Multimed Tools Appl (2007) 34:107–135
DOI 10.1007/s11042-006-0074-7

A scalable signature scheme for video authentication

Pradeep K. Atrey · Wei-Qi Yan ·
Mohan S. Kankanhalli

Published online: 9 December 2006

Abstract This paper addresses the problem of ensuring the integrity of a digital video and presents a scalable signature scheme for video authentication based on cryptographic secret sharing. The proposed method detects spatial cropping and temporal jittering in a video, yet is robust against frame dropping in the streaming video scenario. In our scheme, the authentication signature is compact and independent of the size of the video. Given a video, we identify the key frames based on differential energy between the frames. Considering video frames as shares, we compute the corresponding secret at three hierarchical levels. The master secret is used as digital signature to authenticate the video. The proposed signature scheme is scalable to three hierarchical levels of signature computation based on the needs of different scenarios. We provide extensive experimental results to show the utility of our technique in three different scenarios—streaming video, video identification and face tampering.

Keywords Digital video · Video authentication · Authentication signature · Secret sharing

1 Introduction

Video authentication techniques have witnessed a tremendous rise in interest over the past few years. By definition, video authentication is a process that is used to

P. K. Atrey (✉) · W.-Q. Yan · M. S. Kankanhalli
Department of Computer Science, School of Computing, National University of Singapore,
Singapore 117543, Singapore
e-mail: pradeepk@comp.nus.edu.sg

W.-Q. Yan
e-mail: yanwq@comp.nus.edu.sg

M. S. Kankanhalli
e-mail: mohan@comp.nus.edu.sg

ascertain the trustworthiness of a digital video. In other words, a video authentication system ensures the integrity of digital video, and verifies that the video taken into use has not been tampered.

We motivate the need for authenticating a digital video by giving the following examples:

- A video clip can be doctored to defame a person. On the other hand, criminals get away from being punished because the video showing their crime can't be proved conclusively in the court of law.
- In surveillance systems, it is hard to reassure that the digital video produced as evidence is the one that was actually shot by the camera.
- A journalist cannot prove that the video played by a news channel is trustworthy.
- A video viewer who receives video through a communication channel cannot ensure that video being viewed is really the one that was transmitted.

So there is a compelling need for video, wherever it is and in whatever form it is, be made authenticable before use.

In this paper, we describe our proposed scheme for video authentication with detailed analysis and results. The earlier version of this work with preliminary results has been described in [1]. The proposed video authentication scheme is sensitive to spatial and temporal tampering; and also robust to frame dropping. Our method can be used in the following three different kinds of scenarios:

- In the scenarios where video is streamed through a communication channel, due to the large size of video data, the streaming often suffers from congestion problem at the bottlenecks on the network. To overcome the network congestion problem, some data loss (e.g. loss of few video frames) is common. For instance, the video transcoder or the designated router intentionally drops frames to save bandwidth or to avoid buffer overflow. The proposed method exploits the temporal relationship in video to afford frame drops yet maintains the integrity of the video.
- The proposed method is also useful in video identification. Video identification refers to a process that recognizes the existence of a particular video clip in large set of video data. For example, in an advertisement monitoring scenario where a commercial company or an individual can automatically identify in real time whether or not a TV channel is playing their video advertisement for the stipulated time. A TV channel may cut few frames to earn more time and the money. Our method can detect this type of tampering.
- Detection of object/region (such as human faces) tampering in a video is another application where our method can be used. The proposed method localizes the important spatial regions in a video and assigns higher weights to them in the authentication process.

The core idea of our technique is to utilize three hierarchical levels of a video and to use cryptographic secret sharing [15] to create what we call as a 'secret frame.' We authenticate a given video by computing the secret frames based on randomly generated private key at three hierarchical levels i.e. key frame level, shot level, and video level. We first segment the video into shots. Then, for each shot we identify the key frames. Here key frames refer to the frames that cannot be inferred semantically from the other key frames within the same video shot. At the key frame level, we compute the secret for each pair of key frames using secret sharing considering all non-key frames between the two key frames as shares. The secrets computed at this level and the key frames are treated as shares to compute the secret at the shot level. Finally, all shot secrets are used to compute a master secret that is considered as the media signature (which we will refer to as the 'signature' from now onwards) for the video. The proposed method is scalable to the three levels of signature computation and can be tuned to the need of the different scenarios. In our scheme, the size of the authenticating signature is equal to a video frame size and it is independent of the length of the video in time.

