INVITED PAPER *Special Section on Enriched Multimedia—Application of Multimedia Technology and Its Security—*

Lightweight Authentication for MP4 Format Container Using Subtitle Track

KokSheik WONG[†a)], **ChuanSheng CHAN**[†], ***and*** **AprilPyone MAUNGMAUNG**[††], ***Nonmembers***

SUMMARY With massive utilization of video in every aspect of our daily lives, managing videos is crucial and demanding. The rich literature of data embedding has proven its viability in managing as well as enriching videos and other multimedia contents, but conventional methods are designed to operate in the media/compression layer. In this work, the synchronization between the audio-video and subtitle tracks within an MP4 format container is manipulated to insert data. Specifically, the data are derived from the statistics of the audio samples and video frames, and it serves as the authentication data for verification purpose. When needed, the inserted authentication data can be extracted and compared against the information computed from the received audio samples and video frames. The proposed method is lightweight because simple statistics, i.e., '0' and '1' at the bit stream level, are treated as the authentication data. Furthermore, unlike conventional MP4 container format-based data insertion technique, the bit stream size remains unchanged before and after data insertion using the proposed method. The proposed authentication method can also be deployed for joint utilization with any existing authentication technique for audio / video as long as these media can be multiplexed into a single bit stream and contained within an MP4 container. Experiments are carried out to verify the basic functionality of the proposed technique as an authentication method.

key words: *authentication, synchronization, lightweight, subtitle, MP4*

1. Introduction

Video has become one of the mostly communicated content online. To put number into context, according to [1], 400 hours of video content is uploaded every minute to YouTube itself. Some of the driving factors for such phenomena include ubiquitous network environment and the availability of smart device at affordable price. Most smart devices including the entry level models allow users to capture, edit, and share high quality videos. These videos can be easily edited using smart phone or tablet, which include cropping, deletion, substitution, rearrangement of event sequences, splicing (viz., merging two or more audios/videos into one), to name a few. Although simple in nature, these basic editing techniques could have great impact to the semantic video. For example, in an audio-video recording during a confession, the original clip of "I am not guilty" can be cropped to "I am guilty" [2]. Therefore, it is crucial to be able to verify the originality of a given video. Authentication through data hiding is one of the ways in providing the capability to verify the video [3].

In the past, researchers focused on providing authentication solutions for image, video, and audio. For example, Tew et al. [4] proposed to use statistics of the coding unit (CU) sizes in HEVC video as the authentication data. This information is embedded by manipulating the CU in the next video frame, and the process is repeated for video frames. Similarly, Maung et al. [5] compute the authentication (hash) value using quantized spectral coefficients for audio stored in the advance audio coding (AAC) format. The resulting hash value of the coefficients in the current audio frame is embedded into the spectral coefficients in the next frame, and the process is repeated for all frames. The conventional techniques are designed for one specific type of content (also known as track in the context of container format). In addition, elements of the content are manipulated to encode authentication information, which could either lead to quality degradation or bit stream size increment.

Recently, Maung et al. proposed an authentication method by exploiting the synchronization information between audio and video tracks [6]. Specifically, redundant representation of the (*sample_count*, *time*) pair for video is introduced to encode authentication information, where significantly more information can be encoded by modifying the *time* tuple of each newly introduced (*sample_count*, *time*) pair. In their work, the video hash information is embedded into the audio track, while the audio hash is in turn inserted by utilizing the (*sample_count*, *time*) pairs. Although video track remains unchanged, the audio track and the audio-video synchronization information are modified to hide data, which lead to some distortion in audio and bit stream size increment.

Our previous work [7] exploits the subtitle track to hide data, where basic information about the sychronization between audio-video and subittle tracks are manipulated. Motivated by the importance of video integrity, we extend our previous work [7] to provide light weight authentication. This work makes the following contributions: (a) proposing a joint authentication technique for audio samples and video group; (b) allowing direct joint utilization with existing authentication techniques proposed for the audio and video tracks, and; (c) conduct experiments by using more videos. For the purpose of this work, we focus on non-fragmented MP4 file [8]. We consider the scenario where the MP4 file is downloaded entirely or shared via cloud storage for consumption instead of streaming.

Manuscript received April 3, 2019.
Manuscript revised August 19, 2019.
Manuscript publicized October 24, 2019.

[†]The authors are with the School of Information Technology, Monash University Malaysia, Malaysia.
[††]The author is with the Graduate School of System Design, Tokyo Metropolitan University, Hino-shi, 191–0065 Japan.
a) E-mail: wong.koksheik@monash.edu

DOI: 10.1587/transinf.2019MUI0001

The rest of the paper is structured as follows: Sect. 2 gives an overview of the MP4 container, where the syntax and parameter involved in this work are emphasized. The proposed authentication method is put forward in Sect. 3. Experiment results are discussed in Sect. 4, followed by a Discussion section in 5. Finally, Sect. 6 concludes this article.

2. Overview of MP4 Container

A container format serves as a venue for different elements such as data and metadata to coexist in a single file. Container formats are widely used in multimedia data, especially in the case of audio, image, and video. The commonly utilized multimedia containers include WAV, FLAC, MP3 and M4A for audio, and 3GP, AVI, FLV, MKV, MPEG, WebM, Ogg, and MP4 for video. These container formats provide some mechanism to interleave audio, video and subtitle (i.e., multiplexing), as well as parsing the interleaved data. They also provide synchronization and other necessary information (viz., metadata) for decoding.

