Data Hiding in MP4 Video Container based on Subtitle Track

ChuanSheng Chan*, KokSheik Wong*, Imdad Maungmuang†,
* School of Information Technology, Monash University Malaysia, Malaysia.
† Faculty of Computer Science and Information Technology, University of Malaya, Malaysia.

***Abstract*—This paper proposes a data hiding method in MP4 container format. Specifically, the synchronization between subtitle and audio-video tracks is exploited to hide data. The time scale is first scaled, and the sample duration pair is modified to hide data. The proposed method is able to hide data reversibly when the payload size is relative small, and it switches to the irreversible mode to offer higher payload. Although synchronization between audio-video and subtitle tracks are manipulated, the delay or ahead in displaying subtitle is imperceptible. The filesize of the processed MP4 file is also completely preserved. Subjective evaluations are carried out to verify the basic performance of the proposed method.**

I. Introduction

Data hiding is the art and science of inserting a payload into a content [1]. A payload is a specific piece of data encoded in a binary representation and it can be derived from the host, external to the host or the mixture of both, depending on the application. Owing to the advancement of internet technologies, cloud computing and the ever increasing adoption of mobile devices, information are significantly more accessible nowadays. One of the most popular multimedia content circulated in the web is video. The reason is mainly due to the increased utilization of video streaming sites such as YouTube, Vimeo, DailyMotion etc. by the internet users, with YouTube leading the pack by being the most visited video sharing website. According to statistics [2], Youtube is the second mostly visited website as of March 2018.

While video is able to provide simultaneous and continuous audio-visual stimulation, subtitle, which is the text-based descriptions of the dialog or commentary of a video, is able to provide different experience in watching a video. For example, someone with hearing impairment will rely on the subtitle as supplementary information in order to understand the video. By using sub-title, non-native speaker can enjoy the original production of foreign movie without dubbing the movie into the local language. Another closely related technology is called closed-captioning, where it also includes non-dialog audio as well such as "(panting)", "(bird chirping)", etc.

In order to efficiently deliver multiple tracks such as encoded video (e.g., H.265/HEVC, VP9), audio streams (e.g., MP3, AAC), and subtitle / closed-caption, a container format is utilized. A container also consists of metadata that governs the playback and synchronisation of the tracks. Some examples of video container format include MP4, MKV, FLV, WMV, MOV, and WebM. Despite the variety of choices, 69% of the web videos and 58% of the mobile videos use MP4 container [3]. Being the preferred container, MP4 offers various features [4]. For instance, MP4 is supported by most platforms and major media players, where there is comparably less quality loss with higher degree of compression. Besides, it can also store data types other than video and audio, such as object descriptors, scene descriptors and other object oriented file structures and MPEG features.

Given the ease of capturing and storing a video nowadays, users and administrators wish to have some mechanism for them to label videos as well as linking related videos. Streaming companies may want to insert a watermark in each video uploaded to their platform in order to trace back the illegally downloaded and distributed videos elsewhere on the internet [5]. The extra features should be accomplished by introducing minimal or zero distortion to the host content and as transparent as possible to prevent reverse engineering.

While various techniques were put forward to insert data into the image, audio and video, only a handful of techniques are designed to insert data into multimedia container format. Specifically, Jokay [6] hides data into MP4 container by exploiting odd/even parity of GOP structure. Cosimo [7] introduced a steganographic application called OpenPuff Tool, which manipulates selected flags in MP4 container to hide data. However, the method used in this tool [7] has been defeated by the steganalysis proposed by Sloan et al [8]. Another steganographic tool [9] injects True-Crypt container (in which itself is an encryption container) within a MP4 container to form a hybrid MP4/TrueCrypt container file. Specifically, the True-Crypt container is inserted into the mdat box and the stco chunk offset is modified to point to the position of actual media data, i.e., audio or video. Recently, MaungMaung [10] propose to hide the perceptual hash of the video frames into audio samples and the audio hash vector is embedded into synchronization information into MP4 container, where both the video and audio tracks are packed within a MP4 container.

