LETTER *Special Section on Enriched Multimedia*

Integrity Verification Scheme of Video Contents in Surveillance Cameras for Digital Forensic Investigations*

Sangwook LEE[†], **Ji Eun SONG**[††], **Wan Yeon LEE**[††], ***Nonmembers***, **Young Woong KO**[†††], ***and*** **Heejo LEE**[†a)], ***Members***

SUMMARY For digital forensic investigations, the proposed scheme verifies the integrity of video contents in legacy surveillance camera systems with no built-in integrity protection. The scheme exploits video frames remaining in slack space of storage media, instead of timestamp information vulnerable to tampering. The scheme is applied to integrity verification of video contents formatted with AVI or MP4 files in automobile blackboxes.

key words: *digital forensics, integrity, surveillance camera, video frame*

1. Introduction

Recently, surveillance cameras such as CCTV and automobile blackbox (i.e., in-car dashboard camera recoder) have been increasingly popular. In criminal investigation or accident site examinations, video contents stored in storage media of surveillance cameras frequently include critical scenes and can provide crucial evidence. Hence integrity verification of video contents in surveillance cameras is very important for digital forensic investigations [1]. However, many commercial surveillance camera systems do not support any integrity protection scheme such as digital signature and digital watermarking [2] due to their cost increment. For these legacy surveillance camera systems with no built-in integrity protection, the proposed scheme verifies the integrity of video contents.

Timestamp information in metadata of file systems [3] can be utilized for integrity verification, if the metadata including the creation time and the last modification time is reliable. When the creation time is different from the last modification time, its integrity is broken. However, the metadata is vulnerable to tampering and thus unreliable. The timestamp is generated based on a system internal clock, and the current internal clock is easily set to the creation time of video contents.

Instead of timestamp information vulnerable to tampering, the proposed scheme exploits residual data in unused slack space of storage media of legacy surveillance cameras. When modified video contents are overwritten upon original video contents, the tail part of the original video contents remains in slack space of storage media because the size of the modified video contents is usually smaller than that of the original video contents. The proposed scheme extracts video frames in the slack space, and compares them with video frames extracted from allocated space of storage media. If the video frame in the slack space is equal with that in the allocated space, the scheme determines that the integrity of video contents including the extracted video frame is broken.

As a case study, the scheme is applied to integrity verification of video contents in commercial automobile blackboxes prevalently used nowadays, where the video contents are formatted with AVI or MP4 files. Note the proposed scheme is not limited to AVI and MP4 formats, but eligible for other video file formats.

2. Preliminaries

Video contents only in storage media of surveillance cameras are lawfully approved [1], [3]. Video contents consist of sequential static images, called *video frames*. Typically video contents for one second consist of about 30 video frames. Video contents recorded for a long time are generally divided into multiple video file, so that each video file has video contents recorded for a fixed short time.

The video files are managed by file schemes [3] that provide a mechanism to store, categorize and access data of video frames upon storage media such as SD flash memory card and magnetic tape. Figure 1(a) shows an example of three video files managed by a file system in storage media. The minimum unit logically addressable is called *cluster* (or *sector*). Suppose that file A, B and C occupy two, three and two clusters, respectively. When the size of a file is not a multiple of the cluster size, the unused space in the last cluster is called *slack space*. Figure 1(b) shows a case of overwriting the modified file B, whose name is underlined for differentiation. Boxes filled with dense gray denote modified files, while boxes filled with light gray denote unmodified original files. Stored area of a modified file depends on the file allocation strategy [3]. The modified file B is stored either at the same start cluster with the original file B as shown in Fig. 1(b) or at the next available cluster as shown in Fig. 1(c). In this study, we focus on the former case in Fig. 1(b) because the integrity verification for the former

Manuscript received March 17, 2014.

[†]The authors are with Korea University, South Korea.

[††]The authors are with Dongduk Women's University, South Korea.

[†††]The author is with Hallym University, South Korea.

*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).

a) E-mail: heejo@korea.ac.kr

DOI: 10.1587/transinf.2014MUL0001

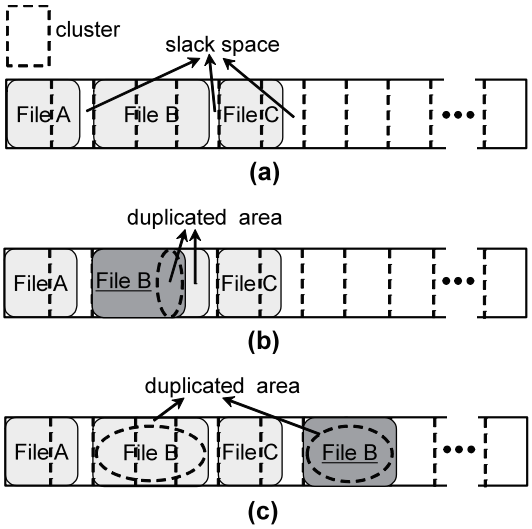

Fig. 1 Example of video files managed by file system.