Our scheme does not depend upon any specific key frame selection algorithm. Any of the well known key frame selection method [5, 20] can be used. However we modify the existing key frame extraction method [20] slightly to incorporate the sensitivity against spatial cropping. Zhang et al. [20] used color histogram difference as a feature while we compute the weighted sum of the block-by-block difference (we call it 'differential energy') between the frames using pixels' luminance values. This offers security right to the pixel level.

We begin the paper with a brief discussion on the video authentication system and its properties, benign and tampering operations on the video in Section 2. In Section 2, we also provide a classification of authentication scenarios. Section 3 provides an overview of the related work. In Section 4, we present our scheme of video authentication with the analysis of its security against various attacks and robustness to frame dropping. We present the results in Section 5. Finally, Section 6 concludes the paper with a discussion on the future work.

2 Video authentication system

A typical video authentication system is shown in Fig. 1. In the authentication process (Fig. 1a), for a given video, the authentication algorithm processes the features extracted from the video and outputs the authentication data which is encrypted using the encryption key to form the signature. The video integrity is verified by computing the new authentication data using the same authentication algorithm and features. The new authentication data is compared with the original authentication data as shown in Fig. 1b. If both match, the video is treated as authentic otherwise it is construed to be tampered.

An ideal video authentication system, to be effective, must follow the properties such as sensitivity to alterations, localization and self-recovery of altered regions, robustness to benign operations, tolerance to some loss of information due to benign operations, compactness of authentication data, one-way property of authentication data, sensitivity against false alarm and computational feasibility.

2.1 Classification of benign and tampering operations

Several video processing operations that do not modify its content semantically such as geometric transformations, image enhancement, and compression are classified as benign operations. In addition to having robustness against benign operations, an ideal video authentication system must make a given video resistant against all

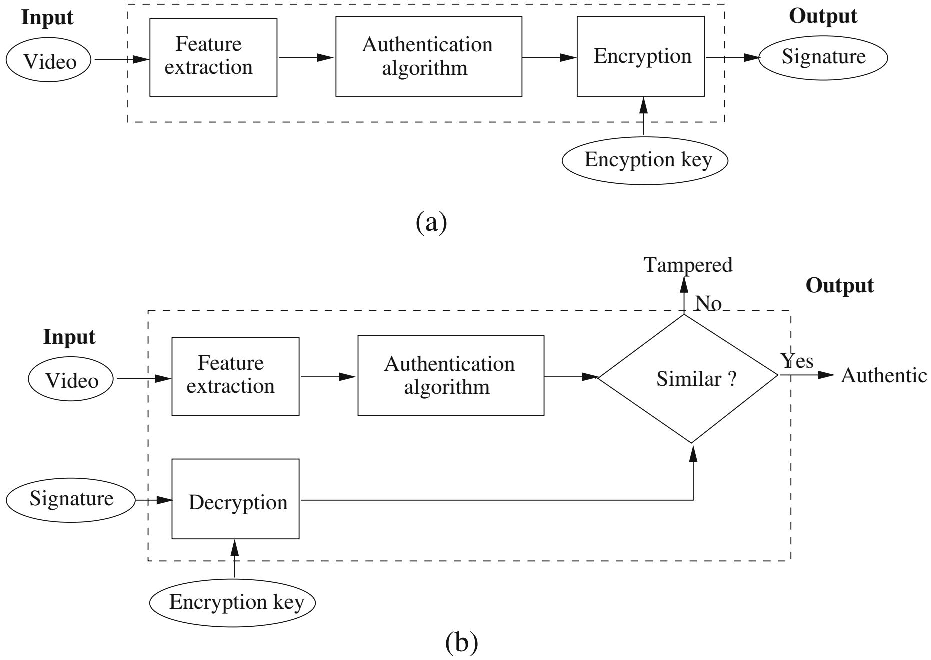


Fig. 1 A typical video authentication system **a** Authentication process **b** Verification process

possible attacks and must verify whether a given video is tampered or not. It is really useful to find where (i.e. localization of alterations) and how the tampering has been done.