This work aims to put forward a method to hide data into MP4 container by exploiting the subtitle track. To the best of our knowledge, this is the first of its kind, although there are some existing works that hide subtitles into the video track, including [1]. Unlike the synchronization between the audio and video tracks, the error in synchronization between sub-title and audio-video tracks are less noticeable. Although it is recommendation that each subtitle should appear for at least 0.3 seconds per word [11], there is no upper bound to the display duration. Therefore, the display duration can be exploited for data hiding purpose, as long as the duration of each subtitle does not overflow to the display time of its next subtitle. In this work, the time scale is first scaled, and the sample-duration pairs are modified to hide data. Experiments are carried to verified the basic performance of the proposed method. The proposed method is able to insert data reversibly when the payload size is small, and it switches to the irreversible mode to offer higher payload.

II. Preliminary

MPEG-4 Part 14, or in short, MP4, is a digital multimedia container format developed by Motion Picture Expert Group (MPEG) and standardised by International Organization for Standardization (ISO). The file format specification is based on QuickTime file format which is developed by Apple Inc. The only official and the most common filename extension for MPEG-4 Part 14 is *.mp4*, although there are also several other extensions which serve different purposes. For example, *m4a* which only stores audio stream, *m4b* which stores audio book file, *m4r* stores ringtone file in iPhone, etc. An MP4 container can hold video stream encoded in MPEG-H Part 2 (H.265/HEVC), MPEG-4 Part 10 (H.264/AVC), MPEG-4 Part 2, etc. as well as audio stream encoded in MPEG-1 (Layers I, II, III), MPEG-2, AC-3, subtitles, etc.

The container is composed of objects called boxes as shown in Fig. 1. In detail, each box in the structure is made up of three parts, namely size, type and data. In a typical 32-bit MP4 file, size and type each occupies 4 bytes. The first four bytes store the size of the entire box, the next following four bytes form the characters code identifier of the particular box and data assumes a variable length. In most cases, the term *header* refers to the first eight bytes, which is the combination of size and type, and it is treated as an unique identifier of each box. Each box can be categorised into parent / container and child box. A parent / container box is the one which contains other boxes (sub-boxes) in its data section and a child box is the one which contains information. Despite the existence of various top-level boxes, only three top-level boxes are housing the primary information. They are *ftyp* (file type box) that stores the identification information of the MP4 file, *moov* (movie header box) that stores the metadata of the streams, and *mdat* (movie data box) that stores the actual media data. Specifically, *moov* independently stores the metadata for each multimedia streams / track (i.e., audio, video, subtitle etc.) available in the container.

A. Streams synchronization in MP4

There are three important entities in each track which are utilized as the synchronization information. The first entity τ is utilized for frame timing during playback. τ exists within *mdhd* (media header box), which is in turn contained within *moov*. The other two entities lie within another box called *stts* (decoding time to sample box). In this particular box, there is a table which contains one or more 2-tuple (sample count s, duration δ) entry. All entries in the table determine the number of samples (i.e., video frame, audio sample or subtitle sample)

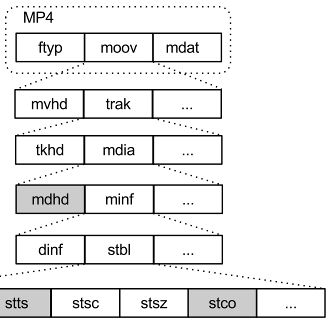


Fig. 1: The general structure of an MP4 box [10].

and how long (in unit of timescale specified in τ) each sample will be displayed/played. During the playback of the track, the duration of an entry will be converted into unit per second by multiplying a time base. For example, if an entry has the value of $(s = 1, \delta = 20)$ and the timescale for the track is 60, it implies that the particular sample will be displayed/played for $(1/60 \times 20) = 0.33$ secs.

