Journal of Theoretical and Applied Information Technology
15th November 2017. Vol.95. No 21
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MP4 VIDEO STEGANOGRAPHY USING LEAST SIGNIFICANT BIT (LSB) SUBSTITUTION AND ADVANCED ENCRYPTION STANDARD (AES)

[1]PUTU ARI SRI LESTARI EKA NINGSIH, [2]GUSTI MADE ARYA SASMITA, [3]NI MADE IKA MARINI MANDENNI

[123]Departement of Information Technology, Udayana University, Bali, Indonesia

E-mail: [1]arisrilestari95@gmail.com, [2]aryasasmita@it.unud.ac.id , [3]ika_made@yahoo.com

ABSTRACT

Data transmission over the internet is vulnerable for attacked, therefore additional data security is required. Steganography is one of security protocol which used for protect the data. This research applied *steganography* using MP4 video as the host (cover media) and JPEG image as the embedded data. An AES-128 bit encryption was applied to the image before it embedded into the video. Encrypted image then stored bit-per-bit into an array. Those bits array then were embedded into the video by modifying LSB of the bytes of video. The image was embedded in the first still-encoded *sample* of the video. Sample was found by using atoms of MP4 container, those were stco/co64, stsc, stsz, stts dan mdat. Those atoms stored metadata of the video, thus could be used for mapping position of all the samples. The LSB substitution process did not touched the color spaces of the video, thus video quality can not measured using Peak Signal-to-Noise Ratio (PSNR) value. Video quality measured by a software called Qualify and the stego-video is scored 80 for its quality. That value indicated stego-video had an excellent quality. Stego-video could played well at all the common video player on any devices and there was no quality degradations or any changed as seen by human eyes. This research could contribute on increased the security of an image data when stored at the cloud storage or being transmitted over a network.

Keywords: *Steganography, Video, Sample, LSB, AES*

1. INTRODUCTION

Internet-based storage media or known as cloud storage becomes a trend of data storage today. Most Internet users take advantage of cloud storage to store various data. There are various data stored inside it, from documents, photos, audio, video, and so on. Utilization of cloud storage as a storage media supported by the evolution of the Internet and the existence of various applications that synchronize data from various devices to the owned cloud storage[1].

Many activities through the internet is vulnerable for attacked, because there is no limit to anyone who wants to access it. Therefore, no matter how small is it, data should be protected[2]. Steganography is one of security protocol that could be applied for data protection. Human visual system is not able to distinguish a few changes in some media like image, audio, or video[3].

There are many algorithms which used for doing steganography. Paper[4] describe some image-steganography methods which could be divided into 4 domains, those are image spatial domain (using Least Significant Bit substitution algorithm), image frequency domain (using Discrete Cosine Transform algorithm, Fourier Fast Transform, F5, Discrete Wavelet Transform) and adaptive steganography. One of the mentioned algorithm, Least Significant Bit (LSB) substitution is the most frequent algorithm used for data hiding. In its application, LSB substitution often combined with another method, because of its weakness, which is embedded message quite easy to find[5].

In this paper we presented result of our research about image-to-video steganography. Algorithm used in this research is LSB substitution combined with Advanced Encryption Standard (AES) as the additional security protocol. AES encryption is applied to the image, then embedded to the video. The image embedded into one of still-encoded *sample* of the video, so that process did not affect to the color spaces of the video. The term of *sample* extraction here means a process of finding the first and the last position of concerned *sample*. The finding process using atoms of MP4 container strucutre.

This research were applied in Android platform, because Android is one of popular operating system which is used by people. LSB Substitution and AES Encryption were suitable for steganogrphy at Android, because those method did not take up a lot of resource, so that suitable for smartphone which more lack of processing and memory than a desktop computer. This research was expected to be an alternative for image security on mobile devices, especially on Android devices. This could contribute on increased the security of an image data when stored at the cloud storage or being transmitted over a network.

