Broken Integrity Detection of Video Files in Video Event Data Recorders

Choongin Lee[1], Jehyun Lee[2], Youngbin Pyo[1], and Heejo Lee[1]
[1] Department of Computer Science and Engineering, Korea University
Seoul 02841, Republic of Korea
[e-mail: {choonginlee, vydudqls, heejo}@korea.ac.kr]
[2] Cyber Security Research Center, Korea Advanced Institute of Science & Technology
Daejeon 34141, Republic of Korea
[e-mail: jehyunlee@kaist.ac.kr]
*Corresponding author: Heejo Lee

Received April 21, 2016; revised July 6, 2016; accepted August 1, 2016; published August 31, 2016

Abstract

As digital evidence has a highly influential role in proving the innocence of suspects, methods for integrity verification of such digital evidence have become essential in the digital forensic field. Most surveillance camera systems are not equipped with proper built-in integrity protection functions. Because digital forgery techniques are becoming increasingly sophisticated, manually determining whether digital content has been falsified is becoming extremely difficult for investigators. Hence, systematic approaches to forensic integrity verification are essential for ascertaining truth or falsehood. We propose an integrity determination method that utilizes the structure of the video content in a Video Event Data Recorder (VEDR). The proposed method identifies the difference in frame index fields between a forged file and an original file. Experiments conducted using real VEDRs in the market and video files forged by a video editing tool demonstrate that the proposed integrity verification scheme can detect broken integrity in video content.

Keywords: Digital Forensics, Multimedia Forensics, Multimedia Security, Computer Security

A preliminary version of this paper was presented at ICONI 2015, and was selected as an outstanding paper. This research was supported by the Public Welfare & Safety Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2012M3A2A1051118).

1. Introduction

The increased use of surveillance cameras such as Video Event Data Recorders (VEDRs) and Closed Circuit Television (CCTV) cameras in the recent past has resulted in their rapid worldwide deployment [1]. Even considering that surveillance devices could infringe on privacy, this distribution trend is unavoidable because recorded videos play a key role as evidence in disputatious situations [2-4]. Further, surveillance devices assist criminal investigators by providing significant evidence.

Nevertheless, video evidence is susceptible to tampering. With the advent of easy-to-use video editing tools, video files are increasingly at risk of being forged [5-6]. Consequently, it is entirely possible that a forged video could be submitted to a court as evidence [7]. In such a scenario, because there is no guarantee that the video is genuine, the evidence could result in an incorrect verdict in a criminal matter. Further, even when the video is not modified intentionally, the video data stored in the surveillance devices' storage can be damaged by external crashes. Thus, it is essential that the integrity of video data be verified. Consequently, the reliability of video data integrity verification techniques is a critical issue for digital forensics investigators. For example, recently Garfinkel et al. [8] introduced a digital media carving technique by using sector hashing and hashdb to verify the integrity of media files.

In this paper, we propose an integrity verification scheme for a VEDR video that uses the frame index data in video files. By checking for anomalies in the index artifact of the video files stored in the File Allocation Table 32 (FAT32) file system, the scheme verifies the integrity of Audio Video Interleave (AVI) video data. (AVI is one of the most popular video file formats used in VEDRs.)

Pre-employed integrity check codes or watermarks of video frames reserved at the time of recording have been proposed for video integrity verification in previous studies [9-10]. However, these pre-employment approaches have deployment problems because they must be embedded in the target VEDR devices by the manufacturers. Post-processing approaches involving image analysis have also been proposed to detect image manipulation [11-12]. However, they do not address frame-editing cases, where no changes occur in the remaining images, such as frame deletion without re-encoding. Our previous work [13] used frame-based verification to rectify the limitation of previous image-processing techniques. However, frame-based verification has limitations detecting cases where video frames are deleted at the end of a frame sequence. In this study, we achieve video file integrity verification without using any pre-employment schemes and overcome the limitations identified in previous proposals.

We evaluate the proposed method using video files recorded by VEDR devices and modified using a video editing tool in several scenarios. The evaluation results confirm that the proposed scheme can identify compromised integrity wherever video frames are deleted in a frame sequence, overcoming the limitation of our previous work. Of the eight forged samples, all eight were identified as being modified when various numbers of frames were deleted from the original video file.

The remainder of this paper is organized as follows. In Sections 2 and 3, we briefly explain VEDR file systems and video file attributes and provide an overview of previous studies related to video file integrity verification, respectively. In Section 4, we outline the proposed method. In Section 5, the efficacy of the proposed verification method is evaluated. In Section 6, we discuss several topics associated with frame data recovery and the compatibility of the proposed method with other file systems and video formats. Finally, we conclude this paper in Section 7.

2. Background

2.1 Video Frames

Video files consist of sequential static images called video frames. **Fig. 1** illustrates an example of video frame composition in a compressed video. The frames comprise a sequential Group of Pictures (GOPs), with each GOP containing a key frame followed by delta frames and an encoding profile. In a compressed video, only the key frames have the entire image information; the delta frames contain only information regarding the bits that have changed compared to the previous frame.

