Video Authentication: A Self Embedding Steganography Approach

Pratheepan. Y[*], Joan V. Condell, Kevin Curran and Paul Mc Kevit
School of Computing and Intelligent Systems,
University of Ulster, Northern Ireland, UK.
p.yogarajah, j.condell, kj.curran, p.mckevitt@ulster.ac.uk

Abbas Cheddad
Umea Centre for Molecular Medicine (UCMM),
Umea Universitet, 901 87 Umea, Sweden.
abbas.cheddad@ucmm.umu.se

Abstract

Digital video has become increasingly susceptible to spatio-temporal manipulations as a result of recent advances in video editing tools. Therefore it is difficult to get video records to stand up as 100% secure evidence in court, for example for a criminal evidence.

In this paper, we propose a novel digital video authentication approach based on a secure self-embedding technique. Our approach allows authentication of the origin of the video data from the embedded information after reasonable spatial manipulations. In addition, temporal manipulations can also be detected using our approach.

1. Introduction

Video forgery is a technique for generating fake video by altering, combining, or creating new video content. Using commonly available hardware and software, it is now feasible to perform digital video editing not only to insert, delete or replace groups of frames, but also to insert or delete objects from those frames without introducing visible artifacts.

The ease of editing visual data in the digital domain has facilitated unauthorized tampering without leaving any perceptible traces. Therefore recorded CCTV (Closed-Circuit TeleVision) does not stand up in court as a 100% reliable evidence. Most of the research work carried out in the area of CCTV surveillance has dealt with object detection, object recognition, tracking, behaviour analysis and image retrieval. Not much work has been done in CCTV video forgery detection.

[*]This work was part funded by the Invest NI Proof of Concept fund.

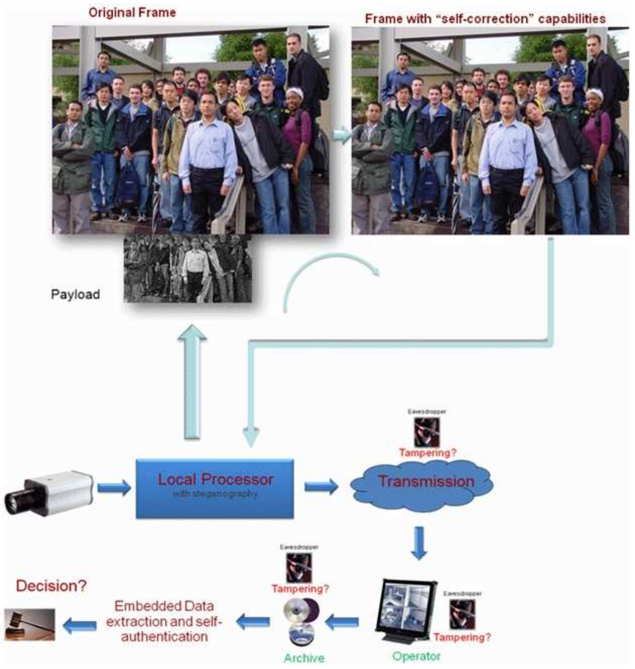


Figure 1. Our proposed method.

The main goal of our work is to prevent the possibility of creating a forgery that goes undetected. For example, a secure CCTV camera equipped with a watermarking/steganography chip may authenticate every image it takes before storing it on devices, see Figure 1. Such smart images may play an important role in detecting digital forgeries or establishing the origin of digital video content. The case and extent of such manipulations highlights the need for authentication and integrity verification mechanisms in applications such as video for court evidence.

Manipulations on video signals fall into two categories, [6, 14]. The first type covers attacks that tamper with the intensity patterns of the video, e.g. compression, noise, etc. The second type covers time-based tampering (such as frame inserting and dropping), disrupting the frame sequencing, e.g. frame cuts, swapping, deletion or foreign frame insertion and spatial-based attacks (such as removing and replacing content in a frame).

This paper is mainly focused on the second type of attacks. To detect the unauthorised manipulation of such video footages, a video authentication system should verify that the video taken has not been tampered with in the temporal or in the spatial domains. Authentication has always been an important issue [11]. Content authentication is a process by which a user is guaranteed that video content is original and has not been maliciously modified. One example is surveillance and site monitoring footage where incentives exist to remove incriminating material.