The rest of the paper is organized as follows. Section 2 presents some of the previous work about LSB substitution in steganography. Section 3 introduces an overview of the LSB Substitution, AES Encryption, MP4 Container Structure, *sample* extraction process and our proposed steganography scheme. Section 4 presents about the experimental result and analyzed it. Section 5 presents the conclusions and further work of this research.

2. RELATED WORK

Research about steganography which applied LSB substitution method had been being the subject of research in several times. In 2011, Ramalingam built a desktop application which applied LSB substitutionof steganography. The hidden data is text data, otherwise, host media which used here is an AVI-video format. Developed application also can be used in many desktop platforms[6].

In 2012, Swathi et al did a research about steganography by using LSB substitution method and Different Polynomial Equation. Host media that hides data is a video with AVI container format. This research focused on hiding data on a specific video frame in a specific location. Location of hidden data can be determined with Different Polynomial Equation[7].

In 2013 Yadav et al proposed a scheme about secure of video steganography by using LSB substitution technique, symmetric encryption, and sequential encoding. The proposed scheme was hiding the video stream in host media video stream. Each frame on the secret video divided into multiple frames then converted become binary 8 bits and encrypted by XOR with a secret key. The video that used for host media and inserted video using format container AVI. The result showed there is no distortion which showed on media cover and quality of the secret video which extracted from media cover can be accepted perseptually[8].

In 2014, Tew and Wong reviewed about steganography or hiding data on video which compressed with H264/AVC scheme. Hiding data on H264/AVC media video became a complex process because of compression H264/AVC using so many procedures. The locations that possible to become the place of inserting data in H264/AVC video are Inter Frame and Intra Frame Prediction, DCT Koeficient, Parameter Quantization, Variable Length Coding and Binary Arithmetic Coding. Each inserting location has its own difficulty and complexity of computational[9].

In 2015, Jeswani et al proposed a video steganography scheme on android platform to secure communication via MMS. The video used as media cover, meanwhile the secret message which is inserted can be a text, image, or audio. Inserting secret data to media cover done using Pixel Differencing Value (PDV) method and LSB substitution.Videos that have been inserted messages directly generated become MMS and ready to sent via communication network[10].

All those previous researches become the based for this research. The difference between this research and all the previous researches lies in the way the data was embedded. At this research, the image data was embedded into and sample of the video, which the *sample* was found by using atoms of container of MP4 video.

3. PROPOSED SYSTEM

This research applied in Android platform, and the inputs used MP4 video format (.mp4 extension) as the host (cover media) and JPEG image (.jpg extension) as the secret data which embedded into the host. The image embedded into the video using LSB substitution algorithm, which is encrypted first using AES encryption.

The image was embedded into the first *sample* of the video. *Sample* is obtained by mapping all the MP4 video's atoms. MP4 video has atoms which its arrangement is almost constant, although generated from different devices. It makes a possiblity for the mapping process to get a *sample* from the video.

3.1 Least Significant Bit (LSB) Substitution

Least Significant Bit substitution is a commonly used algorithm in steganography. The process of bit substitution means changing LSB bit of host media with bits of data (secret message). There are no visual differences (as seen with human eyes) between embedded and original media because the changes are very slight[11]. This substitution technique works well for image, audio, and video steganography.

An example of bit substitution is presented below. For instance, a message "A" is embedded into three pixels (there is no compression for this condition). The original data raster for three pixels (nine bytes) is as follows.

00100111	11101001	11001000
00100111	11100100	11010001
11001000	00100011	11101001

To store each message "A" in low order bit plane of the raster data, it is necessary to obtain an 8-bit presentation of character "A". Character is represented by the number 65, so that the equivalent 8-bit binary representation is '01000001'. Each of 8 bits is used to represent charcter "A" is then stored in the low order bit of one byte of raster data. Thus, to store a single character "A", 8 bytes of raster data are consumed. This leads to a limit of embeddable information of size to a host media[12].