III. Data hiding in Subtitle track

In this work, we exploit the synchronization between the subtitle track and the audio-video tracks to hide data. It is assumed that the subtitle track is present in the MP4 file and contains some texts. In other words, the MP4 file chosen as the host should be soft-subbed, i.e., a type of video that contains subtitles where its display can be turned on or off in the media player, instead of a hard-subbed video where the subtitle texts are burnt into the frames of the video.

A. Pre-processing Time Coordinate System

Before embedding the data into the host, we will modify τ and δ while maintaining the original audio-video-subtitle synchronization. Consider an instance of $\tau = 30$ and $\delta = 1$ which implies that a unit of duration is $(1/30)$ or $0.0\dot{3}$ sec. If we update the values to $\tau = 60$ and $\delta = 2$, the ratio is still maintained because a unit of duration now represents (1/60) or $0.1\dot{6}$ sec. but since the value of the duration is doubled, the final duration is still (1/30) or $0.0\dot{3}$ sec. Similarly, we can update the value of τ to a much bigger value (e.g., $\tau = 90, 120, 150, \cdots$) and update the duration δ (e.g., $3, 4, 5, \cdots$) with the ratio to keep the synchronization process unchanged. Although τ and δ are capped at 2147483647 (i.e., 4-byte signed integer), they cannot be set at the said maximum value as considered in [10]. It is because the display duration of a subtitle is significantly longer than that of a audio sample. Hence, when the duration of a subtitle is scaled accordingly to maintain synchronization, an error will be raised when there is an attempt to write a value of δ which exceeds the permissible maximum. In order to minimize the chances of this error, the system needs to be assigned a smaller τ value as the constant to reduce the multiplier.

TABLE I: Summary of test videos

Video Sequence	Total Frames	Total Subtitles
TED [13]	20248	315
Bloomberg [15]	8759	77
National Geographic [16]	5622	50
Fibonacci [17]	16709	272
Car [18]	10781	274

TABLE II: Length of payload that can be hidden (bits).

Video	Irreversible	Reversible
TED [13]	5048	215
Bloomberg [15]	776	10
National Geography [16]	504	8
Fibonacci	2176	17
Car	2192	15

TABLE III: Mean square error when treating least significant byte as an integer in the range of $[0, 255]$ for data hiding.

Video	MSE
TED	392.45
Bloomberg	1547.86
National Geography	998.62
Fibonacci	10.67
Car	10.22

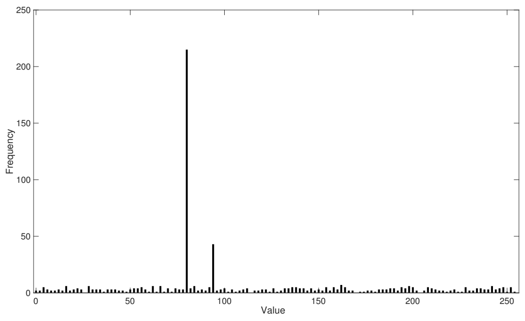


Fig. 2: Distribution of δ for the test video [13].

Due to the the theoretical limit of average human eyes (i.e., see up to 1000 FPS) and the fact that two modes proposed are making use of the least significant bytes, a formula to update τ is set as $\lceil 255000/\tau_0 \rceil$, where τ_0 is the original τ value in the MP4 container. Ultimately, the reason of increasing τ to a huge value is to ensure that the perceptual difference of the subtitle synchronization before and after the data hiding is minimal so that it is simply unnoticeable by comparing the videos side by side. From a different perspective, if τ is a small value (e.g., $\tau = 1$), a unit of duration is $(1/1) = 1$ sec. Now if we increase the duration of a sample merely by 1, we are increasing the duration of it by 1 sec., which will in turn produce a very obvious difference.