Based on the embedding of specific digital signatures (i.e. a set of authentication bits that summarise the image content), video may be viewed by authorized personnel but made unavailable to others. Self-embedding refers to the process where a compressed copy of the image/video is embedded into itself during watermarking. Most of the algorithms deal with image authentication, for example the work in [4, 9]. After self-embedding, it is possible to recover portions of the image that have been cropped, replaced, damaged etc. without accessing the original image.

Authentication of video is very similar to that of still images. In particular, we can evaluate a video as a sequence of still images where each image (or frame) is authenticated individually. Therefore we have developed a new video authentication method based on a secure and improved self-embedding algorithm proposed in [2]. Our method is discussed in Section 2. Section 3 explains the self-embedding approach. The secure image encryption algorithm is explained in section 4. Sections 5 and 6 explain result and provide discussion and conclusions respectively.

2. Video Authentication Method

Authentication of video is very similar to that of still images. In particular, we can evaluate a video as a sequence of still images where each image (or frame) is authenticated individually. Cheddad et al. [2] proposed a method to detect tampering within frames (i.e. spatial manipulations) using the image verification scheme. In addition to spatial manipulations, a common class of attacks on digital video is re-indexing, where the sequence of events is tampered. Since random ordering of authentic frames is not an authentic video sequence, the temporal location of frames should also be verified. Therefore we further developed Cheddad et al.'s [2] work to handle temporal attacks. As our main goal is to prevent the possibility of creating a forgery that goes undetected and establishing the origin of the digital video content, we embed the spatial and temporal information in the video sequence.

Video embedding is not the same as an image embedding as video contains the temporal information. We can consider a video file as an ordered collection of images. Embedding the spatial and temporal content in each image of a video file will be a time consuming task and also may be an unessential task in that most neighbouring image frames represent similar spatial content. Therefore instead of hiding spatial content in each image sequence, the spatial contents are only hidden in the detected key frames of image sequences. At the same time temporal contents are hidden in every image in the video file. Therefore each image is checked for key frames and if the key frame is detected then the temporal and spatial contents are embedded. Otherwise only the temporal content is embedded.

It is true that important events of the video file can be represented by a set of key frames. The key frames can be detected using shot change detection techniques. Therefore first the video is converted to an ordered collection of images and then the key frames are detected. Our key frame detection method is explained in Section 2.1.

2.1. Key frame detection

There are many algorithms available to detect shot boundaries within video sequences [10, 12, 13, 18]. These algorithms calculate different features of the video data, and can be used as stand-alone shot boundary detection systems. The algorithm [10] computes the average of the intensity values for each component (YUV, RGB, etc.) in the current frame and compares it with that for the following frame. In the algorithm [12], each region of the image is represented by second order statistics under the assumption that this property remains constant over the region.

The shot change boundary position detection based on histogram comparison methods [13, 18] are quite popular because they are fast and motion insensitive. Therefore we used a histogram based technique for the key frame detection process. The key frames are detected based on the shot change boundary positions in video sequences. In the method, a time series of discontinuity feature values $f(n)$ are calculated for each frame. The dissimilarity feature value for frame n is $f(n) = d(n-1, n)$. Here we use a function $d(n-1, n)$ to measure the dissimilarity between frame $n-1$ and n and it is measured by:

$$d(H_{n-1}, H_n) = \sqrt{\sum_{i=1}^{G} (H_{n-1}(i) - H_n(i))^2} \qquad (1)$$

Here H_n denotes the intensity histogram of the n^{th}

frame and i is one of the G possible intensity (grayscale) values. The shot boundary detection decision can be made based on a threshold of dissimilarity measure. We pick key frame positions from $f(n)$ based on the adaptive thresholding technique which will be explained in 2.2.