The process of embedding character "A" is done as follows. The first bit of binary representation of "A" is 0, which will be stored into the first-byte raster data, that is 000100111. We need to set the low order bit of the first byte to zero. This is accomplished by AND-ing with the mask 11111110 (0xFE). The result from this AND-ing process is:

```
  00100111
  11111110
--------------- AND
  00100110
```

The next process is OR-ing the byte which resulted from AND-ing process with one bit of binary representation of "A". This bit value is 0, thus OR-ing with the byte in order as follows:

```
  00100110
  00000000
--------------- OR
  00100110
```

The result of OR-ing process did not cause any changes to the byte data. But, due to LSB substitution principle, the resulted byte data has been changed if comparing to the original byte. All 8-byte raster data which has been embedded with character "A" is:

0010011**0**	1110100**1**	1100100**0**
0010011**0**	1110010**0**	1101000**0**
1100100**0**	0010001**1**	11101001

There are some bytes data showing differences between the original byte an embedded byte, but some bytes still in the same. This case happened at the fourth byte 0010011**0**, where the original byte formed 0010011**1**. Although several bytes are the same as the original byte, the low order bit still contains part of the message bit. At the time of confidential data retrieval, there is conformity between the embedded data and the retrieved data.

3.2 Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) is one of cryptography algorithm which is published by National Institute of Standard and Technology (NIST) at 2001. This algorithm is a part of symmetric encryption which operated in a 4x4 array-byte, that called a state[13]. A state could be encrypted using three kinds of AES cryptographic key. AES consist of three key types, those are 128, 192, and 256-bit.

The state could be encrypted and decrypted with applying four-transformation in a certain number of rounds (10, 12, and 14 rounds). The number of rounds depending on the chosen key, where 10 rounds assign for 128-bit key, 12 rounds assign for 192-bit key, and 14 rounds assign for 256-bit key.

The first round (Nr = 0) consist of these following phases:

a) Key Expansion: round keys are derived from the cipher key by using a key expansion algorithm
b) Initial Round: combining each byte of the state with round key using bitwise operation, where consist of AddRoundKey.

The other round (Nr = 1 until Nr-1) repeatedly perform these following transformation:

a) SubBytes transformation: nonlinear substitution process using S-Box. Each of byte is substituted with another byte on the basis of S-Box.
b) ShiftRows transformation: cyclic shifting operation that rotates the rows (except the first row) of the state with different numbers of byte (offset).
c) MixColumn transformation: mixing process that mix the byte in each column by

multiplying the state with polynomial modulo X^4+1.

d) AddRoundKey transformation: transformation which contains XOR operation that adds the round key to the result of the MixColumn transformation.

3.3 MP4 Container

MP4 is one of the most frequently used video formats on any devices nowadays. MP4 becomes a container for MPEG-4 sub-section 14 and H624 / AVC codecs. MP4 container based on ISO/IEC 14496-12 Information Technology - Coding of audio-visual objects Part 12: ISO based media file format. The MP4 Container is derived from the QuickTime container which is developed by Apple.

This container holds various data streams. The structure of physical data in the MP4 container is shown in Table 1. All data streams are grouped in containers called atoms. Each atom can be identified by its type and length[14]. An atom can stand on its own or contain another atom. These atoms store information such as video metadata, number of *samples*, video or audio data positions, and so on.

Table 1: Container MP4 Constructing Atoms

MP4					
1st hierarchy	2nd hierarchy	3rd hierarchy	4th hierarchy	5th hierarchy	6th hierarchy
Moov	mdhd				
	trak /video	tkhd			
		mdia	mdhd		
			hdlr		
			minf	vmhd	
				dinf	dref
				stbl	stsd
					stts
					stss
					stsc
					stsz
					stco/ co64
	trak /audio	tkhd			
		mdia	mdhd		
			hdlr		
			minf	smhd	
				dinf	dref
				stbl	stsd
					stts
					stss
					stsc
					stsz
					stco/ co64
	udta				
mdat					
free					
ftpy					

The video consists of video data and audio data. The video and audio data are mutually interleaved in the mdat atoms. Video and audio data are stored in a form of data called chunk [15]. A chunk can consist of one or several *samples* (frames), so each chunk can have different sizes and *sample* numbers. The interleaved data structure on the mdat atom is shown in Figure 1.