MP4 [9] (MPEG-4 Part 14) container format is developed by the Motion Pictures Expert Group (MPEG) based on QuickTime file format [10], which is in turn developed by Apple Inc. It has the file extension of *.mp4*. It can hold video encoded in the format of MPEG-2 Part 2, MPEG-4 ASP, H.263, H.264/AVC, Dirac, etc., as well as audio encoded in the format of MPEG-1 (Layers I, II, III), MPEG-2, AC-3, Vorbis, AAC, HE-AAC, etc. As for subtitle, the arguably most commonly used format is SubRip Text (*.srt*). It contains groups of plain texts separated by blank lines, each having four parts, namely: the sequence number, start and end time for the texts in the current group, the actual texts for display, and the empty line [11]. When combined into the subtitle track within the MP4 container, each 3-triplets will appear in the format of 3GPP text access units. A 3GPP text access unit contains data from exactly one text sample. While each text sample consists of a string that represents the characters (including space and hence a sentence) to be displayed and, optionally, followed by one or more text modifiers (e.g., such as text colour, size, font) that are applied to the string during the time that the string is displayed within a text box. Each 3GPP text access unit is to be presented during a certain period of time, specified by the duration information.

The MP4 container assumes an object-oriented structure and composes of objects called boxes (viz., atoms). Figure 1 (a) shows the general structure of an MP4 box. Specifically, each box consists of three parts, namely size (i.e., the entire length of the box in terms of bytes), type (i.e., 4 characters code identifier) and data (i.e., the data of this box or other boxes). For a 32-bit MP4 file, size and type each occupies 4 bytes, while data assumes variable length of '*size - header*' bytes. Hereinafter, the term header refers to the combination of size and type, where its total length is 4 + 4 = 8 bytes, and each box is identified by its header (i.e., unique identifier) [12]. Each box can be categorized as a parent or a child box. Specifically, a parent box contains other boxes (i.e., sub-boxes). For example, the parent boxes are *moov*, *trak*, *minf*, *stbl*, etc. On the other hand, a child box carries information (i.e., data) such as *ftyp*, *mvhd*, *tkhd*, *stts*, *stco*, etc. Note that there is also a 64-bit version of MP4 container, but without loss of generality, we focus on the 32-bit version in this work. Figure 1 (b) shows the structure of a typical MP4 file. There are three top-level boxes, namely, ftyp, moov and mdat. The ftyp box stores the identification information of the MP4 file. The moov box includes the necessary information (viz., metadata) of the streams. The mdat box houses the actual media data of the container. Particularly, we exploit three boxes, namely, mdhd, stts and stco, to embed data. These boxes are shaded in Fig. 1 (b). The *mdhd* box is the media header box that stores the track level timescale for the specific track, which is used for frame timing during playback. Note that the video and audio tracks each has its own (i.e., independent) timescale. On the other hand, the *stts* box stores time-to-sample (i.e., frame timing) information. The 2-tuple, i.e., (sample count, delta), in the *stts* box determines the number of samples (i.e., either video

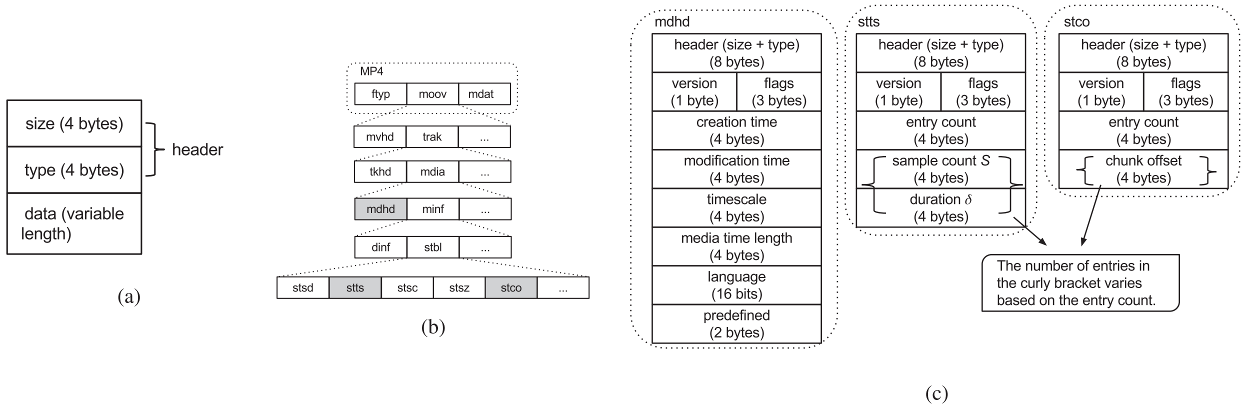


Fig. 1 Structure of MP4. The label (a) demonstrates a structure of MP4 box, (b) shows the general structure of MP4 file and (c) depicts the structure of *mdhd*, *stts* as well as *stco* boxes.

frames or audio samples) and how long (in unit of timescale) each sample will be displayed/played. The (sample count, delta) pairs within the *stts* box are shown in Fig. 1 (c). Last but not least, the *stco* box is the sample table chunk offset box. This box records where the media chunks are located. There are other boxes that carry other information. The interested readers can refer to the MP4 specifications [8].

3. Proposed Method

In this work, we exploit the synchronization between the subtitle track and the audio-video tracks to insert authentication data for the audio and video tracks. Specifically, the basic statistics S of the audio and video tracks are combined to form the authentication data, which is in turn encoded by manipulating the subtitle track within the same MP4 container. Here, it is assumed that the subtitle track is present in the MP4 file and contains some texts. In other words, the MP4 file in question 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.

3.1 Preparation of Payload

First, both the video and audio tracks are parsed to extract their statistics. These statistics S, which could be of variable length, are then independently hashed by using SHA256 to obtain a fixed hash code. To ease the presentation, we also refer to a video frame as a group (of pixels). Let $H(S(A_i))$ and $H(S(V_i))$ denote the SHA256 digest of the statistics S for the i-th group of audio samples and the i-th video frame, respectively. The XOR operations is applied to generate the intermediate data as follows:

$$\alpha_i \leftarrow H(S(A_{i+1})) \oplus H(S(V_i)). \tag{1}$$

The process is performed for $i = 1, 2, \cdots$, until all audio groups and video frames are considered. Figure 2 shows the computation of the hash values.