2.2. The adaptive thresholding technique

We have developed an adaptive thresholding technique for the adaptation process based on [7]. In [17], the authors apply a local thresholding method whereby the frame differences of successive m frames is examined. They then declare a shot change when two conditions are simultaneously satisfied: (1) the difference is the maximum within a symmetric sliding windows of size $2m - 1$. (2) the difference is n times the second largest maximum in the sliding window. Expanding this work, a method was proposed in which the means and standard deviations from either side of the middle sample in the window are calculated [7]. The middle sample represents a shot change if the conditions below are simultaneously satisfied: (1) the middle sample is the maximum in the window. (2) the middle sample is greater than $max(\mu_{left} + T_d\sqrt{\sigma_{left}}, \mu_{right} + T_d\sqrt{\sigma_{right}})$, where T_d is given a value of 5. The method is as follows.

$$m_T = \mu_N + T_d\sqrt{\sigma_N} \qquad (2)$$

We estimate the mean μ_N and variance σ_N of N dynamically, from the similarity measures m of M neighbouring frames. The decision threshold m_T is recalculated for each new frame using the above method and a decision is made. The key frames are detected based on the above histogram similarity and adaptive thresholding techniques. The origin of spatial content is embedded in the detected key frames. Our self-embedding approach is explained in Section 3.

3. Self-Embedding

Our self-embedding method consists of two parts, self-embedding and authentication. In the self-embedding phase, a binary version of the original content (also called payload) is embedded into the original image (also called the cover image). The payload includes spatial and temporal contents.

Since we need means of protecting the video files against forgery, it is essential that the payload will carry as much information from the cover image as possible. An approximation of the cover video image file can be achieved by applying the binarization techniques which results in a binary image demanding only 1 bit per pixel for storage. In our method, three different binary versions of the original image are considered as spatial payload, see Figure 2. Figure 2, (b), (c) and (d) represent payload of the cover image.

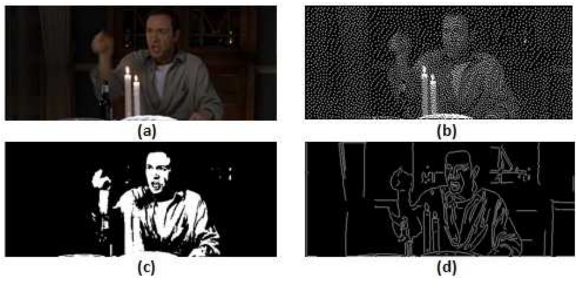


Figure 2. (a) original image. (b) created by dithering using the method described by Floyd and Steinberg [8], (c) created by thresholding the original image and (d) created by Canny edge operator respectively.

Dittman et al. [5] address the problem of embedding the temporal content by embedding the SMPTE (Society of Motion Picture and Television Engineers [15]) time code in each frame. Any alteration in the sequence of events is detected by checking the timing information. SMPTE offers a timing signal for a whole day (hh:mm:ss:ff, where f stands for frame) with a resolution down to single video frames. If a time code like SMPTE could be embbeded in a video stream as a watermark, every position frame in the stream would include a complete timing information, see Figure 3.

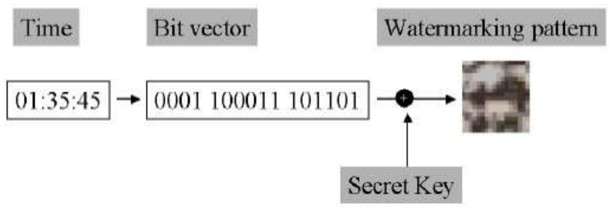
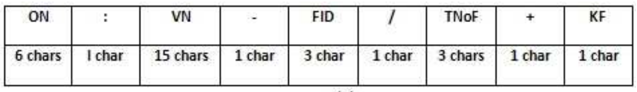


Figure 3. SMPTE representation [5].