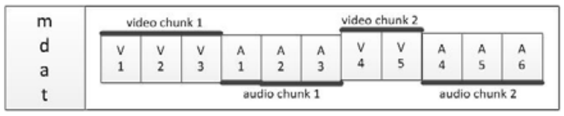


Figure 1: Ilustration of Samples in `mdat` Atom

There are several atoms related to this research. The atoms are `moov`, `mdat`, `stbl`, `stss`, `stsc`, `stsz` and `stco` / `co64` [15]. Atom `moov` encapsulates video metadata and data structures that exist in mdat atoms. In the `moov` atom, there are other atoms as shown in Table 2, where one of them is a `stbl` atom. The Atom `stbl` encapsulates the information table about individual stream *sample*s. The `stbl` atom consists of six atoms, some of which are related to steganography are `stss`, `stsc`, `stsz` and `stco` / `co64`. The `stss` atom stores information about the time length for each *sample*. The `stsc` atom stores information about all *sample*s in the whole chunk. The `stsz` atom stores information about the size of each *sample*. The `stco` (32-bit) /`co64` (64-bit) atom stores information about the initial offset of all chunks.

3.4 Qualify

Qualify is a licensed software used to analyze video and audio quality. Qualify is developed and traded by the DekTec company since 2012. This software runs on the entire Windows operating system on desktop computers [16].

There are three main modules in the Qualify software. The three modules are Video Elementary Stream Analyzer (VES) module, Video Quality Analyzer (VQA) and Audio Analysis [16]. The VES module is used for deep MPEG video (MPEG-1/2 and H.264 / AVC) analysis. The VQA module is used to analyze the video and audio asset quality. Audio Analysis module is used to analyze encoded audio level, clipping and muting.

The VQA module can be used to measure the quality of the video, to see how the quality of the video when viewed frame-by-frame. The result of this quality measurement is the distribution percentage of the average quality index. The numbers showing the average quality index are given a range of 0-100. The range of values is categorized into groups, where each category shows the video quality. Table 2 shows the range of values and categories present in the VQA module.

Table 2: Range of Values and Categories

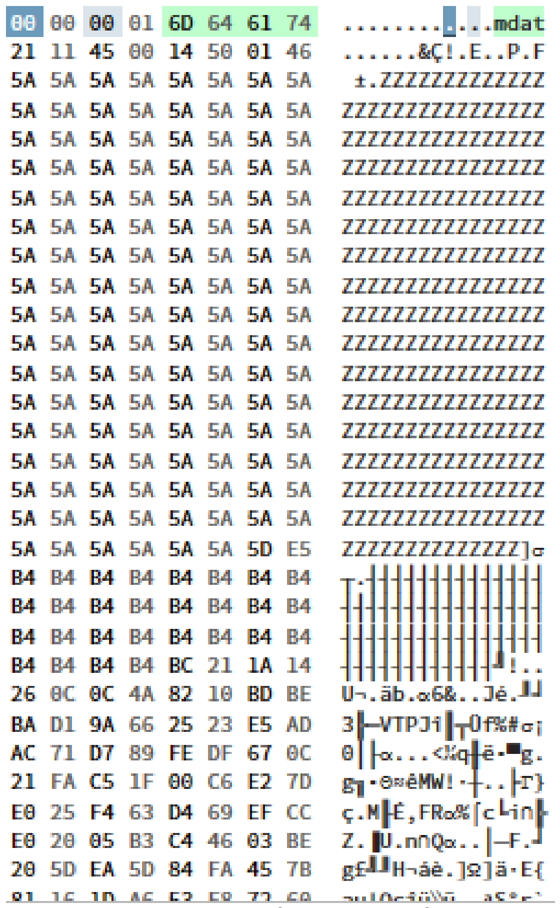
Average Quality Index (Distribution %)	Category	Explanation
0-19	BAD	At this category, video quality is very poor
20-39	POOR	At this category, video quality is pretty bad
40-59	FAIR	At this category, video quality is pretty good
60-79	GOOD	At this category, video quality is good
80-100	EXCL	At this category, video quality is very good (excellent)