Based on the dimension, the tampering is divided into spatial tampering and temporal tampering. Spatial tampering, also called as intra-frame tampering, refers to the alterations in frame content. The malicious operations include cropping, replacement, content adding and removing at pixel level, block level, frame-level, sequence of frames level, and shot level. The type of manipulation includes spatial cropping in a specified region in a frame, in several frames, or even in entire shot/video. Temporal tampering is an inter-frame jittering. It refers to manipulations performed with respect to time. The possible alterations are to drop (or remove), to replace, to add extra frames, and to reorder the video frames (or the sequence of frames). These manipulations can be performed on sequence of frames level, shot level, or video level. Due to temporal redundancy in the video data, it is noticed that dropping few adjacent frames does not affect the visual appearance and semantic meaning much. So, we can afford to drop or reorder the frames up to certain extent if the application so demands.

2.2 Classification of authentication scenarios

We present a classification of video authentication scenarios based on whether authentication and verification processes are performed online or offline. We further categorize the scenarios based on streaming or non-streaming of video, and also based on live or recorded video. The main classification is as follows:

- Online-authentication: online-verification.
- Online-authentication: offline-verification.
- Offline-authentication: online-verification.
- Offline-authentication: offline-verification.

This classification is summarized in Table 1 with various example scenarios and the robustness issues in those scenarios. The *online* refers to the realtimeness in computation, while *offline* implies the vice-versa. As shown in Table 1, each of the main groups (e.g online-authentication: online verification) are further classified into streaming and non-streaming, and also in live and recorded categories. Streaming implies that the video is transported over the network, whereas in the non-streaming class no network issues are involved. Similarly, in the live video, no editing is allowed since the video should be received as it is shot. On the other hand, in recorded video, it is first recorded, edited, and then made ready for use. The robustness issues in the design of a video authentication system are provided in Table 2. These robustness issues given in Table 2 are referred in Table 1.

Table 1 Classification of authentication scenarios

Authentication-verification	Streaming/ Non-streaming and live/ recorded video	Example scenarios	Robustness issues (referred from Table 2)
Online-online	Streaming and live	– Live monitoring by surveillance camera – Live broadcast of an event – Online video-conference	1–4
Online–offline	Streaming and live	– Offline monitoring by surveillance camera	1–4
	Non-streaming and recorded	– To verify the creditability of a recorded event (e.g. journalist's report) – Face tampering disputes	2, 3, 5
Offline–online	Streaming and recorded	– Advertisement monitoring – Video-on-demand – Broadcast of a recorded TV-program	1–4
Offline-offline	Streaming and recorded	– Online shopping of a video	1–4
	Non-streaming and recorded	– Manual or offline shopping of a video CD	2, 3, 6

Table 2 Robustness issues referred in Table 1

1. Occasional frame dropping to save the bandwidth and to avoid network congestion.
2. Geometric transformations and image processing operations.
3. Compression.
4. Network issues and transmission error.
5. Editing is allowed as long as the relative ordering of video sequences are preserved.
6. Data storage error.

3 Related work

Video authentication problem has been studied by many researchers. Mainly two approaches have been used: digital signature [10, 11, 13, 17, 18] and digital watermarking [2, 3, 6, 8, 9, 19]. The cryptographic hash function has been widely used to ensure security by hashing the inherent features of video to produce a compact size signature or watermark. Cryptographic hash function is good for security but it does not offer robustness to benign operations. Therefore, the need of visual hash function [14] has been discussed.

In the past, researchers have proposed several schemes to authenticate video in specific scenarios that meet certain robustness requirements but their schemes have one or the other weaknesses. Lin and Chang [10] and Yin and Hu [19] have proposed compressed domain schemes that are robust against transcoding [10, 19] and editing [10] operations. To compute the signature, Lin and Chang [10] used the difference in DCT coefficients of frame-pairs which is vulnerable to counterfeiting attack since the value of DCT coefficients can be modified keeping their relationship preserved. Yin an Hu [19] used DC-DCT coefficients as features to build a watermark. Since the DC-DCT coefficients are computed using a linear equation based on the statistics of pixel value, so can be compromised. An attacker can replace a block of 8×8 pixels by another 64 pixel values retaining the same DC-DCT coefficient. If this operation is repeated for the entire frame, then it may lead to a major cropping still undetected by the authentication system. The features such as edges in the frame used by Dittman et al. [4] and by Tzeng and Tsai [17] are also highly vulnerable to content modification if a smart attacker modifies the content keeping the edges preserved. In other work, He et al. [8] proposed an object-based scheme that uses the background features to embed the watermark into foreground objects to establish a relation between background and the foreground of a video.