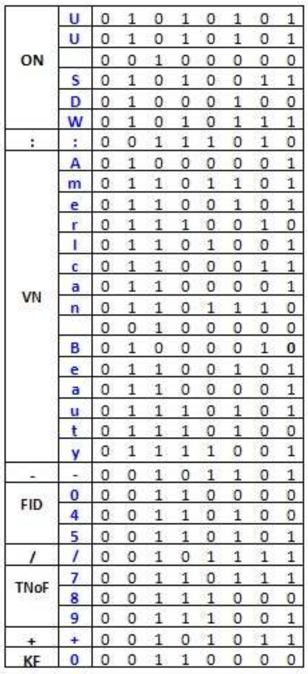
Instead of timing content embedding [5], we considered frame number (i.e. frame ID) with some additional information for embedding. In total 32 characters are allocated to represent temporal content for each frames. The fields, owner name (i.e. authorised body of the video) (ON), name of the video (NV), frame number (FID), total number of frames (TNoF) and key frame (KF), are allocated 6, 15, 3, 3 and 1 characters respectively. In addition 1 char is used to represent the symbols $:, -, /, +$, see Figure 4(a).

Here the symbols $:, -, /, +$ are used to separate each field. Each character is represented as an 8 bits binary representation, i.e. say, a temporal content, "**UU SDW:American Beauty-045/789+0**", is converted to an 8 bits binary representation, see Figure 4(b). Here **KF = 0** means that this frame is not a key frame and 1 means it is. Then the calculated **(32 x 8)** binary matrix is directly converted to a binary image, see Figure 4(c).



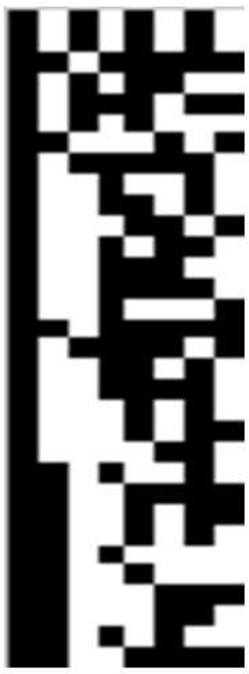
ON	:	VN	-	FID	/	TNoF	+	KF
6 chars	l char	15 chars	1 char	3 char	1 char	3 chars	1 char	1 char

(a)



ON	U	0	1	0	1	0	1	0	1
	U	0	1	0	1	0	1	0	1
		0	0	1	0	0	0	0	0
	S	0	1	0	1	0	0	1	1
	D	0	1	0	0	0	1	0	0
	W	0	1	0	1	0	1	1	1
:	:	0	0	1	1	1	0	1	0
VN	A	0	1	0	0	0	0	0	1
	m	0	1	1	0	1	1	0	1
	e	0	1	1	0	0	1	0	1
	r	0	1	1	1	0	0	1	0
	I	0	1	1	0	1	0	0	1
	c	0	1	1	0	0	0	1	1
	a	0	1	1	0	0	0	0	1
	n	0	1	1	0	1	1	1	0
		0	0	1	0	0	0	0	0
	B	0	1	0	0	0	0	1	0
	e	0	1	1	0	0	1	0	1
	a	0	1	1	0	0	0	0	1
	u	0	1	1	1	0	1	0	1
	t	0	1	1	1	0	1	0	0
	y	0	1	1	1	1	0	0	1
-	-	0	0	1	0	1	1	0	1
FID	0	0	0	1	1	0	0	0	0
	4	0	0	1	1	0	1	0	0
	5	0	0	1	1	0	1	0	1
/	/	0	0	1	0	1	1	1	1
TNoF	7	0	0	1	1	0	1	1	1
	8	0	0	1	1	1	0	0	0
	9	0	0	1	1	1	0	0	1
+	+	0	0	1	0	1	0	1	1
KF	0	0	0	1	1	0	0	0	0

(b)



(c)

Figure 4. (a) 32 bits allocation table. The binary representation of the 32 characters are shown in (b). (c) corresponding binary image.

Then the binary images calculated from the spatial and temporal contents (payloads) are embedded in the original image (cover image) based on the highly secure self-embedding algorithm proposed in [2].

4. Image Encryption Algorithm

This algorithm is explained based on [2] and the encryption algorithm is fully described in [3]. A hash function is more formally defined as the mapping of bit strings of an arbitrary finite length to strings of fixed length [16]. Here we attempt to extend SHA-1 (the terminology and functions used as building blocks to form SHA-1 are described in the US Secure Hash Algorithm 1, [1]) to encrypting digital 2D data. The introduction of Fast Fourier Transform (FFT) forms together with the output of SHA-1 a strong image encryption setting. Let the key bit stream be $\lambda_{k,l}$ where the subscripts k and l denote the width and height after resizing the keys bit stream respectively, i.e., 8, $M*N$, where M, N are the plain image's dimension.