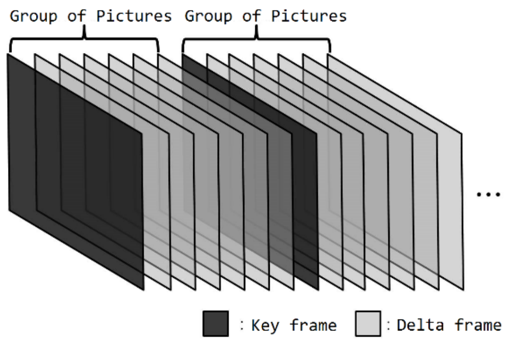

Fig. 1. Example of video frame composition

2.2 File Format of AVI Videos

The AVI video format [14] is designed following the RIFF standard presented by Microsoft. It contains both video and audio streams. The general model of the AVI format is depicted in **Fig. 2**. The actual frame data cover a field starting with `LIST movi`. The ensuing field, starting with `idx1`, contains frame index information including the offset and size of each actual set of frame data.

RIFF AVI identifier	LIST hdrl	...	...	LIST movi	idx1

Fig. 2. Basic Architecture of AVI Files

2.3 Allocation of Video Content in FAT32

Video files stored in storage media such as SD and NAND flash memory are managed by file system rules that provide a mechanism to store, categorize, and access data [15]. We adopted FAT32 as our target file system because FAT32 is the most compatible file system across all embedded systems including VEDRs. Basically, the relative positions of files stored in FAT32 file system is correlated with their creation order [16-17]. With the property of the file allocation, we can set several possible scenarios. **Fig. 3** presents three cases of a file allocation scenario in the FAT32 file system. A cell in the areas indicated by C1 to CX represents *a*, which is composed of sectors, the minimum unit logically addressable in the FAT32 file system.

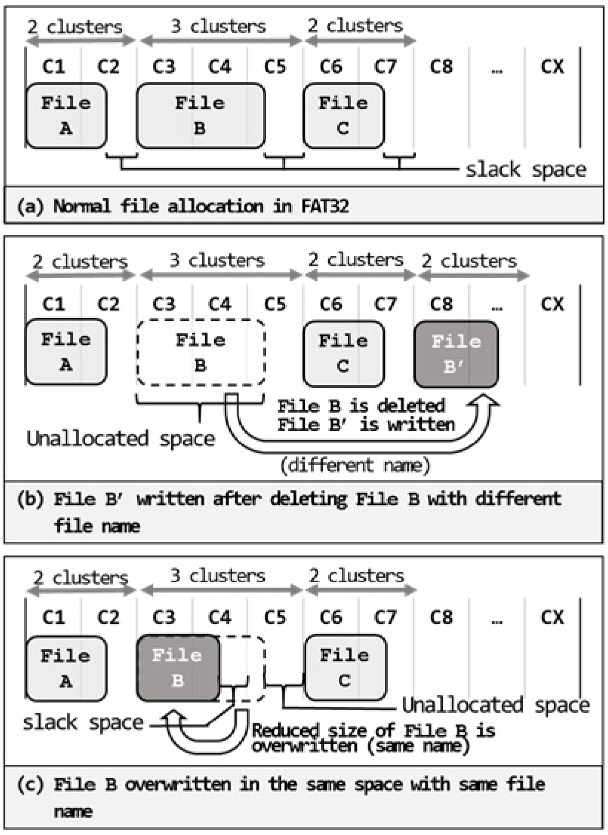
2 clusters
3 clusters
2 clusters
C1 C2 C3 C4 C5 C6 C7 C8 ... CX
File A
File B
File C
slack space
(a) Normal file allocation in FAT32
2 clusters
3 clusters
2 clusters
2 clusters
C1 C2 C3 C4 C5 C6 C7 C8 ... CX
File A
File B
File C
File B′
File B is deleted
File B′ is written
Unallocated space
(different name)
(b) File B′ written after deleting File B with different file name
2 clusters
3 clusters
2 clusters
C1 C2 C3 C4 C5 C6 C7 C8 ... CX
File A
File B
File C
Unallocated space
slack space
Reduced size of File B is overwritten (same name)
(c) File B overwritten in the same space with same file name

Fig. 3. Examples of file allocations in a FAT32 file system

Fig. 3 (a) is a case where three files are continuously allocated in the file system. We suppose that File A, B, and C have taken possession of two, three, and two clusters, respectively. At the end of each file, there remains a space called *slack space* [15]. Slack space is the unused space in the last cluster when the size of a file is not an exact multiple of the cluster size.

Fig. 3 (b) is a case where another video data File B', depicted as a dark grey square, is written after deleting the original File B. File B' is smaller than File B because data frames have been deleted using a video editing tool. The three clusters originally occupied by File B are referred to as *unallocated space*. Unallocated space refers to the unused region of a file allocation system. This space is present when either no files have yet been allocated or previously allocated files have been deleted from the file system. Because parts of the remaining frame data found in the original File B may correspond to frame data comprising File B', we assume that both files have a *duplicated area*.

Fig. 3 (c) describes a situation where File B, with a smaller size, overwrites the area of the original File B. This can happen because file systems such as FAT32 allocate a smaller sized file to the same position if the file fits into the unallocated sectors between two other files. In this case, there are both a slack space and unallocated space in the residual area because File B is not sufficiently large to occupy all three clusters. For ease of explanation, we will call both slack space and unallocated space *residual space*. The duplicated area of both the original File B and the secondary File B can be found in the allocated space of secondary File B.