B. Data Hiding using (s, δ) pairs

The underlying data hiding mechanism is to modify the least significant byte of δ, which is represented an 4-byte signed integer. For example, given a duration with the value of 614_{10}, it is equivalent binary representation is 00000000 00000000 00000010 01100110_2. Hence only the least significant byte (i.e., 01100110) will be manipulated based on the payload, where 8 bits can be hidden. In this work, the entire byte is replaced by a byte-segment of the payload. Since the actual display time, in unit of second, of a specific subtitle entry is computed based on the equation below

$$\frac{1}{\tau} \times \delta, \tag{1}$$

larger τ will lead to less out-of-sync between the subtitle track and the audio-video tracks when the least significant byte of δ is consistently exploited for data hiding.

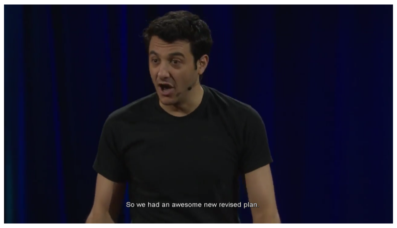
Further analysis reveal that the least significant byte of δ tend to cluster. An example is shown in Fig. 2, where the peak appears at the value of 28, with many empty bins. Therefore, when the payload size is smaller than the frequency of the peak bin, it is suitable to deploy the histogram shifting technique [12] to manipulate the least significant byte of δ to reversibly hide data.

IV. Experiment

Due to the lack of suitable *soft-subbed* MP4 files available on the internet, the videos used in the experiments are produced by re-encoding a standalone MP4 file at 30 frames per second and its corresponding SRT file with a video encoding software called *HandBrake* [14]. For experiment purpose, we set $\tau = \lceil 255000/\tau_0 \rceil$ and since $\tau_0 = 90000$, $\tau = 270000$. The test videos used in all our experiments are detailed in Table I. A representative frame of each video, along with the sub-title displayed, are shown in Fig. 3. For each movie, the frame resolution is 1280×720 pixels encoded with H.264, and the audio track is encoded with AAC. It is verified that all hidden data can be extracted, and the filesize remain unchanged. It is also confirmed that, the display and duration of the subtitle texts appear natural, and there is no noticeable differences between the original and processed videos.

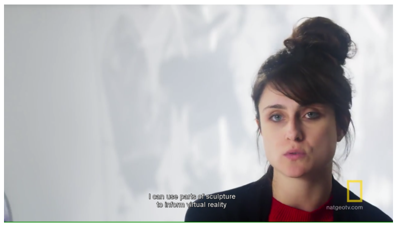
The number of bits which can be hidden into each MP4 file are recorded in Table II. Naturally, a longer video or a video with more conversations will have more subtitles, hence more bits can be hidden. On average, 0.153 bits can be hidden into each frame, which translates to 4.59 bits per second for a video at 30 frames per second. Without causing any bitstream size increment. In addition, when operating in the reversible mode through histogram shifting, the length of payload reduces significantly as recorded in Table II, although some data can still be hidden.

Since the least significant byte of the duration δ parameter for each subtitle is modified, we compute the mean square error (MSE) between the new and original durations. The results are recorded in Table III. For analysis purpose, the least significant byte of the duration parameter δ is treated as an unsigned integer in the range of $[0, 255]$. Although the MSE appears to be large in terms of integer, the actual time

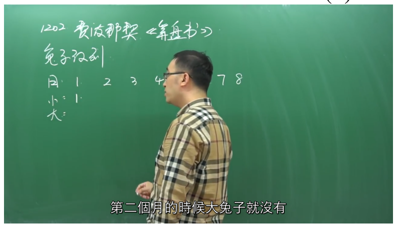


(a) TED

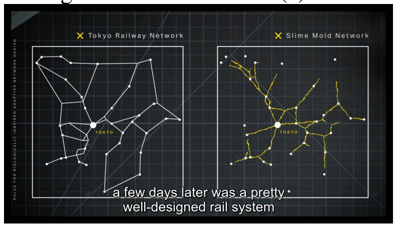
(b) Bloomberg



(c) National Geography



(d) Fibonacci



(e) Car

Fig. 3: Representative frame of each video with sub-title displayed.