In most situations, the number of video frames and audio samples are not the same. Hence, we solve this problem by implementing the XOR operation between group pairs in a cyclic fashion. Specifically, when all the hash digests of the track with lesser groups are processed, the subsequent XOR operations will take input from the first group of the said track again. For example, when there are more groups of audio samples than video frames, $V_i \mod M$ is considered to produce α_i for $i > M_V$ where M_V is the total number of video frames in the MP4 file. This guarantees that the number of XOR-ed outputs α_i is the same as the number of groups of the track with more groups. Other forms of recycling is possible, including options determined by a private key.

Note that α_i is of length 256 bits, i.e., 64 hexadecimal digits. To reduce the storage space, each α_i is further processed. First, α_i is partitioned into segments of 32 bytes, and let each byte be denoted by α_i^j for $j = 1, 2, \cdots, 32$. Next, all partitions α_i^j are XOR-ed to obtain a single byte β_i as follows:

$$\beta_i = \alpha_i^1 \oplus \alpha_i^2 \oplus \alpha_i^3 \oplus \cdots \alpha_i^{32}. \tag{2}$$

The resulting value β_i, which is of length 8 bits, will be encoded by manipulating the i-th group of texts in the subtitle track.

3.2 Pre-Processing Time Coordinate System

Before inserting 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 $0.0\dot{3}$ seconds. If we set the new parameter values of $\tau = 60$ and $\delta = 2$, the ratio of $\tau : \delta$ remains unchanged because a unit of duration now represents (1/60) or $0.01\dot{6}$ seconds, but since the value of the duration is doubled, the final duration is still (1/30) or $0.0\dot{3}$ seconds. Technically, any combination of $(\tau, \delta) = (x\tau_0, x\delta_0)$ for integer $x > 0$ is equivalent to the combination of (τ_0, δ_0), for τ_0 and δ_0 being the original parameter values. Although τ and δ are 4-byte unsigned integer,

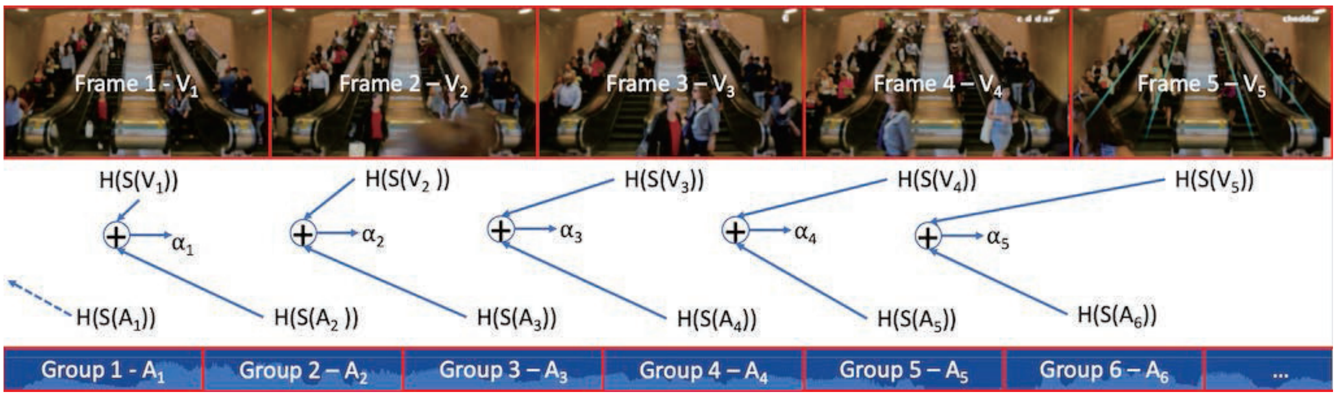


Fig. 2 The flow of processes in the proposed authentication technique. $\oplus$ denotes the XOR operation.

they cannot be set at the said maximum value as considered in [6]. It is because the display duration of a subtitle is significantly longer than that of a video frame. Hence, when the duration of a subtitle is scaled accordingly to maintain synchronization, an error will occur when there is an attempt to write a value of δ which exceeds the permissible maximum value. In order to avoid the occurence of this error, the system needs to be assigned with a smaller τ value as the constant to reduce the multiplier.

Due to the limitation of human visual system of seeing only up to 1000 FPS, a formula to update τ is set as $\lceil 255000/\tau_0 \rceil$. 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 insertion process is minimal so that it is 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 $(1/1) = 1$ second. If we increase the duration of a sample merely by 1, we are increasing its duration by 1 second, which results in obvious difference. Therefore, it is practical to consider a large τ value to ensure imperceptibility.

3.3 Embedding the Authentication Information

In this work, we distribute the payload evenly into all samples in the subtitle track T. Let $|T|$ denote the total number of samples (i.e., groups of texts) in the MP4 file. Specifically, we calculate how many payload to be encoded by a sample in T by using the formula below:

$$\mu = \left\lceil \frac{\max\{M_A, M_V\}}{|T|} \right\rceil, \tag{3}$$

where M_A and M_V denote the number of audio sample groups and video frames, respectively. When multiple payloads have to be embedded in one single subtitle sample, exactly μ number of payload β_i will be combined by repeatedly applying the XOR operations until a signal byte is obtained, i.e.,

$$\rho_k = \beta_{i=1+(k-1)} \oplus \beta_{i=2+(k-1)} \oplus \cdots \oplus \beta_{i=\mu\times k}. \tag{4}$$

To actually insert data into the subtitle track, the least significant byte of the δ parameter (i.e., the duration time of the i-th subtitle sample) of T_k is replaced by ρ_k. For example, given a duration with the value of 614_{10}, its equivalent binary representation is

$$00000000\ 00000000\ 00000010\ 01100110_2. \tag{5}$$

The least significant byte (i.e., 01100110) will be replaced by the authentication information ρ.