This paper focuses on case (c), identifying broken integrity when there is no suspicious factor in the metadata of the file system. In case (b), the area where the video content has been deleted is obvious as we can analyze metadata such as the directory entry in FAT32; however, it is not possible to verify integrity with only metadata information in case (c). Note that the proposed scheme can also be applied to case (b).

3. Related Work

Integrity verification of video content can be studied from either of the following two perspectives: pre-processing or post-processing. The pre-processing method operates when the video content is being stored. An example of this approach is the integrity verification method proposed by Jayamalar *et al.*, where digital watermarking is embedded in the video content [10]. This method enables investigators to observe illegal copying or manipulation while playing the video in real time.

Kim *et al.* also proposed a data integrity scheme [9] for preventing falsification. The proposed scheme stores video content in two secure channels for original files and encrypts each file. Then, to verify the integrity of the video files, the original video files are encrypted and compared with the previously encrypted stored videos. However, these pre-processing approaches are difficult to incorporate into legacy surveillance camera systems because they must be applied in the design phase while the systems are being manufactured. The scheme proposed in this paper is applicable to surveillance cameras that are not equipped with a built-in protection module. The scheme verifies integrity only with the disk storage itself.

Similar to the scheme proposed in this paper, post-processing methods have been applied to investigations after an incident has occurred. Previous proposals detect digital forgeries using the video noise data caused by the instability of camera sensors or inter-frame correlation. Wang *et al.* [11] and Hsu *et al.* [12] proposed a forgery detection method that analyzes the noise data of images in video files. Wang *et al.* [18] proposed a method that detects duplicated frames using inter-frame correlation. However, the previous methods only operate in image modification cases. Conversely, the scheme proposed in this paper achieves integrity verification in cases of frame editing by skimming the frame indices of the AVI files and disk space. Searching for the frame index area is not costly because, in a video file, the size of the index area that must be parsed is significantly smaller than that of the frame areas.

4. Integrity Verification Scheme for Disk Storage in VEDR

In this section, we examine a scenario where a video editing tool is used to modify a video. Verification of video file integrity with metadata in video frames is applicable to the majority of practical video file formats. By setting cases that address the most practical scenarios, we primarily target the AVI file format, which is one of the most popular formats used by VEDRs in practice. Next, we introduce the proposed scheme using the AVI file structure as an example. For the AVI file, we verify integrity by comparing the frame index information retrieved from both the allocated and residual space.

Fig. 4 presents the overall flow of the proposed VEDR video file-integrity verification scheme. The scheme extracts the frame indices from the logical disk storage of a VEDR device using residual space and allocated space parsers. Then, forged video files are detected by comparing the frame indices extracted from each space.

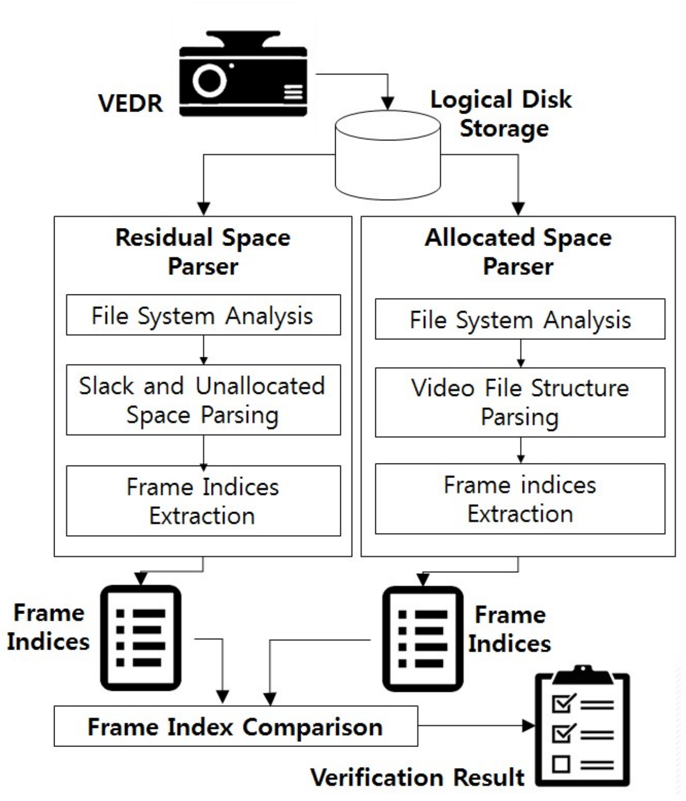

Fig. 4. Process flow of proposed VEDR video file-integrity verification scheme

4.1 Video Content Fabrication Scenarios

Video file fabrication can be performed with two kinds of techniques: image editing and frame editing [4].

- **Image editing**: In cases where frame images contain crucial evidence, a suspect may attempt to manipulate the image. For instance, it is possible to change a traffic light from red to green using a video editing tool such as Adobe Premiere or Sony Vegas.
- **Frame editing**: If suspicious frames are present in a video file, a suspect may delete parts of the unfavorable frames. After merging previous frames with subsequent frames, the suspect may then overwrite an original video file with the forged file.