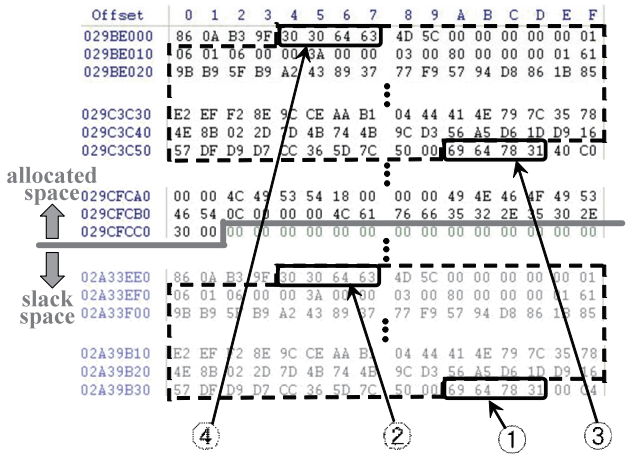

Fig. 2 Integrity verification upon AVI video file.

is more restricted and complex than that for the latter.

If the size of a modified video file is smaller than that of its original video file, the tail part of the original video file in slack space is duplicated with the tail part of the modified file, as shown in Fig. 1(b). Deleting a part of video frames clearly reduces the size of video files. Modifying video frames also reduces the size of video files, because common video manipulating tools such as Sony Vegas, Adobe Premier and Avid Studio require re-compression of modified video frames, referred to as re-rendering. This study assumes that the last video frame in original video files is not overwritten and residual data in unused slack space is not wiped.

3. Case Study of Proposed Scheme

In the first phase, the proposed scheme extracts the last video frame in slack space based on the format specifications of video files. In the second phase, the scheme extracts the last video frame of a normal video file stored in allocated space similarly. In the third phase, the scheme compares the two extracted video frames for integrity verification. If the two video frames are equal, the scheme determines that the video contents including the extracted video frame in the allocated space were overwritten after modification.

As for comparison unit between slack space and allocated space, the scheme chooses the last video frame because the last video frame does not include critical scenes and thus is not modified usually. It is an optional process to extract and compare more video frames in slack space and in allocated space for more rigorous integrity verification.

As a case study, the proposed scheme is applied to integrity verification of video contents in two commercial automobile blackboxes, Black FXD700 Mach of Inavi corporation and I-Pass Black ITB-100HD of Itronics corporation. Video files in SD flash memory card of the former blackbox are formatted with AVI file, and those in the latter are formatted with MP4 file. *WinHex* forensic tool is used for examining all binary codes in the SD flash memory card.

To evaluate the proposed scheme, an arbitrary video file on storage media is selected, modified and overwritten. About 30 video frames in the selected video file are deleted using Virtualdub program running on Window 7 operating system. 30 video frames construct video contents with about one second running time. When commercial automobile blackboxes are shocked by external crashes, their recording function often stops only for a while and works well soon. It is very hard to distinguish whether disappearance of some video frames comes either from malfunctions of automobile blackboxes or from intentional modifications of original video files.

3.1 Integrity Verification upon AVI Video Files

Figure 2 shows a detection case of integrity violation of the proposed scheme upon the AVI video file format, which is a partial view of the *WinHex* forensic tool. The proposed scheme extracts video frames based on the AVI file format specifications [5]. The AVI file format consists of four parts: *RIFF* containing AVI file type identifier, *hdrl* containing metadata of this file, *movi* containing data stream of video contents, and *idx1* containing the index of each video frame. The location sequence of these four parts is fixed as *RIFF*, *hdrl*, *movi* and *idx1*.

In the first phase, the scheme searches for the start indicator of the *idx1* part in slack space because video frames are located in front of the *idx1* part. The start indicator is 0x69647831 (i.e., “idx1” in ASCII code), which is arrowed with label ‘1’ in Fig. 2. In the backward direction from the start indicator of the *idx1* part, the scheme searches for the start indicator of the video frame. The start indicator of video frame is 0x30306463 (i.e., “00dc” in ASCII code), which is arrowed with label ‘2’ in Fig. 2. The four bytes following the start indicator (i.e., 0x4D5C0000) are the length of this frame. Comparing this length information (i.e., 0x00005C4D in little endian) and the traversing distance, it can be checked whether the found indicator is a real frame indicator or a part of data accidentally matched.

In the second phase, the scheme extracts the last video frame of a normal video file stored in allocated space, similarly with the extraction process in the first phase. The found indicators in the second phase are arrowed with labels '3' and '4' in Fig. 2. In the third phase, the scheme compares the two video frames extracted in the slack space and in allocated space. The extracted frames are depicted with broken lines in Fig. 2. If the two video frames are equal, the scheme concludes that the video file including the extracted frame is modified.

The probability that two independent parts are equal by accident is $(\frac{1}{256})^s$, where s is the byte size of the two parts. The probability for the video frame with a byte size of about $20K$ in Fig. 2 is approximately $(\frac{1}{256})^{20,000} < \frac{1}{10^{40,000}}$.