In case of any mismatch between the computed and extracted authentication data, we are able to identify a tampered region (either audio, video, or subtitle sample) to a certain extent. For instance, when we have 10 subtitle samples and 20 β's, we need to combine every 2 β's and insert the resulting value ρ into a single δ. As a result, if there's a mismatch between the third inserted payload and the XOR-ed digests obtained during the authentication phase, we can identify that the tampered sample, i.e., either the fifth or sixth group of audio samples, fifth or sixth video frame, or the third subtitle sample. Since the actual display time, in unit of second, of a specific subtitle entry is computed as δ/τ, 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. Specifically, the difference, in terms of seconds, will be computed as $\Delta \times \lceil 255000/\tau_0 \rceil$ seconds, where Δ is difference between original and modified δ value.

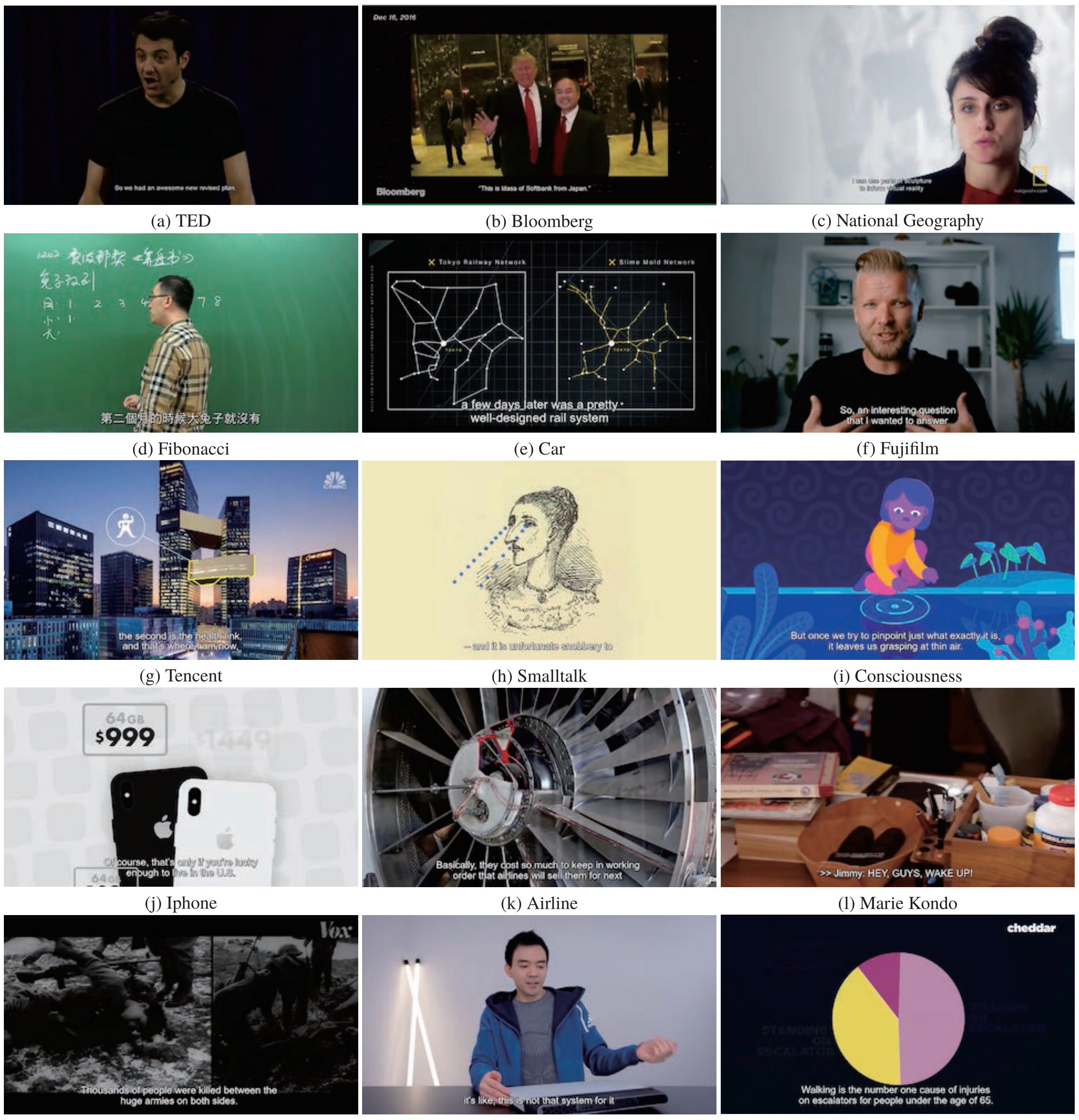
Before proceeding to the experiment section, 2 points are noteworthy. The first one is about how many hash values should be computed. In the aforementioned subsections, each group of audio samples and frame are considered to generate a hash value. At the other extreme, both audio and video tracks can be treated entirely as a whole to generate the hash value. The resulting hash value can then be embedded repeatedly into the subtitle track. Alternatively, the hybrid of both approaches can be considered, where a certain percentage of subtitle samples are reserved for the hash values generated from the individual groups of audio samples and video frames, and the rest for the hash value generated from the audio-video tracks as a whole. The second point is about the choice of hash value, which maybe vulnerable since one can modify the audio or video track, compute the new hash value, and then embed it into the subtitle track. One way to withstand this potential attack is to utilize a key based hash function, such as HMAC, in place of the hash function (viz., SHA-256 as mentioned above). When someone attempts to overwrite the authentication data in the subtitle track after unauthorized editing, the specific private key will be required. The key can be pre-communicated through any existing public key encryption infrastructure.

As shown in Fig. 2 and detailed in Sect. 3.1, the hash data computed by using the statistics of the audio and video samples are first aligned then XOR-ed to produce the intermediate data α_i, which is eventually converted to β_i. Therefore, each β_i is responsible for authenticating the $(i+1)$-th group of audio samples as well as the i-th video frame. Since β_i is embedded into the subtitle track, it can be extracted for verification purposes. For example, if the processed MP4 is modified by means of removing the 10-th frame, the original 11-th video frame will assume the position of the 10-th frame. When the authentication code is generated using the tampered video, a mismatch will occur. Specifically, β_{10} extracted from the subtitle track should match the authentication code generated by the 11-th group of audio samples and the 10-th video frame. However, the 10-th frame in the tampered video is actually the 11-th frame in the original video. When this mismatch happens, we can pinpoint that either 11-th group of audio samples and/or the 10-th frame are/is tampered with. Similar argument can be formulated for other forms of attack, and we omit the details here.