Score of average quality index is generated by Qualify through measurements of several parameters. These parameters include video frame resolution, image fineness level at each frame, blurring rate on each frame, corrupted image, and others[16]. Measurements are performed on each parameter in each video frame, then the value is averaged to obtain the average quality index of a video.

3.5 *Sample* Extraction Process

The *sample* extraction process used several boxes on the MP4 container. This process does not use entire boxes on the MP4 container, but only uses four boxes, those are stco / co64, stsc, stsz, stts and mdat. This mapping generates the starting and ending positions of each *sample* in the video array-bytes.

Each steps of getting all the video *samples* is described as follows. Step one in the *sample* extraction process is to convert the video into array-bytes. Figure 1 shows an example of a video that has been converted into array-bytes. Figure 1 is taken from a software that converts the video into a set of bytes.



```
00 00 00 01 6D 64 61 74   .........  ...mdat
21 11 45 00 14 50 01 46   ......&Ç!.E..P.F
5A 5A 5A 5A 5A 5A 5A 5A    ±.ZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5A 5A   ZZZZZZZZZZZZZZZZ
5A 5A 5A 5A 5A 5A 5D E5   ZZZZZZZZZZZZZZ]σ
B4 B4 B4 B4 B4 B4 B4 B4   ╤.┤┤┤┤┤┤┤┤┤┤┤┤┤┤
B4 B4 B4 B4 B4 B4 B4 B4   ┤┤┤┤┤┤┤┤┤┤┤┤┤┤┤┤
B4 B4 B4 B4 B4 B4 B4 B4   ┤┤┤┤┤┤┤┤┤┤┤┤┤┤┤┤
B4 B4 B4 B4 BC 21 1A 14   ┤┤┤┤┤┤┤┤┤┤┤┤┘!..
26 0C 0C 4A 82 10 BD BE   U¬.äb.α6&..Jé.╜╛
BA D1 9A 66 25 23 E5 AD   3║─VTPJí║╥0f%#σ¡
AC 71 D7 89 FE DF 67 0C   0│├α...<¼q╫ë·■g.
21 FA C5 1F 00 C6 E2 7D   g╖·Θ≈êMW!·┼..├Γ}
E0 25 F4 63 D4 69 EF CC   ç.M╟Ê,FRα%⌠c└in╠
E0 20 05 B3 C4 46 03 BE   Z.▐U.n∩Qα..│─F.╛
20 5D EA 5D 84 FA 45 7B   g£╜╜H¬áè.]Ω]ä·E{
```

Figure 2: Example of Array-Bytes of Video

Videos that have become bytes arrays then processed for getting required atoms. The first required atom is stco/co64. This atom contains information about the number and offset of each chunk on the video. This atom provides information about number of chunks in the video, the starting position, and size of each chunk. The initial offset and end of the chunk can be the foundation in the *sample* search.

The next step is to find the stsc atom. This atom stores information about the number of *samples* in each chunk. Each video chunk can store various number of *samples*, so it needs information about the number of *samples* in each. The information obtained from the stsc atom can be a basis for mapping the number of *samples* that present in the video.

The third atom which required in this steganography process is stsz atom. This atom stores information about the size of each *sample*. Two previous atom, stco/co64 and stsc have produced chunk position information and the number of *samples* in each chunk. Thus, the information from the stsz atom can complete all the information used to find the position of each *sample*.

The mapping process began by listing the chunk first, then listing the number of *samples* in each chunk. The offset of each *sample* can be obtained by using the size data of each *sample* obtained from the stsz atom. Thus, the starting and ending position of each *sample* can be obtained. The *sample* extraction process is completed up to this stage.

