sci.*, vol. 3, no. 2, pp. 77–92, 2019, doi: 10.5267/j.ijdns.2019.1.001.

[6] I. F. Ashari, M. Alfarizi, M. N. K, and M. A. H, "Vulnerability Analysis and Proven On The neonime . co Website Using OWASP ZAP 4 and XSpear," *J. Teknol. Komput. dan Sist. Inf.*, vol. 5, no. 2, pp. 75–81, 2022.

[7] I. F. Ashari and V. Adhelia, "Expert System and IoT for Diagnose of Feline Panleukopenia Virus Using Certainty Factor," *Matrik J. Manajemen, Tek. Inform. dan Rekayasa Komput.*, vol. 21, no. 2, pp. 451–462, 2022, doi: 10.30812/matrik.v21i2.1517.

[8] C. Biswas, U. Das Gupta, and M. M. Haque, "An Efficient Algorithm for Confidentiality, Integrity and Authentication Using Hybrid Cryptography and Steganography," *2nd Int. Conf. Electr. Comput. Commun. Eng. ECCE 2019*, pp. 1–5, 2019, doi: 10.1109/ECACE.2019.8679136.

[9] I. F. Ashari, A. J. Aryani, and A. M. Ardhi, "DESIGN AND BUILD INVENTORY MANAGEMENT INFORMATION SYSTEM," vol. 9, no. 1, pp. 27–35, 2022.

[10] V. Reshma, S. Joseph Gladwin, and C. Thiruvenkatesan, "Pairing-free CP-ABE based cryptography combined with steganography for multimedia applications," *Proc. 2019 IEEE Int. Conf. Commun. Signal Process. ICCSP 2019*, pp. 501–505, 2019, doi: 10.1109/ICCSP.2019.8698053.

[11] V. Verma, S. K. Muttoo, and V. B. Singh, "Enhanced payload and trade-off for image steganography via a novel pixel digits alteration," *Multimed. Tools Appl.*, vol. 79, no. 11–12, pp. 7471–7490, 2020, doi: 10.1007/s11042-019-08283-9.

[12] M. H. N. Azam, F. Ridzuan, M. N. S. M. Sayuti, and A. A. Alsabhany, "Balancing the Trade-Off between Capacity and Imperceptibility for Least Significant Bit Audio Steganography Method: A New Parameter," *2019 IEEE Conf. Appl. Inf. Netw. Secur. AINS 2019*, pp. 48–53, 2019, doi:

10.1109/AINS47559.2019.8968707.
[13] D. H. Zulfikar, "Quality Factor terhadap Kapasitas Pesan Rahasia pada Steganografi Citra JPEG dan Kualitas Citra Stego," *JUSIFO (Jurnal Sist. Informasi)*, vol. 6, no. 2, pp. 89–100, 2020, doi: 10.19109/jusifo.v6i2.6608.
[14] B. Sinha, "Comparison of PNG & JPEG Format for LSB Steganography," *Int. J. Sci. Res.*, vol. 4, no. 4, pp. 198–201, 2015.
[15] B. D. Raharja and P. Harsadi, "Implementasi Kompresi Citra Digital Dengan Mengatur Kualitas Citra Digital," *J. Ilm. SINUS*, vol. 16, no. 2, pp. 71–77, 2018, doi: 10.30646/sinus.v16i2.363.
[16] H. Antonio, P. W. C. Prasad, and A. Alsadoon, "Implementation of cryptography in steganography for enhanced security," *Multimed. Tools Appl.*, vol. 78, no. 23, pp. 32721–32734, 2019, doi: 10.1007/s11042-019-7559-7.
[17] S. H. WE Pangesti, G Widagdo, D Riana, "Implementasi Kompresi Citra Digital Dengan Membandingkan Metode Lossy dan Lossless Compression Menggunakan Matlab," *J. KHATULISTIWA Inform.*, vol. 8, no. 1, pp. 53–58, 2020.
[18] A. D. Putri Ariyanto, E. H. Rachmawanto, D. R. Ignatius Moses Setiadi, and C. A. Sari, "Performance Analysis of LSB Image Steganography Combined with Blowfish-RC4 Encryption in Various File Extensions," *Proc. 2019 4th Int. Conf. Informatics Comput. ICIC 2019*, pp. 0–5, 2019, doi: 10.1109/ICIC47613.2019.8985848.
[19] I. F. Ashari, "Aplikasi steganografi pesan teks pada media audio mp3 menggunakan metode penyisipan least significant bit dan advanced encryption standard skripsi," pp. 1–114, 2015.
[20] R. Sulaiman, "Combination and comparison of AES and RC4 cryptography in least significant bit (LSB) method in digital image to improve message security," *J. Inform.*, vol. 12, no. 2, p. 45, 2018, doi: 10.26555/jifo.v12i2.a8667.
[21] I. F. Ashari, "The Evaluation of Image Messages in MP3 Audio Steganography Using Modified Low-Bit Encoding," *Telematika*, vol. 15, 2021.
[22] B. Szabari and A. Kiss, "Word pattern prediction using Big Data frameworks," *Acta Univ. Sapientiae, Inform.*, vol. 12, no. 1, pp. 51–69, 2020, doi: 10.2478/ausi-2020-0004.
[23] K. Kovacs, "A Hardware Implementation of the Snappy Compression Algorithm," 2019, [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-85.html.
[24] A. Wenas, "Improving Data Warehouse Performance Using Filesystem Technology with GZIP , LZJB and ZLE Compression," *J. Inform. dan Sist. Inf. Univ. Ciputra*, vol. 02, no. 02, 2016.
[25] K. Saraswati and P. S. Sharma, "A Literature Survey on Stenography Approach Based on Different LSB Technique," no. 2, pp. 1–6, 2021.
[26] J. Fang and H. P. Hofstee, "MSc THESIS An FPGA-based Snappy Decompressor-Filter," no. January, 2018, doi: 10.13140/RG.2.2.30215.44962.
[27] Rihartanto, D. S. B. Utomo, and A. Rizal, "Implementasi Image Tilling Pada Penyembunyian Pesan Menggunakan LSB," *Proceeding SINTAK*, pp. 186–192,

2019.

[28] M. Nursalman, J. R. Rachman, and F. Sidik, "Implementation of Low Bit Coding Algorithm and Cipher Block with Electronic Code Book Mode for Data Legality in Audio Steganographic Streaming," *2018 Int. Conf. Inf. Technol. Syst. Innov. ICITSI 2018 - Proc.*, pp. 330–335, 2018, doi: 10.1109/ICITSI.2018.8695921.

[29] D. E. Kurniawan, N. R. Hartadi, and P. Prasetyawan, "Analisis Hasil Teknik Penyembunyian Hak Cipta Menggunakan Transformasi DCT dan RSPPMC pada Jejaring Sosial," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 5, no. 3, p. 261, 2018, doi: 10.25126/jtiik.201853692.

[30] S. SAIDAH, N. IBRAHIM, and M. H. WIDIANTO, "Pengamanan Pesan pada Steganografi Citra dengan Teknik Penyisipan Spread Spectrum," *ELKOMIKA J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron.*, vol. 7, no. 3, p. 544, 2019, doi: 10.26760/elkomika.v7i3.544.

[31] I. U. W. Mulyono, A. Susanto, M. R. F. Febrian, and G. A. Rosyida, "A Combination of Hill Cipher and LSB for Image Security," *Sci. J. Informatics*, vol. 7, no. 1, pp. 155–165, 2020, doi: 10.15294/sji.v7i1.24393.