Image editing techniques can be used to change images and encode frames on a video file. However, the modifications made can be detected from the changes in the image quality. Conversely, frame-editing techniques do not affect the image itself; rather, they are used to add, delete, and reorder the image frames. The proposed scheme primarily focuses on broken integrity of the frame-editing techniques, an area that is not sufficiently covered by conventional image investigation techniques. In particular, frame editing with frame deletion is critical because the original data does not remain on the forged video file, unlike addition and reorder.

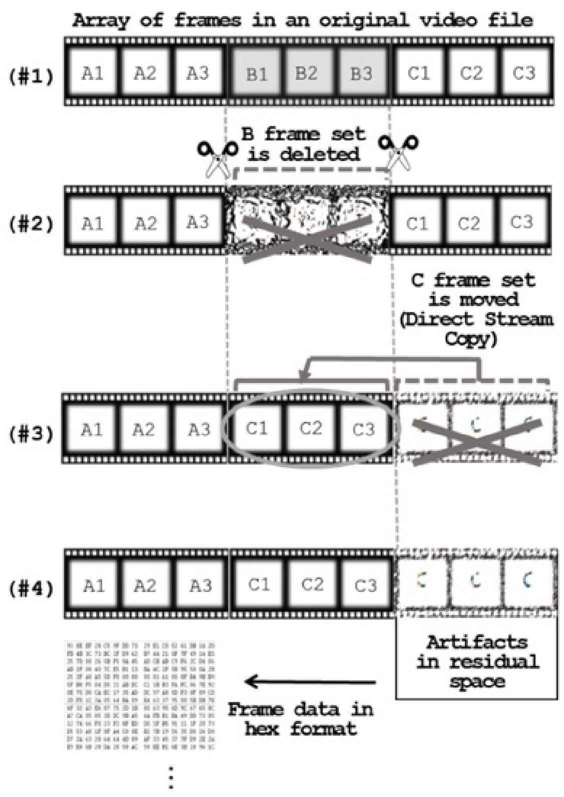
Array of frames in an original video file
(#1)
A1
A2
A3
B1
B2
B3
C1
C2
C3
B frame set is deleted
(#2)
C frame set is moved (Direct Stream Copy)
(#3)
(#4)
Artifacts in residual space
Frame data in hex format

Fig. 5. Process of frame deletion in video files resulting in artifacts in the residual space

Fig. 5 depicts a frame-editing scenario with deletion. In this scenario, an array of video frames is composed of frame sets *A*, *B*, and *C*. Let us assume that frame set *B* contains an unfavorable scene for the suspect (#1). Therefore, the suspect deletes frames in frame set *B* and uses "direct stream copy" not to re-encode the video file and hide the intentional tampering of the video. Then, the following frame set, *C*, is attached to the end of frame set *A* (#2). The suspect then overwrites the original video file with a forged video file that has the same name as the original video (#3). In accordance with the file allocation property in FAT32, artifact data will remain in the residual space (#4).

Depending on the location of the deleted frames in a frame sequence, there are two possible scenarios: Overlapping and Tail cut. An overlapping scenario arises when frames are not deleted at the end of the frame sequence in a video file. Conversely, a tail scenario arises when the frames are deleted at the end of the sequence. Both scenarios follow the simple video editing process illustrated in Fig. 5.

- **Scenario 1: Overlapping**

An overlapping scenario arises when the frame deletion is not conducted at the end of a frame sequence. As illustrated in Fig. 6, the deletion of frames results in partial matching of frames between the stored video file and residual space. The matching frame areas are indicated as (A) and (B). The remaining area, (C), is not matched with any frame in the stored video file. The mismatched area becomes larger as the size of the deleted frames becomes greater.

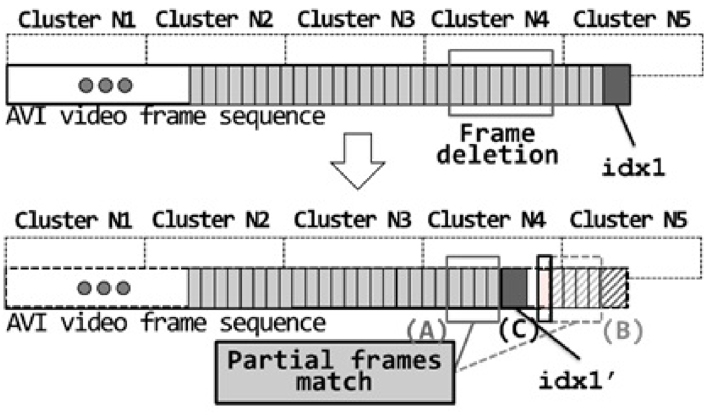

Fig. 6. Frame deletion scenario (Overlapping)

• **Scenario 2: Tail cut**

In this scenario, the deleted frames are located at the end of a frame sequence. As indicated in **Fig. 7**, there is no matching frame between stored video file (A) and residual space (B). When the frame deletion is conducted following our scenario, the new AVI frames index area `idx1′` is located at the end of the new frame sequence.