Offset of each *sample* that has been obtained becomes the place of insertion of image bits. Offset is located in the mdat atom, where this atom contains video data media, in this case, compressed video and audio.

3.6 Embedding Phase

Embedding is a process of image insertion into a host video. This process requires some inputs like video with MP4 format, the image with JPEG format, and keyword along 16 characters. This phase is illustrated by the Figure 3.

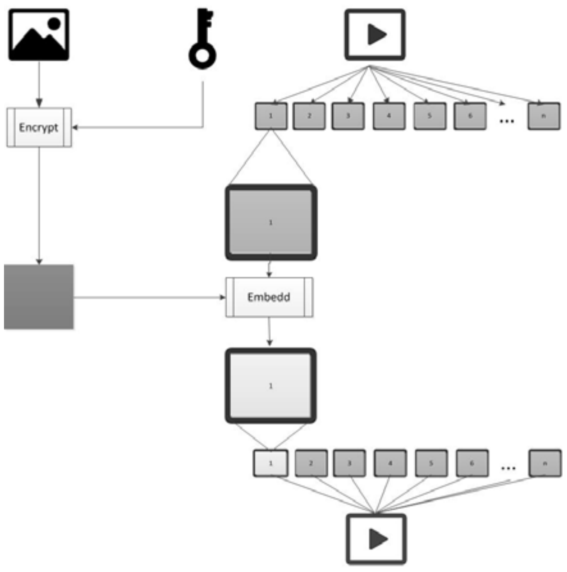


Figure 3: Embedding Process

The image insertion process is described on the following steps:

a) Extracting video *samples*. Extraction process using stsc, stsz and stco/co64 atoms. The result of this extraction process is the starting position and end position of each *sample*.
b) Recording the start and end position of all *samples*.

c) Image converted into byte-array.
d) This byte-array then encrypted using a 16-character (16-bytes) key.
e) The encrypted byte-array then embedded bit per bit into the first *sample* of the video. Embedding process applied LSB substitution principle by replacing the low order bit of bytes in the array-byte.
f) *Sample*s that have been inserted image then put together with other *sample*s
g) Video *sample* data that has been embedded with the image then saved back into one-whole MP4 video.

3.7 Extraction Phase

Embedding is a process of image insertion into a host video. This process requires some inputs like video with MP4 format, the image with JPEG format, and keyword along 16 characters. This phase is illustrated by the Figure 4.

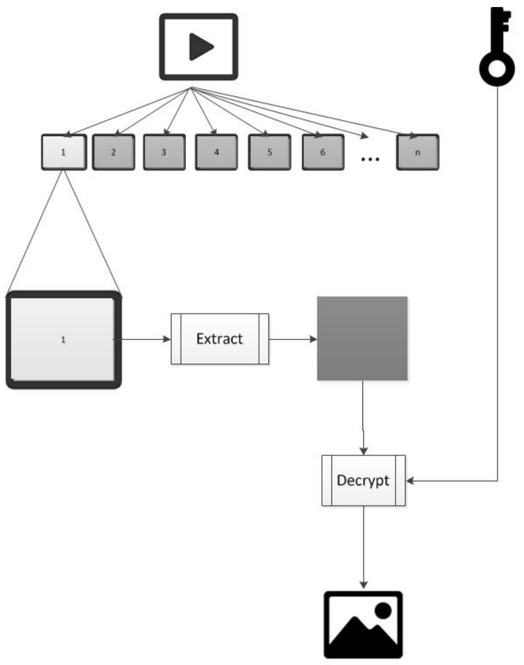


Figure 4: Extraction Process

The extraction process is described on the following steps:

a) Extracting video *sample*s. Extraction process using stsc, stsz and stco/co64 atoms. The result of this extraction process is the starting position and end position of each *sample*.
b) Recording the start and end position of all *sample*s.
c) Each low order bit of every byte of the video is taken and collected. These bits are stored into a bit-array.
d) Each 8-bit data in the bit-array then converted into a form of byte data. All the converted bytes then stored into a byte-array.
e) After the byte-array is completed, it is decrypted by using a 16-character (16-byte) key.
f) Decrypted byte-array data then saved to the storage in a form of JPEG image.