3.2 Integrity Verification upon MP4 Video Files

Figure 3 shows a detection case of integrity violation of the proposed scheme upon the MP4 video file format with H.264 codec. The MP4 file format consists of three parts: *ftyp* containing MP4 file type identifier, *moov* containing metadata of this file, and *mdat* containing data stream of video contents. The *ftyp* part is always located in the file front, and the *mdat* part is located in the file tail in most cases.

In the first phase, the scheme searches for the start indicator of H.264 video frames in the backward direction from the last of slack space. The start indicator is 0x0000????41 which is arrowed with label '1' in Fig. 3, where '?' is a don't care term that can hold any value. The former four bytes (i.e., 0x0000????) means the frame length. Because the frame length is usually smaller than 2^{17}, the first two bytes in big endian are 0. The last byte (i.e., 0x41) means the video frame type. There are three types of video frames in H.264 codecs: IDR frame, B frame and P frame. IDR frame is denoted as 0x61, B frame is denoted as 0x01, and P frame is denoted as 0x41. The type of the last frame in video contents is usually P frame. Using the length information (i.e., 0x000016F4 in big endian), the scheme extracts the last video frame from the slack space. The extracted frame is depicted with broken lines.

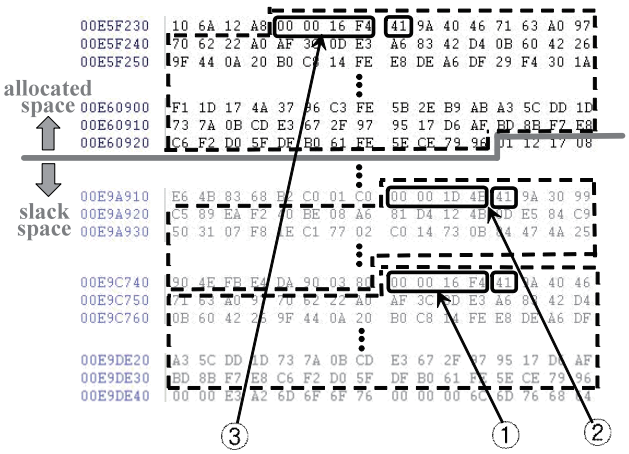

Fig. 3 Integrity verification upon MP4 video file.

It is possible that the found indicator is a part of data. To check whether the found indicator is a real frame indicator or a part of data, the scheme searches another start indicator of video frames, which is arrowed with label '2' in Fig. 3. Comparing the length information (i.e., 0x00001D4B in big endian) and the traversing distance, the two found indicators (i.e., 0x00001D4B41 and 0x000016F441) are determined to be correct or not.

In the second phase, the scheme searches for the start indicator from the last of allocated space. The start indicator is 0x000016F4, which is arrowed with label '3' in Fig. 3. Comparing the length information (i.e., 0x000016F4 in big endian) and the traversing distance, the found indicator is determined to be correct or not. In the third phase, the scheme compares the last video frame extracted from the slack space and that from the allocated space. If the two video frames are equal, the scheme concludes that the video file including the extracted frame is modified.

The probability that the two extracted video frames are equal by accident is approximately $(\frac{1}{256})^{4,000} < \frac{1}{10^{8,000}}$ for the video frame with a byte size of about $4K$ in Fig. 3. Besides H.264 codec, the scheme is applicable to other codecs through frame header interpretation based on codec specifications [6].

4. Conclusions and Discussion

The proposed scheme verifies the integrity of video contents in legacy surveillance cameras with no built-in integrity protection. The scheme exploits video frames extracted from unused slack space of storage media, instead of timestamp information vulnerable to tampering. Although the scheme does not guarantee to detect all integrity violations, it enhances the detection capability of integrity violations for video contents with no built-in integrity protection. Perfect detection of all integrity violations is inherently unachievable without support of built-in integrity protection scheme such as digital signature and digital watermarking.

References

[1] R. Posel and S. Tjoa, "Forensics investigations of multimedia data: A review of the state-of-the-art," Int'l Conf. IT Security Incident Management and IT Forensics (IMF), pp.48–61, 2011.
[2] I. Echizen, S. Singh, T. Yamada, K. Tanimoto, S. Tezuka, and B. Huet, "Integrity verification systems for video content by using digital watermarking," Int'l Conf. Service Systems and Service Management (ICSSM), pp.1619–1624, 2006.
[3] B. Carrier, File system forensic analysis, Addison-Wesley, 2005.
[4] C. Wootton, A practical guide to video and audio compression: From sprockets and rasters to macro blocks, Focal Press, 2005.
[5] http://msdn.microsoft.com/en-us/library/windows/desktop/dd318187(v=vs.85).aspx
[6] G.H. Na, K.S. Shin, K.W. Moon, S.G. Kong, E.S. Kim, and J. Lee, "Frame-based recovery of corrupted video files using video codec specifications," IEEE Trans. Image Process., vol.23, no.2, pp.317–326, Dec. 2013.