4. Experiment

15 YouTube videos are downloaded for experiment purposes. A representative frame of each video, along with the subtitle displayed, are shown in Fig. 3. These videos are re-encoded into a standalone MP4 file at 30 frames per second and its corresponding .srt subtitile file is encoded as a track by using a video encoding software called *Hand-Brake* [28]. These videos are made available online [29] for reproducibility purpose. For each video, the frame resolution is 1280×720 pixels encoded with H.264/AVC, and the audio track is encoded with AAC. For experiment purpose, $\tau = \lceil 255000/\tau_0 \rceil$ and since $\tau_0 = 90000$, $\tau = 270000$ are set. Detailed information about the videos are recorded in Table 1. 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. For experiment purpose, both video and audio tracks are parsed into a series of characters, 8 bits at a time, to form a string. The resulting string will be utilized as the statistics S.

First, the number of bytes which can be inserted into


So we had an awesome new revised plan.
Dec 16, 2016
Bloomberg
"This is Masa of Softbank from Japan."
第二個月的時候大兔子就沒有
Tokyo Railway Network
Slime Mold Network
a few days later was a pretty well-designed rail system
So, an interesting question that I wanted to answer
the second is the health link, and that's where I am now,
But once we try to pinpoint just what exactly it is, it leaves us grasping at thin air.
64GB
$999
Of course, that's only if you're lucky enough to live in the U.S.
Basically, they cost so much to keep in working order that airlines will sell them for next
>> Jimmy: HEY, GUYS, WAKE UP!
Vox
Thousands of people were killed between the huge armies on both sides.
it's like, this is not that system for it
cheddar
Walking is the number one cause of injuries on escalators for people under the age of 65.

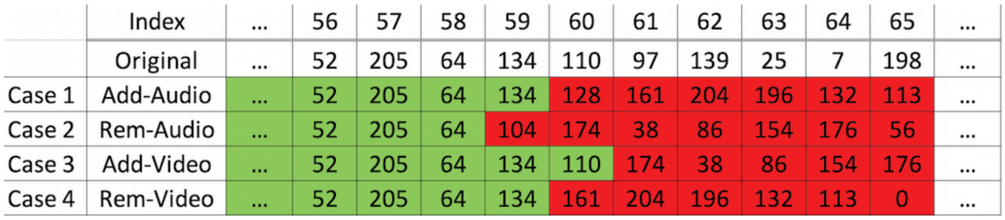
(a) TED (b) Bloomberg (c) National Geography
(d) Fibonacci (e) Car (f) Fujifilm
(g) Tencent (h) Smalltalk (i) Consciousness
(j) Iphone (k) Airline (l) Marie Kondo
(m) Kashmir (n) Alienware (o) Etiquette

Fig. 3 Representative frame of each video with subtitle displayed.

Table 1 Information about test video sequences.

Video Sequence	Total Frames	Total Audio Samples	Total Subtitles	MP4 Size(MB)	Data per subtitle, μ	MSE†
TED [13]	20248	36334	315	98.9	116	9795.33
Bloomberg [14]	8759	16734	77	38.9	218	10589.74
National Geographic [15]	5622	8079	50	41.3	162	11892.48
Fibonacci [16]	16709	24011	272	59.8	89	10882.05
Car [17]	10781	19366	274	74.2	71	9635.96
Fujifilm [18]	25405	45634	436	145	105	12024.71
Tencent [19]	6227	8940	70	61.1	128	8585.37
Smalltalk [20]	8401	15076	75	42.9	202	12729.83
Consciousness [21]	17427	25018	131	55.3	191	10790.10
Iphone [22]	23212	33322	232	70.4	144	12186.84
Airline [23]	9494	13630	109	47.9	126	10650.94
Marie Kondo [24]	13795	19824	261	85.3	76	10575.53
Kashmir [25]	14577	26184	203	90.8	129	12270.77
Alienware [26]	14424	25909	172	62.5	151	10080.13
Etiquette [27]	9240	16598	125	63.2	133	12284.11

†MSE is computed as raw value, without the multiplication by $\lceil 255000/\tau_0 \rceil$.



	Index	...	56	57	58	59	60	61	62	63	64	65	...
	Original	...	52	205	64	134	110	97	139	25	7	198	...
Case 1	Add-Audio	...	52	205	64	134	128	161	204	196	132	113	...
Case 2	Rem-Audio	...	52	205	64	104	174	38	86	154	176	56	...
Case 3	Add-Video	...	52	205	64	134	110	174	38	86	154	176	...
Case 4	Rem-Video	...	52	205	64	134	161	204	196	132	113	0	...

Fig. 4 Authentication data before and after insertion and removal attacks.

each MP4 file are recorded in the forth column of Table 1, i.e., Total Subtitle, because we restrict to hide one byte into each subtitle sample. 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. Table 1 (viz., sixth column) also records the number of β_i that are combined to form ρ_k for actual encoding using the proposed data hiding method. As expected, the value of μ increases when there are less subtitle samples for manipulate, and vice versa.

Second, the means square error (MSE) between the new and original durations, denoted by Δ, is computed and recorded in Table 1 (last column). 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 difference is relatively small. Specifically, since $\Delta(\delta_{\text{ori}}, \delta_{\text{new}}) \in [-255, 255]$, the differences before and after data insertion, in the unit of second, will be bounded by range of

$$\left[\frac{-255}{\tau}, \frac{255}{\tau}\right] = \left[\frac{-255}{\lceil 255000/\tau_0 \rceil}, \frac{255}{\lceil 255000/\tau_0 \rceil}\right] = [-\beta, \beta], \tag{6}$$

where $\beta = 0.00094$ [7]. 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 actually be hidden by considering more bytes of δ (e.g., $1.5 \sim 2$ bytes), as long as the recommendation of 0.3 per words is satisfied.