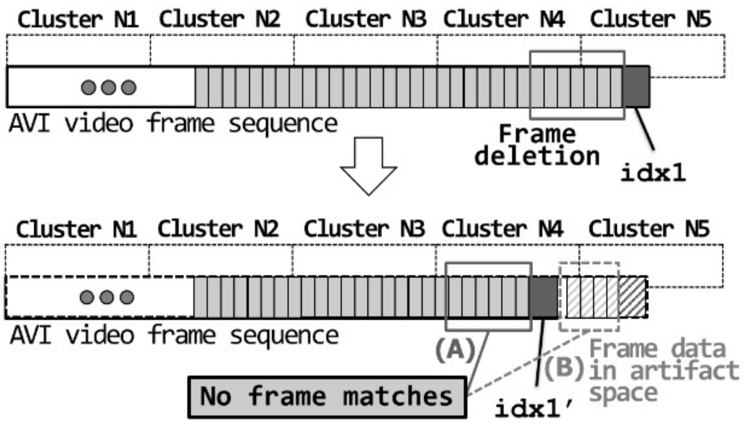

Fig. 7. Frame deletion scenario (Tail cut)

4.2 Integrity Verification of AVI Video File in FAT 32

In the case of AVI video files, it is possible to use the index information to verify integrity. According to Microsoft [14], the AVI format comprises `LIST hdrl`, `LIST movi`, and `idx1`. Byte string `idx1` indicates a starting point to a field listing AVI frame indices. Because `idx1` follows `LIST movi`, it is located at the tail portion of the AVI video file.

Fig. 8 graphically illustrates the concept of index-based verification. If a suspect overwrites a forged video file with an original video file, two `idx1` fields will remain in the file allocation area. Because the forged video usually has fewer frame indices than the original video [13], another `idx1` field remains in the residual space. We call the idx1 field stored in the residual space `idx1′`.

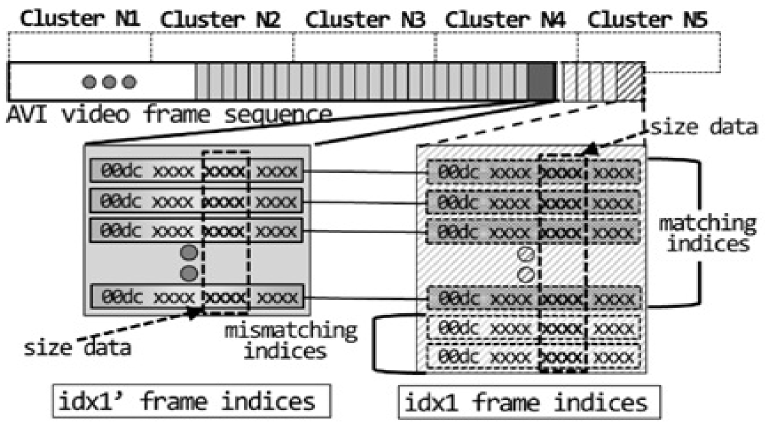

Fig. 8. Comparison of frame indices in both idx1 and idx1'

To verify the integrity of an AVI file, we refer to the size data of every frame index retrieved from `idx1` and `idx1′`. Let us suppose that the suspect deletes part of the frames. The frame size data in all the frame indices from `idx1′` are partially identical to those from `idx1`. In this situation, any mismatching frame between `idx1` and `idx1′` means the existence of frame manipulation in the video file. From this principle, we are able to ascertain that the integrity of the AVI is broken.

5. Evaluation

Table 1 displays the comparison of integrity verification capabilities among pre-deployment approaches, image processing-based schemes, our previously proposed scheme, and the proposed index-based scheme. As a proof of concept, we evaluated the proposed scheme with a self-implemented prototype and AVI video files. To add forgery to the videos, we used a commercial video editing tool that supports a frame-editing technique. In the experiment, the proposed scheme successfully detected videos forged in both overlapping and tail cut scenarios. Note that it does not require any pre-deployment of the algorithm, even though it detects forged videos in both scenarios.

Table 1. Comparison of integrity verification capabilities of various proposed schemes

Scheme	Frame-editing detection		Image editing detection	Pre-deployment
	Overlapping detection	Tail cut detection		
Kim *et al.* [9]	Yes	Yes	Yes	**Required**
Jayamalar *et al.* [10]	Yes	Yes	Yes	**Required**
Wang *et al.* [11]	No	No	Yes	Not Required
Hsu *et al.* [12]	No	No	Yes	Not Required
Lee *et al.* [13]	Yes	No	No	Not Required
Proposed	**Yes**	**Yes**	No	**Not Required**

Table 2. VEDR video files used in the detection test

Sample	Time length	Resolution	# of Frames
V1	60 sec	1920 × 1280	1,809
V2	60 sec	1920 × 1280	1,810
V3	60 sec	1280 × 720	1,800
V4	60 sec	1280 × 720	1,800

The proposed scheme was applied to video contents in two commercial automobile VEDRs, *iPass Black ITB-100HD* from *iTronics Corporation* and *Provia4UFHD* from *Provia Corporation.* Both of the VEDRs are equipped with a FAT32 formatted 8 GB flash memory card.

The recorded video files were stored on the SD flash memory card in AVI format. Note that the proposed integrity verification method is not restricted to AVI format only. We tested 20 video samples, which included four original videos and 16 manipulated videos with each frame-editing scenario. The four original videos are listed in **Table 2**. Two of the original videos were one-minute duration, 1920 × 1080 resolution, 30 fps, H.264 encoding format, and consisted of approximately 1,800 video frames. The other two original videos had the same attributes as the former two original videos except that they had a 1280 × 720 resolution. We labeled the original video samples ***V1*** to ***V4***. The manipulated samples were labeled ***O1*** to ***O4*** and ***T1*** to ***T4*** with frame deletion scenarios **O**verlapping and **T**ail cut, respectively. The subscripts "10" and "50" on the forged samples indicate the percentage of deleted frames.