4. RESULT AND ANALYSIS

The experimental process used an MP4 video and a JPEG image. MP4 video was generated from a smartphone with a resolution of 640x480. The image that was used is a Udayana University logo with 2KB of size. This research was implemented on a smartphone which used Android as the operating system.

There are two factors that were considered in this experiment. First factor was the image embedding process. Second factor was image extraction process. Beside that, there are two aspect that were analyzed. Two aspects that were analyzed from the results of this experiment were quality of stego-video and compared it with original video, and the quality of image extraction results and compared it with the original image.

Image insertion process produced a stego-video which can be played well on various video player applications. This indicates that embedding process on a still-encoded *sample* can be performed without damaging the original video. Comparison between the original video and stego-video is presented at Figure 5 below.

Stego-video could be extracted to regain the image. The image extraction process produced an image that was identic to the original image. There is no deterioration in the quality of the extracted image. Comparison between the original image and extracted image is presented at Figure 5 below.

Original Video	Original Image

Stego-Video	Extracted Image

Figure 5: Comparison Between Original Video and Stego-Video, and Between Original Image and Extracted Image

Many researches usually used Peak Signal-to-Noise Ratio (PSNR) value to evaluate stego-video, but in this research, PSNR measurement is not suitable to be applied. This is because the image insertion process into the video is not touching the color spaces of the video frames, whereas the PSNR values are obtained by looking into the color space of each frame of the video. If PSNR measurement was still forced, it will get a value of 100dB for stego-video, which means stego-video is considered 100% identical to the original video. This is deviating the purpose of analyzing because there are some changes in the video due to the process of image insertion. This causes the analysis proper did through a software which gives a measurement quality tools using parameters other than color space of video. One of them is a software called Qualify.

Quality measurement process using Qualify aims to assess the quality of the video because the LSB substitution in still-encoded video *samples* can cause damage when played, few of frames become grayish, broken pixels, the appearance of lines that indicate damage videos, and so on. Video quality measurement using Qualify software is shown in Figure 6 at the end of this paper.

Result of the mesurement by Qualify showed the stego-video had a quality with a value 80. This value indicated the *stego-video* had an excellent quality, so it can be concluded that the stego-video which was produced by the proposed method has a very good quality.

5. CONCLUSIONS

This research proposed a steganography scheme with a combination of LSB substitution and additional AES encryption, which was applied on the Android platform. This research was used MP4 video and JPEG image as the inputs, where the video be the media cover and the image be the embedded secret message. The results of experiment and analyses showed that stego-video did not suffer damage as seen by the human eye, and video could be played by the video player on the smartphones or desktop computers. The same results were also obtained when extracting images from stego-video, where the image was not damaged and identical with the original image. Stego-video quality is measured with Qualify software. Stego-video has excellent quality with a value of 80. This research could be developed by embedding data in more than one *sample*, with the number of images inserted more than one image. This research would be applied to various operating systems in many devices, thus will be more useful.

6. REFERENCES

[1] C. H. Wang and S. L. Lin, "Why are People Willing to Pay for Cloud Storage Service?", In *IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, 2016, pp.1-6.

[2] I. G. N. A. Jayarana, A. A. K. A Cahyawan, and G. M. A. Sasmita, "Dynamic Mobile Token for Web Security using MD5 and One Time Password Method", *International Journal of Computer Applications*, Vol.55, No.2, 2012, pp.1-6.

[3] R. J. Mstafa and K. M. Elleithy, "A Highly Secure Video Steganography using Hamming Code (7,4)", In *System, Applications and Technology Conference (LISAT)*, 2014, pp.1-6.