Next, we illustrate the ability of the proposed method in detecting tampered audio samples, video frames, or subtitle samples. The original authentication data is recorded in the first row of Fig. 4. Here, 4 forms of attack are considered, and the resulting authentication data is recorded in Fig. 4:

A1. Add audio samples: The 60-th group of audio samples, i.e., A_{60} is repeated. Therefore, the new audio content becomes $[A_1, A_2, \cdots A_{60}, A_{60}, A_{61}, A_{62}, \cdots]$, which is of longer duration than the original audio. The video and subtitle tracks remain unmodified. When considering the authentication data, the computed data (3rd row in Fig. 4) matches that of the original until the 59-th position, but mismatches occur from the 60-th position onward. Therefore, one can conclude that some form of tamparing occurs at 60-th and potentially beyond that point as well.

A2. Remove audio samples: The 60-th group of audio samples is removed. Therefore, the new audio content becomes $[A_1, A_2, \cdots A_{59}, A_{61}, A_{62}, A_{63}, \cdots]$, which is of shorter duration than the original audio. The video and subtitle tracks remain unmodified. The mismatch between the embedded and the computed authentication data starts at the 59-th position, and continue thereinafter. It is because the 59-th data is computed by using $S(V_{59})$ and $S(A_{60})$.

A3. Add video frame: The 60-th video frame is repeated. Therefore, the new video content becomes

$[V_1, V_2, \cdots V_{60}, V_{60}, V_{61}, A_{62}, \cdots]$, which is of longer duration than the original video. The mismatch between the embedded and the computed authentication data starts at the 61-st position, and continue thereinafter. It is interesting to note that the new codes match those of case A2, but with some delay. It is because β_i is computed by using A_{i+1} and V_i, hence removing 1 group of audio samples or adding a frame will have a similar effect.

A4. Remove video frame: The 60-th video frame is removed. Therefore, the new video content becomes $[V_1, V_2, \cdots V_{59}, V_{61}, V_{62}, A_{63}, \cdots]$, which is of shorter duration than the original video. The mismatch between the embedded and the computed authentication starts at the 60-th position, and continue thereinafter. Again, it is interesting to note that the new codes match those of case A1, but with some delay for the same reason provided in A3.

In all cases, the position of the first mismatch can be identified, hence localizing the first tampered audio/video/subtitle sample. Here, we do not consider the case where a subtitle sample is added or removed, because it has a similar effect as removing or adding audio / video sample. One may argue that simultaneous injection of audio samples, video frame and subtitle will be able to foil the system. However, it should be noted that the i-th authentication data (i.e., β_i) is generated by using V_i and A_{i+1}. Therefore, the point at which these new elements are simultaneously added will be detected. One may also argue that overwriting the subtitle samples will defeat the system, but HMAC (which requires a key) can be introduced as mentioned above. Therefore, the proposed system is able to deliver its intended function, i.e., authentication.

5. Discussions

To the best of our knowledge, there is only one authentication technique exploiting the container structure of MP4 [6]. When compared to [6], the proposed method is able to preserve the bit stream size irregardless of the length of the video. It is because in the proposed method, the value of the parameter δ, which is stored using a fixed size data structure (i.e., 4 bytes), is modified to hide data. Since no additional (s, δ) pairs are injected into the MP4 container, the bit stream size is maintained. Nonetheless, similar to [6], the proposed method can also inject empty groups of texts to increase the (s, δ) entries so that more data can be hidden, but at the expense of bit stream size increment. In addition, [6] requires one to decode and reconstruct the video in order to compute the video hash, while the proposed method considers low level statistics S extract from the bit stream.

One advantage of the proposed method over the conventional semantic based authentication methods is speed. Specifically, the proposed method relies on the bit patterns of an MP4 file, while the conventional audio/video authentication methods focus on semantics derived from the audio/video samples. In the case of a compressed video, the encoded bit stream needs to be parsed in order to obtain individual arithmetic codewords, which are in turn converted (by means of table look-up) to reconstruct the macroblocks or coding unit. The process is followed by a dequantization process, then an inverse transformation process to obtain the raw pixel values. In some cases, a video frame is reconstructed based on the previous frames by means of motion compensation, and this leads to dependency of the reconstructed data in the authentication process. Similar arguments can be said about audio authentication technologies considering raw audio samples. Besides, the conventional methods for audio/video commonly store the authentication data within the content itself [4], [30], while the proposed method generates authentication data from the audio and video tracks, then embeds the data into the subtitle track.

In terms of recompression, the proposed method will flag the recompressed MP4 file as tampered, while the conventional semantic-based techniques can potentially allow authentication to take place. To cater for recompression in the proposed method, a rightful (co-)owner can first verifies the authenticity of the received MP4. If it is genuine, the MP4 file (technically the audio and video tracks) is then recompressed. The resulting MP4 file will undergo the same procedures in the proposed method to update the subtitle track with the new authentication data. This process is fast since bit patterns are considered instead of audio/video sample values.

In any case, the proposed method can be combined with the conventional semantic based techniques to achieve a more complete authentication scheme, including those that manipulate syntax elements in audio (e.g., scale factor, MDCT coefficients in MP3 [31]) and video (e.g., prediction mode [32], [33], coefficients [34], motion vectors [35], quantization parameters [36]). For example, the proposed method can be utilized as the first layer of authentication, while the conventional authentication methods for audio [2] and video [4] can be applied to localized modified part of the audio and/or video. Therefore, when a MP4 fails the first layer of authentication, the video is deemed not original. In such situation, there is no point to further view / transmit / process the content in question [4]. If the modified parts need to be localized, the proposed method identifies the range of samples where the tampered samples start to appear. Then, the second layer of audio- or video based-authentication can be invoked to pinpoint the exact location. Table 2 summarizes the differences between the proposed and conventional authentication methods.