In our method, we make use of cryptographic secret sharing to ensure security right to the pixel level by computing its secret through the temporal axis [15]. Cryptographic secret sharing has been successfully used for non-traditional applications such as message authentication [7]. The novel feature of this method is that it ensures unconditional security [16]. In other words, the security of secret sharing does not rely on any unproven assumptions (unlike that of many other cryptographic schemes). This motivated us to use the secret sharing in place of traditional hash function but with a fundamental difference. In normal secret sharing, we hold the original secret and we utilize the interpolating polynomial to compute the shares (corresponding to frames here). In our scheme, we have the shares as given and we use them to compute the polynomial. Each of the coefficients of this polynomial is hashed to a new coefficient. We then construct another polynomial using the new set of new coefficients and then deduce the secret by extrapolating the new polynomial at a position known to the authenticator and the verifier. We also exploit the temporal redundancy in video to reduce the shares required in secret computation. It reduces the computational efforts to a high degree. In secret sharing framework, by adding

Table 3 The symbols used in this paper

Symbol	Description
a_u, a'_u, b_u, b'_u	Coefficients of the polynomials
b	Total number of blocks in the video frame
$D_{l,m}$	Differential energy between the frame l and frame m
$d^k_{l,m}$	Euclidean difference between the kth block of the frame l and frame m
D_T	Threshold for differential energy between the two frames
D	Differential energy factor (The configurable input parameter)
$F(x)$	Function f of x (in Eq. 3)
g, h	Indices used in the K-frame identification algorithm
$G(x)$	Function g of x
i	Index for the key frame, $1 \leq i \leq q$
j	Index for the shot, $1 \leq j \leq p$
k	Index for the block, $1 \leq k \leq b$
K_i	ith key frame (Also denoted by K-frame)
M	Maximum size of the buffer (in number of frames) for processing the video
N	Total number of frames in the video
p	Number of shots in the video
P	Prime number
P'_K	Private key used for hashing the polynomial coefficients
P_K	Private key used for extrapolating the secret
q	Number of key frames in a video shot
Q	Total number of key frames in the video
r	Number of non-key frames used to determine key-secret
S	Signature for the video
$Sign_{new}$ & $Sign_{old}$	New and original authentication data (signatures)
sim	Similarity value between two signatures
S^K_i	ith key-secret (i.e. Secret at key-frame level, also denoted by S^K-frame)
T_p	Quantization factor
u, v	Indices used in the Lagrange's interpolation (in Eq. 3)
V	Input video
x_u	x-position for uth frame (to be interpolated in Eq. 3)
Y_u	Pixel's luminance value for uth frame (to be interpolated in Eq. 3)
w_k	Weight for the kth block
W	Weight factor (The configurable input parameter)
W^K_i	ith Weight frame at key-frame level
W^S_j	jth Weight frame at shot level
W^M	Weight frame at video level

some redundant frames (like error correcting codes) lying on the interpolating polynomial we can afford to lose some frames yet compute the same secret [7].

4 Proposed method

Our algorithm is designed to work in uncompressed domain so that it remains independent of the video format and it fits into two categories ‘offline-authentication: online-verification’ and ‘offline-authentication: offline-verification’ (as per the classification described in Section 2.2). Various symbols used in this paper are summarized in Table 3.

4.1 Overview of the method

The proposed method uses the three hierarchical levels (*key-frame*, *shot* and *video*) of a video and computes the signatures at each of the three levels as shown in Fig. 2. The signatures at three levels are combined hierarchically to form the master signature for the input video. The complete process of video authentication is described as follows:

1. The input video is segmented into p shots. We define a *video shot* as a contiguous recording of one or more video frames describing a contiguous action in time and space.
2. For each of the video shots, we find the key frames (denoted by K-frame), K_i, $1 \leq i \leq q$.
3. After identifying the key frames, we quantize the pixels’ luminance values for all the frames (key frames and non-key frames) using a quantization factor T_p. This brings all the pixels, those vary in their luminance value by T_p, to the same quantization level; that eventually allows us to better utilize the temporal redundancy among the video frames.