The detection mechanism used to identify the manipulated video file is simple. When the verification tool found both matched frame indices and mismatched frames, it determined that the target file was a manipulated sample. Conversely, if only matched frames were identified, it was determined that the sample was cloned from the original, however, not forged. The original video samples, ***V1*** to ***V4,*** were found to be genuine when the detection mechanism was applied.

- **Overlapping scenario**: In the overlapping scenario, we deleted 180 frames (10% of total) and 900 frames (50% of total) from a random position of ***V1*** to ***V4*** and reorganized the video file to convert them into playable videos ***O1*** to ***O4***. From the detection results in **Table 3**, it is clear that the identical frame indices of the manipulated samples, ***O1*** to ***O4,*** were detected from the residual space and the indices of the deleted or overlapped frames were estimated as the mismatched frames. According to the detection results, we detected all eight overlapping deleted video files (***O1*** to ***O4***).
- **Tail cut scenario**: In the tail cut scenario, we deleted 180 frames (10% of total) and 900 frames (50% of total) from the end of ***V1*** to ***V4***, then constructed ***T1*** to ***T4***. According to the detection results, we detected all eight tail-cut deleted video files (***T1*** to ***T4***).

Table 3. Result of Integrity verification

Original sample	Original frames	Overlap sample	Matched frames	Mismatched frames	Forgery detected	Tail-cut sample	Matched frames	Mismatched frames	Forgery detected
V1	1,809	***$O1_{10}$***	**1,629**	**180**	**Yes**	***$T1_{10}$***	**1,629**	**180**	**Yes**
V1	1,809	***$O1_{50}$***	**935**	**874**	**Yes**	***$T1_{50}$***	**909**	**900**	**Yes**
V2	1,810	***$O2_{10}$***	**1,630**	**180**	**Yes**	***$T2_{10}$***	**1,630**	**180**	**Yes**
V2	1,810	***$O2_{50}$***	**935**	**875**	**Yes**	***$T2_{50}$***	**910**	**900**	**Yes**
V3	1,800	***$O3_{10}$***	**1,620**	**180**	**Yes**	***$T3_{10}$***	**1,620**	**180**	**Yes**
V3	1,800	***$O3_{50}$***	**900**	**900**	**Yes**	***$T3_{50}$***	**900**	**900**	**Yes**
V4	1,800	***$O4_{10}$***	**1,620**	**180**	**Yes**	***$T4_{10}$***	**1,620**	**180**	**Yes**
V4	1,800	***$O4_{50}$***	**900**	**900**	**Yes**	***$T4_{50}$***	**900**	**900**	**Yes**

6. Discussion

6.1 Time Consumption and Scalability

For digital forensics investigators, the time consumption of an integrity verification scheme is one of the important criteria in terms of practicality [19-21]. The size of a target video file can be large, up to several gigabytes. To demonstrate the scalability of the proposed verification scheme against an increase of the target video file size, we analyzed the time consumption of the proposed scheme with larger video samples and more deleted frames. The experiments are performed on a desktop PC with an Intel i5 3.4 GHz CPU, 10 GB RAM, Windows 8, and USB 3.0 interface for reading the SD card storages.

According to empirical experiments, the proposed index-based integrity verification scheme had a linear time complexity increase with the target video file size and the number of deleted frames. In a larger file, file I/O time requires a more significant portion of the entire time consumption and the file I/O time increases linearly with the increase of target file size. The verification time consumption T_v consists of *the residual space search time* T_r, *the index comparison time* T_c, and *file read and parsing I/O time* T_{io}. T_r is only affected by the number of deleted frames. As the size of the video file increases, T_c and T_{io} increase linearly, and the portions of T_{io} in T_v occupy a more dominant portion. In 10 MB, 100 MB, and 300 MB files, the proportions of T_{io} in T_v were 66%, 87%, and 93%, respectively. The overall manner of T_v is indicated in **Fig. 9**.

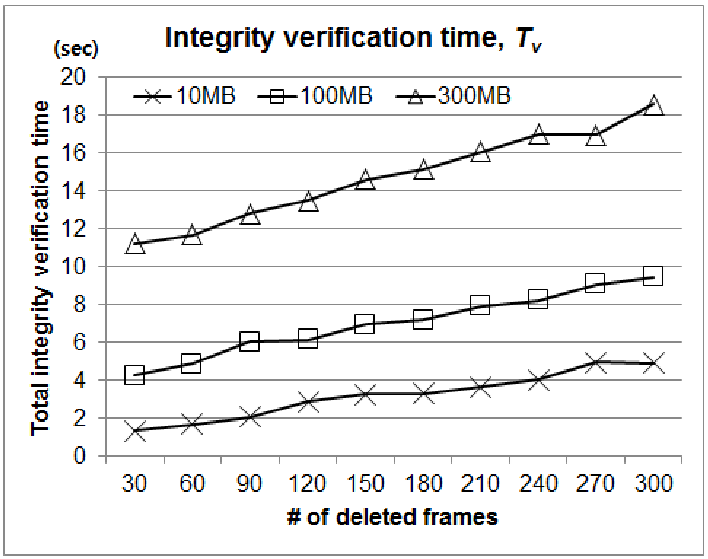

Fig. 9. Time consumption for integrity verification as increase of target video file size and the number of deleted frames

6.2. How Many Frames Can Be Recovered?

For digital forensics investigators, in addition to integrity verification, determining how many frames and the location in disk storage from which they can be recovered are also interesting issues [22-23]. In our scenarios, we created a scheme associated with frame recovery by calculating the size of the recoverable frame area.