The FFT will operate on the Discrete Cosine Transform (DCT) of $\lambda_{k,l}$ subject to Eq. 4.

$$f(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} F(x,y)e^{-2\pi i(xu+yv)/N} \tag{3}$$

where $F(x,y) = DCT(\lambda_{k,l})$ satisfying Eq. (4). Note that for the transformation at the FFT and DCT levels we do not utilise all of the coefficients. Rather, we impose the following rule, which generates at the end a binary random-like map. Given the output of Eq. 3 we can derive the binary map straightforwardly as:

$$Map(x,y) = \begin{cases} 1 & iff \quad f(u,v) > 0 \\ 0 & otherwise \end{cases} \tag{4}$$

This map takes the positive coefficients of the imaginary part to form the ON pixels in the map. Since the coefficients are omitted the reconstruction of the password phrase is impossible, hence the name Irreversible Fast Fourier Transform (IrFFT). In other words, it is a one way function which accepts initially a user password. This map finally is XORed with the binary version of each colour component separately. Another phenomenon that we noticed and we would like to exploit is the sensitivity of the spread of the FFT coefficients to changes in the spatial domain.

Therefore if we couple this with the sensitivity of the SHA-1 algorithm to changes of the initial condition, i.e., password phrase, we can easily meet the Shannon law requirements, i.e., confusion and diffusion. For instance a small change in the password string will, with overwhelming probability, result in a completely different hash and thus a different image by extension. So, the core idea here is to transform these changes into the spatial domain where we can apply 2D-DCT and 2D-FFT that introduce the aforementioned sensitivity to the two dimensional space. As such, images can be easily encoded securely with password protection.

The payloads are securely embedded using the the encryption algorithm explained above. Here the embedded image is called the Stego image. Each Stego image contains two parts: the Header and the Body, see Figure 5.

The temporal content is embedded in the Header part of every image in the video file. Here the temporal content size is fixed, **(32 x 8)** bits. So the size of the header part is allocated as **(4 x column size)**. Therefore the size of body part is allocated as **((rows-4) x column size)**. The spatial content, the binary version of the particular image, is only embedded in the body part of that image if that image is identified as a key frame.

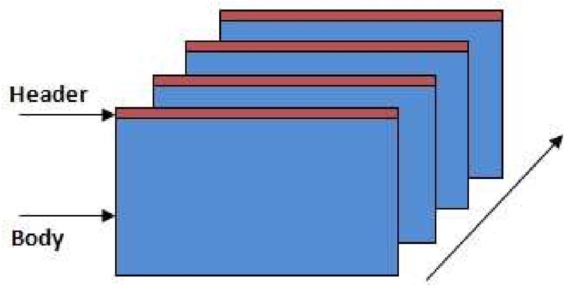


Figure 5. Header and body representation of the Stego images.

5. Results and Discussion

In this paper, we have developed key frame detection and video authentication methods. We have downloaded "**American Beauty - 01500.avi**" video clip[1] for our experiments. This video is 24.64 seconds long, frame rate is 25 frames/second and width and height sizes are 528x224 pixels.

5.1. Key Frame Detection

In this experiment, T_d and the window size are set to 5 and 20 respectively. The key frames positions in the above mentioned video sequence are shown in Figure 6 and the detected key frame images are shown in Figure 7.

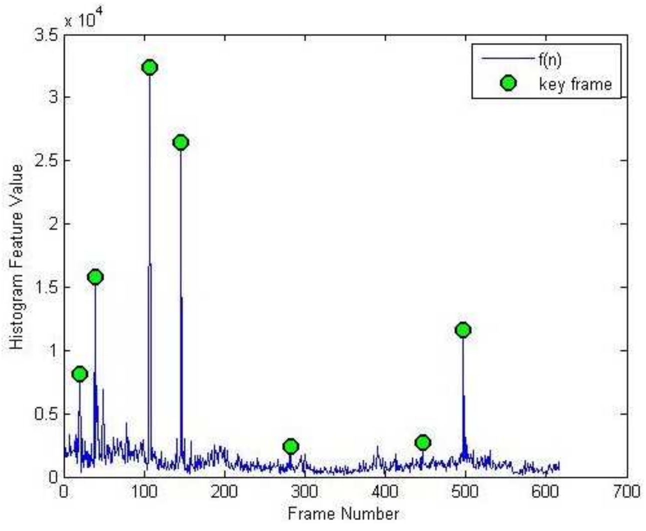


Figure 6. Key frame detection graph.