Furthermore, in this study, we only consider videos with the subtitle track, where the subtitle track basically display the dialogue / conversation in the video. Our proposed authentication technique can be utilized for MP4 containing the closed-captioning track. Closed-captioning is made available for audience with hearing impairment, and it serves as narrative of the video scenes [37]. In such situation, the closed-captioning sequence will contain more groups of texts, which translates directly to more oppor-

Table 2 Functional comparison between conventional and proposed authentication methods.

	Conventional	Proposed
Source of data derivation	Pixel values, audio samples	Statistics of bit patterns
Venue to store authentication data	Within the content of interest	Subtitle track carries the multiplexed authentication data from audio and video tracks
Decoding to raw / intermediate sample values required	Yes (slow)	No (fast)
Dependency on previously decoded data	Yes	No
Robust to compression	Yes	No

tunities for data insertion. It is because closed-captioning contains significantly more groups of texts (e.g., text narrating *door opened*, *engine rumbling*, *laughing evilly*) as oppose to subtitle. The proposed method is also applicable for Karaoke video with texts. Moreover, we can also modify the subtitle (closed-caption or Karaoke) texts, where uni-code space characters, spaces, and newline characters can be introduced to improve the payload without causing any visual distortion [38].

Last but not least, one may argue that the conventional MP4 based data hiding methods can be refined to provide authentication feature. However, there are only a handful of techniques based on MP4 container format. For example, Jokay's method exploits the size of the GOP structure (i.e., odd or even) or the length of video chunk in the MP4 container (viz., chunk structure) to hide data [39]. It is not suitable for authentication because some video frames will be discarded. Second, Cosimo et al. proposes a steganographic method called OpenPuff Tool, which manipulates selected flags in the MP4 container to embed data while maintaining format compliance [40]. This method is reversible but its embedding capacity is low, because it depends on the number of flags available, which is fixed. Hence, it is not able to provide sufficient payload for authentication purpose, especially when the video is long. In the third method [41], the TrueCrypt container (in which itself is an encryption container) is injected within an MP4 container to form a hybrid MP4/TrueCrypt container file. Specifically, the TrueCrypt container is inserted into the mdat box, and the stco chunk offset is modified to point to the position of the actual media data, i.e., audio or video. This method is reversible, and its embedding capacity is high. However, two requirements are imposed by this method: the TrueCrypt container must have two volumes (i.e., outer volume and hidden volume) and MP4 structure has to be in the form of *ftyp-mdat-moov*. In addition, this method [41] inserts opaque data into the MP4 file to form MP4/TrueCrypt Hybrid file. Therefore, the hidden data can be removed easily. One can notice the hybrid file being large and the player bit rate is small. By design, there will be 2 *mdat* headers: one is fake and one is real. Moreover, there will be unused space that is not referenced by *stco*. To this end, the hidden data in MP4/TrueCrypt file is not suitable for authentication purposes.

6. Conclusions

A lightweight authentication technique is put forward for MP4 file containing the audio, video and subtitle tracks. The synchronization of the subtitle samples, viz., duration time, is manipulated to encode the authentication information. The hash values from the audio- and video-samples are computed and utilized as the authentication data. Although the proposed authentication method can only coarsely locate the tampered samples, it can be combined with conventional authentication methods based on audio and/or video. Experiments confirm that the file size of the processed MP4 is completely maintained, while the mean square error in terms of the display duration of the modified groups of texts is small and well within the noticeable range.

As future work, we shall explore different type of statistics to be utilized as the authentication information. Actual joint utilization of audio-, video-based as well as text-based authentication methods along with the proposed method will be investigated.

Acknowledgments

This work is supported by the Engineering-IT collaborative grant scheme 2018.

References

[1] K. Smith, "52 fascinating and incredible youtube statistics," https://www.brandwatch.com/blog/youtube-stats/, 2019.

[2] M. Steinebach and J. Dittmann, "Watermarking-based digital audio data authentication," EURASIP Journal on Advances in Signal Processing, vol.2003, no.10, 252490, Sept. 2003.

[3] Y. Tew and K. Wong, "An overview of information hiding in H.264/AVC compressed video," IEEE Trans. Circuits Syst. Video Technol., vol.24, no.2, pp.305–319, 2014.

[4] Y. Tew, K. Wong, R.C.-W. Phan, and K.N. Ngan, "Multi-layer authentication scheme for HEVC video based on embedded statistics," J. Vis. Comun. Image Represent., vol.40, Part B, pp.502–515, Oct. 2016.

[5] I.M. Maung, Y. Tew, and K. Wong, "Authentication for AAC compressed audio using data hiding," 2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), pp.1–2, May 2016.

[6] A.P.M. Maung, Y. Tew, and K. Wong, "Authentication of mp4 file by perceptual hash and data hiding," Malaysian Journal of Computer Science, vol.32, no.4, 2019.

[7] C. Chan, K. Wong, and I. Maungmaung, "Data hiding in MP4 video container based on subtitle track," APSIPA, pp.1128–1131, IEEE, 2018.

[8] ISO, "Information technology - coding of audio-visual objects - part 12: ISO base media file format," ISO 14496-12, International Organization for Standardization, Geneva, Switzerland, 2005.

[9] ISO, "Information technology - coding of audio-visual objects - part 14: Mp4 file format," ISO 14496-14, International Organization for Standardization, Geneva, Switzerland, 2003.

[10] Apple, "Introduction to quicktime file format specification," 2015.