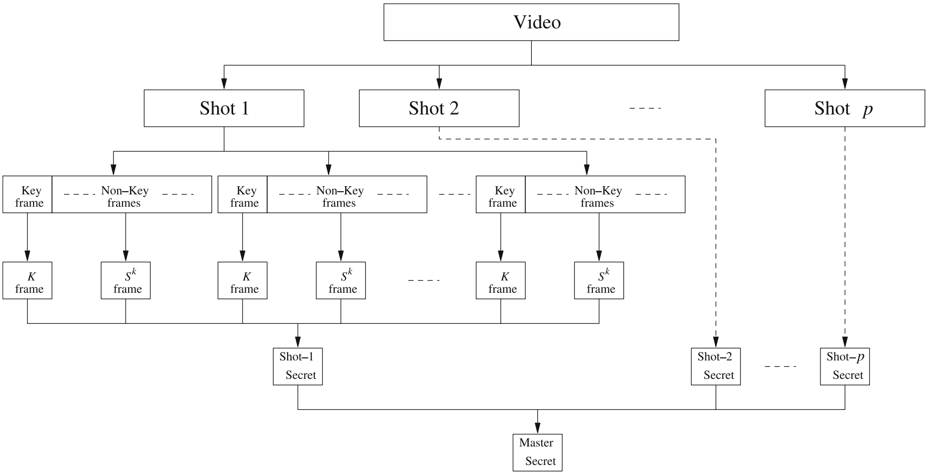


Fig. 2 Hierarchical structure of the proposed scheme

4. Next, we compute signatures at three hierarchical levels: key-frame level, shot level and video level. We call the signature at key-frame level to be *key-secret* (denoted by S^k-frame) which is computed using the non-key frames between each pair of the key-frames.
5. Once key-secrets for all the pairs of key-frames are available from the previous step, signature at shot level (we call it to be *shot-secret*) is computed using the frame sequence K_1, S^k_1, K_2, S^k_2, ..., K_{q-1}, S^k_{q-1}, K_q. This is performed for all the p shots.
6. Finally, the proposed method computes the signature at video level (we call it to be *master secret*) from all shot secrets. The master-secret is encrypted using a private key to form the signature for the video.

4.2 Authentication steps

Key frame extraction Since video shot boundary detection is very well understood, we assume that the shot boundaries of the video have been computed [5, 12]. After a video shot is acquired, we identify the key frames for each shot using the method similar to [20]. However, we use 'differential energy' as a feature which compared to color histogram difference provides better security for the important regions in video frames by assigning higher weight to them.

The first frame in a shot is designated as a key frame. Next, we compute the differential energy between the key frame and the subsequent frames. We define differential energy as the weighted sum of block-wise Euclidean differences between the two frames. Once the differential energy level is found greater than a threshold value, we designate the corresponding frame as the key frame. This process continues till it reaches to the last frame in the video. The last frame is by default designated as key frame.

- *Differential energy computation*: We use pixel luminance values to compute the differential energy. More formally, $D_{l,m}$ is the differential energy between frame l and frame m. The $D_{l,m}$ is computed by the following equation:

$$D_{l,m} = \sum_{k=1}^{b} w_k \times d^k_{l,m} \tag{1}$$

  where, k is the block index and b is the number of blocks in a frame. Each block corresponds to a group of 8 × 8 pixels. The $d^k_{l,m}$ is the Euclidean difference between the frame l and frame m for kth block, and w_k is the weight for kth block that is multiplied to the Euclidean difference between the frame l and frame m to exaggerate it so that we can increase the level of sensitivity for few specific blocks in the video against spatial cropping by increasing weight for the respective blocks.