difference is small. Specifically, since $\Delta\delta \in [-255, 255]$, the differences before and after data hiding, in the unit of second, will be bounded by range of

$$\left[\frac{-255}{\tau}, \frac{255}{\tau}\right] = \left[\frac{-255}{\lceil 255000/\tau_0 \rceil}, \frac{255}{\lceil 255000/\tau_0 \rceil}\right] = [-\beta, \beta], \quad (2)$$

where $\beta = 0.00094$. In other words, the difference in display duration is less than 1 mili-second, which is imperceptible to the human visual system. In fact, based on the current setting, more information can be hidden by considering more bytes of δ, as long as the recommendation of 0.3 per words is satisfied.

When compared to the conventional data hiding method [10] designed for the MP4 container format, the proposed method is able to preserve the bitstream size irregardless of the length of the video. Similar to [10], the proposed method can also inject empty subtitles to increase the (s, δ) entries so that more data can be hidden, but at the expense of bitstream size increment.

V. Conclusion

In this work, a data hiding method exploiting the sychronization between subtitle and audio-video tracks are proposed by means of manipulating the display duration of each subtitle. Data can be hidden without causing noticeable delay or ahead in displaying the subtitles while maintaining bitstream size of the MP4 container. When the length of the payload is small, the proposed method can reversibly hide data into the MP4 container format.

Our future work joint utilization of data hiding in text and the proposed method. Analysis will also be carried for closed-caption texts.

References

[1] Yiqi Tew and KokSheik Wong. An overview of information hiding in h. 264/avc compressed video. *Circuits and Systems for Video Technology, IEEE Transactions on*, 24(2):305–319, 2014.

[2] Alexa top 500 global sites. https://www.alexa.com/topsites, 2018 (accessed April 12, 2018).

[3] Jon Orlin. Survey: Mp4 is top format for web and mobile videos, 2012 (accessed April 12, 2018).

[4] Mp4 file format usage and compression techniques, 2014 (accessed April 12, 2018).

[5] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital watermarking and steganography*. Morgan Kaufmann, 2007.

[6] M Jókay. The design of a steganographic system based on the internal mp4 file structures. *International Journal of Computers and Communications*, 5, 2011.

[7] Cosimo Oliboni. Openpuff v4.00 steganography and watermarking, Jul 2012.

[8] Thomas Sloan and Julio Hernandez-Castro. Steganalysis of openpuff through atomic concatenation of mp4 flags. *Digital Investigation*, 13:15–21, 2015.

[9] TrueCrypt Foundation. Truecrypt.

[10] Imdad MaungMaung, Yiqi Tew, and KokSheik Wong. Authentication of mp4 file by perceptual hash and data hiding. *Malaysian Journal of Computer Science*, Accepted on June 2018.

[11] Subtitle guidelines. http://bbc.github.io/subtitle-guidelines/, 2018 (accessed June 7, 2018).

[12] Zhicheng Ni, Yun-Qing Shi, N. Ansari, and Wei Su. Reversible data hiding. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(3):354–362, March 2006.

[13] TED. Inside the mind of a master procrastinator — tim urban. https://www.youtube.com/watch?v=arj7oStGLkU, 2016.

[14] The HandBrake Team. Handbrake.

[15] Bloomberg. How masayoshi son became an eccentric dealmaker. https://www.youtube.com/watch?v=cDpTPrfw1mQ, 2018.

[16] National Geographic. Re-envisioning reality - tech+art — genius: Picasso. https://www.youtube.com/watch?v=T9chHEEp-0M, 2018.

[17] Yongle Li. A lecture on fibonacci series. https://www.youtube.com/watch?v=VCJsUYeuqaY, 2018.

[18] Verge Science. What self-driving cars can learn from brainless slime mold. https://www.youtube.com/watch?v=40f7_93NIgA, 2018.