[1]http://lear.inrialpes.fr/people/marszalek/data/hoha/hollywood.tar.gz, last visit: 19th Aug 2010.

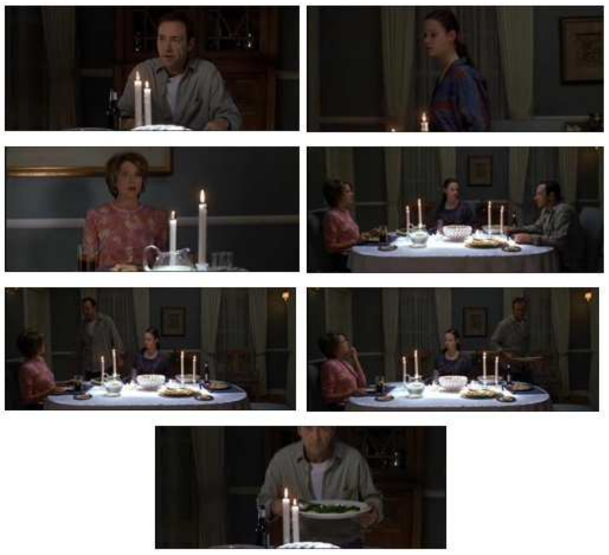


Figure 7. The detected key frames from "American Beauty - 01500.avi".

5.2. Video Authentication

Temporal content can be extracted from the stego image. Any removal of a frame or position changes can be detected because the flow of the frame number (FID) is distorted. Moreover TNoF denotes the total number of frames in the sequences. Based on this, we can identify how many frames are included as extra or deleted from the video sequence. KF field denotes whether the image is key frame or not. In addition ON and NV give more details about the video file.

Spatial content can be extracted only from the key frame. We analyse the spatial attack using an object insertion within the stego image. The attacked stego image is represented by Figure 8(b). Here a new object (i.e. the girl) is inserted into the stego image. Figures 8(c), (e) and (g) represent correspondence payload images and Figures 8 (d), (f) and (h) represent the correspondence extracted payloads from the attacked stego image.

As an objective measure, the Mean Squared Error (SME) between the payload and extracted payload images is calculated. The parameters are given by:

$$MSE = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}(P(i,j) - EP(i,j))^2 \qquad (5)$$

where $P(i,j)$ and $EP(i,j)$ are the pixel values at row i and column j of the payload and extracted payload images respectively. The measured values are shown in Table 1. Even though there is not much difference in measured values between dither, edge and thresholded binary images, thresholded representation of payload embedding provided good

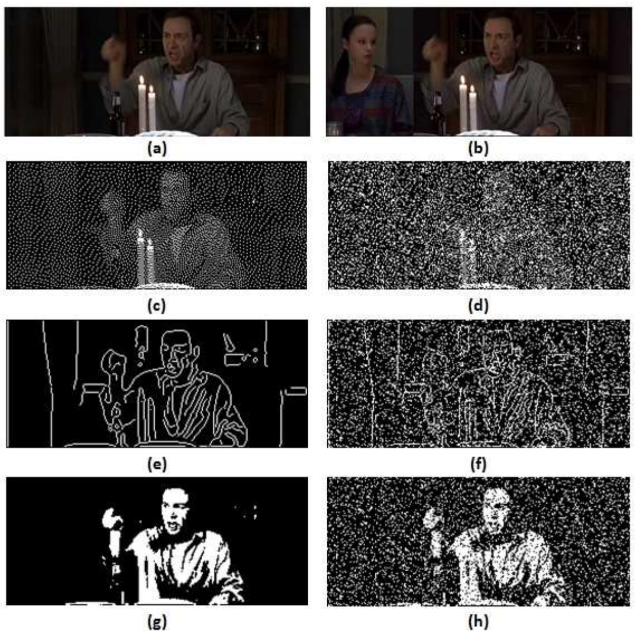


Figure 8. (a) Cover image. (b) attacked stego image. (c), (e) and (g) represent payloads and (d), (f) and (h) represent extracted payload after the spatial attack. These payloads are extracted only from attacked stego image without accessing the original image.