  The value of weight w_k is determined based on how significant that particular block is. For example, if the block corresponds to an object which is more likely to be tampered, we assigned a weight $W > 1$ to it; otherwise we assign a weight 1 to it. Note that W is a configurable input parameter which we call 'Weight factor.'
- *K-frame identification*: The key frames (K-frames) are identified based on the differential energy computed from previous step. We assume that the first and the last frame of video shot are chosen as key frames by default. The intermediate

key frames are determined based on whether the differential energy between a frame and its previous key frames is greater than or equal to the threshold D_T. The D_T is given by:

$$D_T = width \times height \times D \tag{2}$$

where *width* and *height* are the frame-width and frame-height, respectively, of the video. The D is a configurable input parameter which we call 'Differential energy factor.' The proposed algorithm provides a tradeoff between security and robustness by tuning the configurable input parameters D and W (described further in Section 4.5).

The following are the algorithmic steps to identify K-frames:

1. All N frames in shot are the input.
2. Designate the first frame as key frame, *K-frame*[0]=0.
3. Set $h = 1$, $g = 0$.
4. If $h \geq N$ then go to step 8.
5. Compute differential energy $D_{g,h}$.
6. If $D_{g,h} \geq D_T$ then $g = g + 1$, *K-frame*$[g] = h$.
7. $h = h + 1$, go to step 4.
8. $g = g + 1$, *K-frame*$[g] = N - 1$; return *K-frame* array.

Authentication at key-frame level Once the key frames are detected in a video shot, the key-secrets (i.e. S^k-frames) are computed as follows. First, we perform the quantization and the temporal redundancy utilization steps. Then, the S^k-frame is computed by the (r, r)-secret sharing assuming r non-key frames as shares (after the quantization and the temporal redundancy utilization steps). The r is determined for each pixel by knowing the distinct luminance values through the temporal axis of the video. More formally, ith S^k-frame contains of extrapolated value computed by the interpolation of r non-key frames between ith K-frame and $(i + 1)$th K-frame through the temporal axis as shown in Fig. 3. The S^k-frame is of same size as the non-

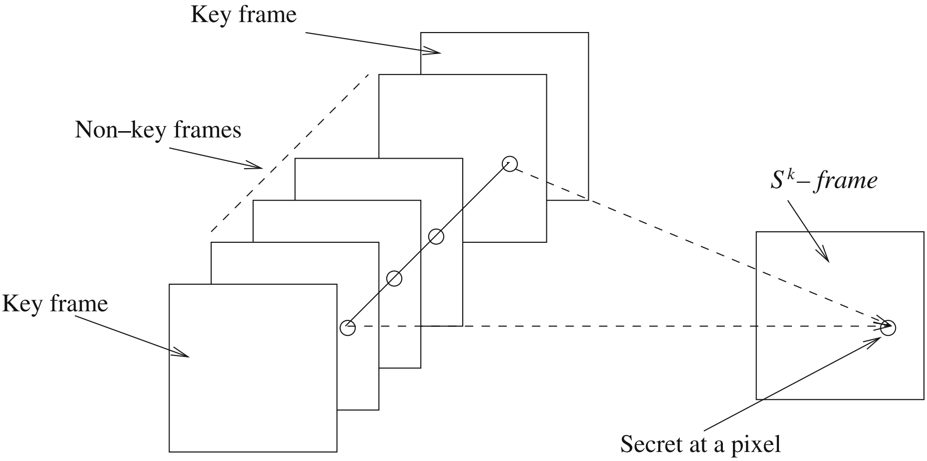


Fig. 3 S^k-frame computation

key frame or any other frame in the video. Finally, the weight frames at key-frame level (denoted by W^K-frames) are computed which contain the weights w_k for each block k. These steps are described below.

- *Quantization*: We uniformly quantize the 256 luminance levels for the pixels in all the frames to fewer levels. A quantization factor T_p is used to reduce the number of luminance levels which eventually lowers the number of shares in the secret computation. For example, $T_p = 10$ reduces 256 levels to $\lceil(256/10)\rceil = 26$ levels.
- *Temporal redundancy utilization*: After quantization step, we get many pixel values quantized to the same level. We exploit the data redundancy through temporal axis by choosing only the unique luminance values while ignoring the repeated ones for the non-key frames. The chosen non-key frames are used for the computation of 'key-secret.'
- *'Key-secret' computation*: The key-secret (S^k-frame) is computed by extrapolation as shown in Fig. 4. The x-axis indicates the position of non-key frames obtained from the previous step while the y-axis indicates the luminance value of pixels in those frames. We compute the interpolating polynomial using (r, r)-secret sharing (Eq. 3):

$$F(x) = \left(\sum_{v=1}^{r} \prod_{u=1, u \neq v}^{r} \frac{x - x_u}{x_v - x_u} Y_u \right) mod\ P \qquad (3)$$