The size of recoverable area ***r*** can be determined using **(1)**, where j, k, l_o, and l_f, denote the number of frames from the deletion starting point to the end of the file, the number of deleted frames, and the size of the frame index area on the original (l_o) and forged file (l_f), respectively. Further, $S(x)$ denotes the size of x frames.

$$r = S(j) - S(k) - l_o - l_f \tag{1}$$

For example, if 300 frames are deleted (k = 300) at 420 frames from the end of a frame sequence (j = 420), exactly 120 frames $(S(j - k) = S(120))$ will remain in the residual space. However, the size of the recoverable area in the residual space is not exact, $S(120)$, and a small number of the frames are not recoverable. Because two frame index areas remain at the end of the frame sequences of both an original and a forged video, the size of the area containing the recoverable frames is less than the 120 frames calculated. Therefore, the recoverable area $\boldsymbol{r}_{ex}$ in this case is calculated as $\boldsymbol{r}_e = S(420) - S(300) - l_o - l_f$. In practice, $l_o + l_f$ has a length less than $S(2)$ and the number of recoverable frames is 118 to 119 frames out of the deleted 120 frames.

6.3 Is the Integrity Verification Scheme Applicable to Other Video Files?

The index-based integrity verification scheme is applicable to video files depending on the video file structure. An MP4 video file, for example, is composed of sequential media information boxes. The actual frame data is located in mdat (media data box), as indicated in **Fig. 10**. Because there is no index list area in the MP4 file structure, the index-based verification method is not applicable to MP4 video files. Thus, a frame-based verification method, such as that proposed in our previous work [13], would be able to verify the integrity of MP4 video files.

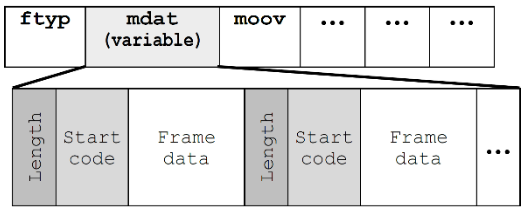

Fig. 10. Basic architecture of the MP4 file format

Windows Media Video (WMV) and Advanced Systems Format (ASF) have file structures that are similar to that of AVI. They are composed of various objects, i.e., header, data, and index objects sequentially. Because WMV and ASF have an index field area at the last part of the file structure, an index-based verification method is applicable to these video formats. Further, the index-based verification method can be applied to other video files where the video formats contain an index field that remains in the residual space after frame deletion.

6.4 Does the Proposed Integrity Verification Method Work in File Systems Other Than FAT32?

Storage systems such as SD cards and USB interface storage used by the majority of VEDR devices use the FAT32 file system. FAT32 stores files in sequential sectors. This sequential property is beneficial to index-based verification schemes. Index-based verification schemes that use the information in residual spaces are limited to file systems that have the non-sequential property. In other file systems such as Extended File system (EXT), because a file is stored fragmented into blocks with relative references, the location and content of the index field in disk storage is not trustworthy. That is, a pair of frames in residual space and the related video file stored in the disk space are not determinable.

7. Conclusion

This study proposed a video file-integrity verification scheme for investigating VEDR records. The proposed scheme detects image frame editing involving frame deletion by checking anomalies in the frame index fields. The proposed index-based scheme is a post-processing approach that operates without the requirement for pre-deployment on the VEDR device. Further, it detects video file manipulation performed using a frame-editing technique, which is difficult using conventional image-based approaches. We demonstrated the verification capability of the proposed scheme in two frame manipulation scenarios with AVI video files recorded using real-world VEDR devices. The evaluation results confirmed that the proposed scheme could be useful in scenarios where maintaining reliable video records is essential for legal evidence.

References

[1] Grand View Research, *North America car DVR market analysis by product (single channel, dual channel) and segment forecasts to 2016*, 2008.

[2] Evening Standard, *How car's black box trapped speeding Rich List heir who left baby paralysed in Range Rover crash*, 2008.

[3] M. Kim and M. Rick, *Expert: Digital evidence just as important as DNA in solving crimes*, 2008.

[4] R. Poisel and S. Tjoa, “Forensics Investigations of Multimedia Data: A Review of the State-of-the-Art,” in *Proc. of 6th IEEE International Conference on IT Security Incident Management and IT Forensics (IMF)*, pp.48-61, May 10-12, 2011. Article (CrossRef Link)

[5] Fox News, *Proposed new federal rule could put ‘big brother’ in your driver’s seat*, 2013.