visible effect of the original spatial content. This experiment shows that the representation of payload (i.e. binary version of the original content) was more important in our approach.

Table 1. Mean Square Error

Binary Image	MSE
Dither	0.1420
Edge	0.1413
Thresholded	0.1398

6. Conclusions

We have proposed a novel approach to prove the original content from the forged video which uses an information hiding technique, [2], that is highly secure. Our method can handle both spatial and temporal attacks. Binary representation of the original spatial content is embedded only into the key frames. Experimental results showed that extracted thresholded payload from attacked stego image shows much more visible original content than edge and dither payloads.

The main drawback of our method is that if any key frame is lost due to the temporal attack then we could not recover the original content. This is the focus of future work.

References

[1] RFC3174 - US Secure Hash Algorithm 1 (SHA1), http://www.faqs.org/rfcs/rfc3174.html, 2001.

[2] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt. A secure and improved self-embedding algorithm to combat digital document forgery. *Signal Process.*, 89(12):2324–2332, 2009.

[3] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt. A hash-based image encryption algorithm. *Optics Communications*, 283(6):879–893, 2010.

[4] F. Deguillaume, S. Voloshynovskiy, and T. Pun. Secure hybrid robust watermarking resistant against tampering and copy attack. In *Signal Processing*, pages 2133–2170, 2003.

[5] J. Dittmann, M. Steinebach, I. Rimac, S. Fischer, and R. Steinmetz. Combined video and audio watermarking: Embedding content information in multimedia data. In *Security and Watermarking of Multimedia Contents II*, pages 455–464, 2000.

[6] G. Doerr and J. Dugelay. A guide tour of video watermarking. *Signal Processing: Image Communication,*, 18(4):263–282, 2003.

[7] R. Dugad, K. Ratakonda, and N. Ahuja. Robust video shot change detection. In *in IEEE Workshop on Multimedia Signal Processing*, pages 376–381, 1998.

[8] R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial gray scale. In *Int. Symposium Digest of Technical Papers, Society for Information Displays*, page 36, 1975.

[9] A. J. Fridrich, B. D. Soukal, and A. J. Luk. Detection of copy-move forgery in digital images. In *in Proceedings of Digital Forensic Research Workshop*, 2003.

[10] A. Hampapur, T. Weymouth, and R. Jain. Digital video segmentation. In *MULTIMEDIA '94: Proceedings of the second ACM international conference on Multimedia*, pages 357–364, 1994.

[11] F. Hartung and M. Kutter. Multimedia watermarking techniques, 1999.

[12] R. Kasturi and R. Jain. *Computer Vision: Principles*. IEEE Computer Society Press, 1998.

[13] R. Lienhart. Comparison of automatic shot boundary detection algorithms. In *In proc. IST/SPIE Conf. on storage and Retrieval for image and video Databases VII*, pages 290–301, 1999.

[14] B. Mobasseri. Content authentication and tamper detection in digital video. In *ICIP00*, pages 458–461, 2000.

[15] P. Rees. Synchronisation and smpte timecode (time code), http://www/philrees.co.uk/articles/timecode.htm. 2001.

[16] Y. Wang, X. Liao, D. Xiao, and K.-W. Wong. One-way hash function construction based on 2d coupled map lattices. *Inf. Sci.*, 178(5):1391–1406, 2008.

[17] B. Yeo and B. Liu. Rapid scene analysis on compressed video. *CirSysVideo*, 5(6):533–544, 1995.

[18] H. Zhang, A. Kankanhalli, and S. W. Smoliar. Automatic partitioning of full-motion video. pages 321–339, 2001.