[4] C. Abbas, *et al*. "Digital image steganography: Survey and analysis of current methods", *Signal processing*, Vol.90, No.3, 2010, pp.727-752.

[5] M. Bashardoost, G. B. Sulong, and P. Gerami, "Enhanced LSB image Steganography method by using knight Tour algorithm, Vigenere Encryption and LZW compression", *IJCSI International Journal of Computer Science Issues*, Vol.10, No.2, 2013, pp.221-227.

[6] R. Mritha, "Stego machine–video steganography using modified LSB algorithm", *World Academy of Science, Engineering and Technology*. Vol.74, 2011, pp.502-505.

[7] A. Swathi, and S. Jilani, "Video Steganography by LSB Substitution Using Different Polynomial Equations", *International Journal Of Computational Engineering Research,* Vol.2, No.5, 2012, pp.1620-1623.

[8] Y. Pooja, N. Mishra, and S. Sharma, "A secure video steganography with encryption based on LSB technique", In *2013 IEEE International Conference on Computational*

Intelligence and Computing Research (ICCIC), 2013, pp.1-5.

[9] Tew, Yiqi, and K. Wong, "An overview of information hiding in H. 264/AVC compressed video", *IEEE transactions on circuits and systems for video technology,* Vol.24, No.2, 2014, pp.305-319.

[10] Jeswani, V. Ramesh, S. Kulkarni, and M. Ingle, "Android Application Development for Secure Data Transmission using Steganography," *Transactions on Networks and Communications* Vol.3, No.3, 2015, p.39.

[11] S. Alam, *et al*, "Analysis of Modified LSB Approaches of Hiding Information in Digital Images", In *5th International Conference on Computational Intelligence and Communication Networks IEEE*, 2013, pp. 280-285.

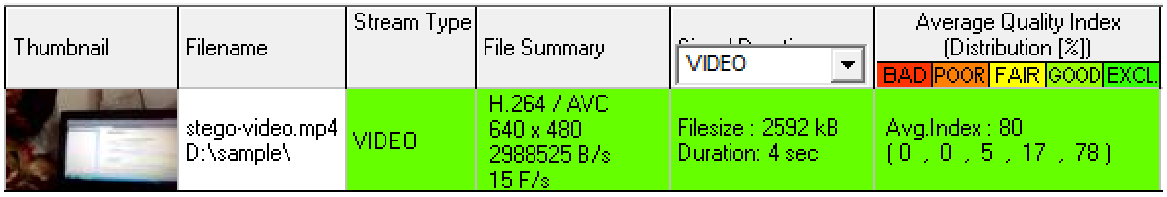
[12] A. K. Hmood, *et al*, "An overview on hiding information technique in images", *Journal of Applied Science*, Vol.10, 2010, pp. 2094-2100.

[13] A. Abdulgader, M. Ismail, N. Zainal, and T. Idbeaa, "Enhancement of AES Algorithm Based on Chaotic Maps and Shift Operation for Image Encryption", *Journal of Theoretical and Applied Information Technology*, Vol.71, No.1, 2015, pp.1-12.

[14] T. Gloe, A. Fischer, and M. Kirchner, "Forensic analysis of video file formats", *Digital Investigation*, Vol.11, 2014, pp.68-76.

[15] M. Jokay, "The design of steganographic system based on the internal MP4 file structures", *International Journal of Computers and Communication*, Vol.5, No.4, 2011, pp.207-214.

[16] DekTec, 2012, "Qualify Software", DekTec, Hilversum, Netherlands.



Thumbnail	Filename	Stream Type	File Summary	VIDEO	Average Quality Index (Distribution [%]) BAD POOR FAIR GOOD EXCL
	stego-video.mp4 D:\sample\	VIDEO	H.264 / AVC 640 x 480 2988525 B/s 15 F/s	Filesize : 2592 kB Duration: 4 sec	Avg.Index : 80 (0 , 0 , 5 , 17 , 78)

Figure 6: Video Quality Measurement of Video Using Qualify