[11] S. Enamorado, "How to create an SRF file," https://www.3playmedia.com/2017/03/08/create-srt-file/, 2017.
[12] F.C. Pereira and T. Ebrahimi, The MPEG-4 Book, Prentice Hall Professional, 2002.
[13] TED, "Inside the mind of a master procrastinator | Tim Urban," https://www.youtube.com/watch?v=arj7oStGLkU, 2016.
[14] Bloomberg, "How Masayoshi Son became an eccentric dealmaker," https://www.youtube.com/watch?v=cDpTPrfw1mQ, 2018.
[15] National Geographic, "Re-envisioning reality - tech+art | genius: Picasso," https://www.youtube.com/watch?v=T9chHEEp-0M, 2018.
[16] Y. Li, "A lecture on Fibonacci series," https://www.youtube.com/watch?v=VCJsUYeuqaY, 2018.
[17] Verge Science, "What self-driving cars can learn from brainless slime mold," https://www.youtube.com/watch?v=40f7_93NIgA, 2018.
[18] M. Haapoja, "Fujifilm x-t3 review - 4k 60p, 120p for $1500? yes please!," https://www.youtube.com/watch?v=bZ-SvBfAFtM, 2018.
[19] CNBC International, "Inside Chinese tech giant tencent's vertical campus | CNBC reports," https://www.youtube.com/watch?v=Huju5McKgIU, 2018.
[20] The School of Life, "What to do if you hate small talk," https://www.youtube.com/watch?v=SrK5NAgw_g4, 2018.
[21] Kurzgesagt - In a Nutshell, "The origin of consciousness — How unaware things became aware," https://www.youtube.com/watch?v=H6u0VBqNBQ8, 2019.
[22] PolyMatter, "The true cost of the iphone," https://www.youtube.com/watch?v=5kZRY5xlP6Y, 2019.
[23] Half as Interesting, "The 30 year-old airline that's never flown," https://www.youtube.com/watch?v=PB2RuEz2VV4, 2019.
[24] Jimmy Kimmel Live, "Marie Kondo helps Jimmy Kimmel tidy up," https://www.youtube.com/watch?v=-3VCoY27fFI, 2019.
[25] Vox, "The conflict in Kashmir, explained," https://www.youtube.com/watch?v=cyayif_nla8, 2019.
[26] D. Lee, "The alienware beast" https://www.youtube.com/watch?v=pAEVLUeYaf8, 2019.
[27] Cheddar, "The unseen inefficiency of escalator etiquette - cheddar explains," https://www.youtube.com/watch?v=vbsoO2c7gCM, 2019.
[28] The HandBrake Team, Handbrake.
[29] K. Wong, C.S. Chan, and I. Maungmaung, https://bit.ly/2Zfpuxk
[30] V. Amanipour and S. Ghaemmaghami, "Video-tampering detection and content reconstruction via self-embedding," IEEE Trans. Instrum. Meas., vol.67, no.3, pp.505–515, March 2018.
[31] K. Takagi, S. Sakazawa, and Y. Takishima, "Light weight MP3 watermarking method for mobile terminals," IEICE Trans. Fundamentals, vol.E91-A, no.9, pp.2546–2554, Sept. 2008.
[32] L. Pang, K. Wong, and S.T. Liong, "Data embedding in scalable coded video," 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA ASC 2017, Kuala Lumpur, Malaysia, Dec. 12–15, 2017, pp.1190–1194, IEEE, 2017.
[33] N. Mehmood and M. Mushtaq, "A fragile watermarking scheme using prediction modes for H.264/AVC content authentication," 37th Annual IEEE Conference on Local Computer Networks - Workshops, pp.1014–1021, Oct. 2012.
[34] J. Zhang and A.T.S. Ho, "Efficient video authentication for H.264/AVC," First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC '06), pp.46–49, Aug. 2006.
[35] S. Acharjee, S. Chakraborty, S. Samanta, A.T. Azar, A.E. Hassanien, and N. Dey, "Highly secured multilayered motion vector watermarking," Advanced Machine Learning Technologies and Applications, ed. A.E. Hassanien, M.F. Tolba, and A.T. Azar, Cham, pp.121–134, Springer International Publishing, 2014.
[36] T. Shanableh, "Matrix encoding for data hiding using multilayer video coding and transcoding solutions," Signal Processing: Image Communication, vol.27, no.9, pp.1025–1034, Oct. 2012.
[37] Y. Kim, S. Han, S. Choi, and B. Jung, "File-based closed captioning system without captioning delay," SMPTE 2015 Annual Technical Conference and Exhibition, pp.1–8, Oct. 2015.
[38] L.Y. Por, K. Wong, and K.O. Chee, "UniSpaCh: A text-based data hiding method using unicode space characters," J. Syst. Softw., vol.85, no.5, pp.1075–1082, May 2012.
[39] M. Jókay, "The design of a steganographic system based on the internal mp4 file structures," International Journal of Computers and Communications, vol.5, 2011.
[40] C. Oliboni, "Openpuff v4.00 steganography and watermarking," July 2012.
[41] M. Fiedler, "Real steganography with truecrypt," http://keyj.emphy.de/real-steganography-with-truecrypt, Feb. 2011.

KokSheik Wong is an Associate Professor attached to the School of Information Technology at Monash University Malaysia. He received the B.S. and M.S. degrees in both Computer Science and Mathematics from Utah State University, USA, in 2002 and 2005, respectively. In 2009, he received the Doctor of Engineering degree from Shinshu University, Japan, under the scholarship of Monbukagakusho. He was with Multimedia University from 2009 to 2010, and University of Malaya from 2010 to 2016. His research interests include multimedia signal processing, data hiding, multimedia perceptual encryption, joint encryption and data-hiding method, as well as their applications. He is a senior member of IEEE and a member of APSIPA. In 2015, his student's thesis received the Best Ph.D. thesis award from the IEEE Signal Processing Society, Malaysia Section. Currently, he serves as an associate editor of the Journal of Information Security and Applications (JISA), Malaysian Journal of Computer Science and IIEEJ Transactions on Image Electronics and Visual Computing.

ChuanSheng Chan received the B.S. degrees in Computer Science from Monash University, Malaysia, in 2019. His research interests include multimedia signal processing, data hiding, multimedia perceptual encryption, joint encryption and data-hiding method, as well as their applications.

AprilPyone MaungMaung received the BCS (Bachelor of Computer Science Honours) degree from International Islamic University Malaysia in 2013, and the MCS (Master of Computer Science) degree from University of Malaya in 2018. Currently, he is a Ph.D. candidate in Tokyo Metropolitan University, Japan. His research interests include multimedia security and machine learning.