[6] R. Poisel and S. Tjoa, “Roadmap to approaches for carving of fragmented multimedia file,” in *Proc. of 6th IEEE International Conference on Availability, Reliability and Security (ARES).*, pp.752-757, August 22-26, 2011. Article (CrossRef Link)

[7] E. Casey, *Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet*. Academic Press, 2011.

[8] S. L. Garfinkel and M. McCarrin, "Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb," *Digital Investigation*, vol. 14, pp. S95-S105, 2015. Article (CrossRef Link)

[9] M. Kim and K. Kim, “Data Forgery Detection for Vehicle Black Box,” in *Proc. of IEEE Information and Communication Technology Convergence (ICTC)*, pp.636-637, October 22-24, 2014. Article (CrossRef Link)

[10] T. Jayamalar and V. Radha, “Survey on digital video watermarking techniques and attacks on watermarks,” *International Journal of Engineering Science and Technology*, vol. 2, no. 12, pp. 6963-6967, 2010.

[11] J. Wang, G. Liu, Z. Zhang, Z. Wang and Y. Dai, “Detection of forgery in digital video based on pattern noise,” *Journal of Southeast University (Natural Science Edition)*, no. S2, 2008.

[12] C. Hsu, T. Hung, C. Lin and C. Hsu, “Video Forgery Detection Using Correlation of Noise,” in *Proc. of 10th IEEE Workshop,of Multimedia Signal Processing*, pp.170-174, October 8-10, 2008. Article (CrossRef Link)

[13] S. Lee, J. Song, W. Lee, Y. Ko and H. Lee, “Integrity Verification Scheme of Video Contents in Surveillance,” *IEICE TRANSACTIONS on Information and Systems*, vol. 98, no. 1, pp. 95-97, 2015. Article (CrossRef Link)

[14] Microsoft, *AVI RIFF file reference*.

[15] B. Carrier, *File System Forensic Analysis*, Addison-Wesley Professional, 2005.

[16] W. Minnaard, “The Linux FAT32 allocator and file creation order reconstruction,” *Digital Investigation*, vol. 11, no. 3, pp. 224-233, 2014. Article (CrossRef Link)

[17] W. Y. Lee, H. Kwon and H. Lee, “Comments on the Linux FAT32 allocator and file creation order reconstruction [Digit Investig 11 (4), 224–233],” *Digital Investigation*, vol. 15, pp. 119-123, 2015. Article (CrossRef Link)

[18] W. Wang and F. Hany, “Exposing digital forgeries in video by detecting duplication,” in *Proc. of the ACM 9th workshop on Multimedia & Security*, pp.35-42, September 20-21, 2007. Article (CrossRef Link)

[19] S. Becker, A. Brogi, I. Gorton, S. Overhage, A. Romanovsky and M. Tivoli, *Towards an engineering approach to component adaptation*, Springer Berlin Heidelberg, 2006. Article (CrossRef Link)

[20] D. Billard and R. Hauri, “Making sense of unstructured flash-memory dumps,” in *Proc. of the ACM Symposium on Applied Computing*, pp.1579-1583, March 22-26, 2010. Article (CrossRef Link)

[21] V. L. L. Thing, T. W. Chua and M. L. Cheong, “Design of a digital forensics evidence reconstruction system for complex and obscure fragmented file carving,” in *Proc. of the 7th International Conference on Computational Intelligence and Security*, pp.793-797, December 3-4. 2011. Article (CrossRef Link)

[22] L. Huston, R. Sukthankar, J. Campbell, and P. Pillai, “Forensic video reconstruction,” in *Proc. of the ACM 2nd International Workshop on Video Surveillance & Sensor Networks*, pp.20-28, October 10-16, 2004. Article (CrossRef Link)

[23] G. H. Na, K. S. Sim, K. W. Moon, S. G. Kong, E. S. Kim and J. Lee, “Frame-based recovery of corrupted video files using video codec specifications,” *IEEE Transactions on Image Processing*, vol. 23, no. 2, pp. 517-526. 2014. Article (CrossRef Link)

Choongin Lee received a B.S. degree in Computer Science from Korea University, Korea, in 2015. He is currently pursuing a doctorate degree in Computer and Communication Security at Korea University, Korea. His research interests are digital forensics, software vulnerability analysis, and protocol vulnerability detection.

Jehyun Lee is a researcher at the Cyber Security Research Center (CSRC), Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. He received his B.S., M.S., and Ph.D degrees in Computer Science and Engineering at Korea University, Korea in 2007, 2009, and 2015, respectively. His research interests include network security and malware.

Youngbin Pyo received a B.S. degree in Computer Science and Engineering from Korea University, Korea, in 2007 and 2014. He is currently pursuing an M.S. degree with the Department of Computer and Radio Communication Engineering, Korea University, Korea. His research interests include digital forensics and image processing.

Heejo Lee is a professor at the Department of Computer Science and Engineering, Korea University, Seoul, Korea. Before joining Korea University, he was at AhnLab, Inc. as CTO from 2001 to 2003. From 2000 to 2001, he was a postdoctorate at the Department of Computer Sciences and security center CERIAS, Purdue University. He received his B.S., M.S., and Ph.D. degrees in Computer Science and Engineering from POSTECH, Pohang, Korea. He serves as an editor of both the Journal of Communications and Networks and the International Journal of Network Management.