  Eq. 3 is essentially the Lagrange interpolation formulation where x_u position refers to the uth non-key frame, Y_u is the pixel value of the uth frame, and P is a prime number. This formulation is exactly like the problem of secret sharing but we use it in different way. In Eq. 3, we have the shares (non-key frames) as given and we use them to compute the polynomial. The coefficients (say a_u, for

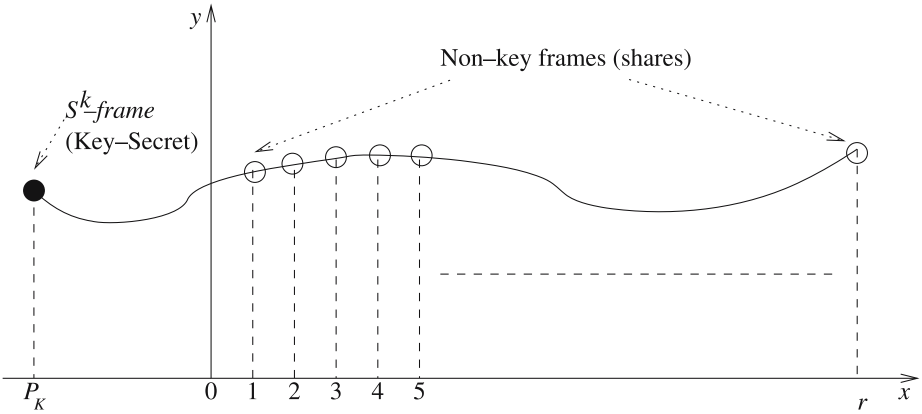


Fig. 4 S^k-frame extrapolation

$0 \leq u \leq r$) of polynomial $F(x) = \sum_{u=0}^{r} a_u x^u$ are hashed to a new set (say b_u, for $0 \leq u \leq r$) of coefficients as given in Eq. 4:

$$b_u = Hash_{P'_K}(a_u) \tag{4}$$

where, *Hash* is a hash function (such as SHA-1) and P'_K is the hash key. Using the set ($b_u, 0 \leq u \leq r$) of coefficients, we construct another polynomial (say $G(x) = \sum_{u=0}^{r} b_u x_u$) and then deduce the secret by extrapolating the $G(x)$ under modulo P at x-position which is determined by the private key P_K. We analytically discuss the security of our scheme in Section 4.5.

The steps to compute S_i^k-frame are summarized here:

1. The r non-key frames between K_ith and K_{i+1}th key frames are the input.
2. Position all the input non-key frames along x-positions to compute the interpolating polynomial $F(x)$.
3. Construct a new polynomial $G(x)$ by replacing the coefficients of $F(x)$ with their hashed values.
4. Obtain the secret frame S_i^k by extrapolating the polynomial $G(x)$ at position $x = P_K$.

- *Weight-frame computation*: For each K-frame K_i, $1 \leq i \leq q$, we compute a weight frame (W_i^K-frame) which contains the values of weights w_k for each block $1 \leq k \leq b$. We assume that the value of weight remains uniform within a 8×8 pixels block. The W^K-frame is computed based on significance level of blocks in video frames between two key frames. For example, in authenticating a video against face tampering, the spatial region in a video frame where face exists is considered more significant compared to other spatial regions. A face detector can be used prior to the authentication to build a weight-frame automatically. Therefore, we assign the weight $w_k = W$ (Weight factor) to the more significant blocks and the weight $w_k = 1$ to normal blocks.

Authentication at shot level

- *Shot-secret computation*: The shot-secret frame encapsulates the sequence of K-frames and the S^k-frames computed in the previous sections. Considering the quantized K-frames ($K_i, 1 \leq i \leq q$) and the S^k-frames ($S_i^k, 1 \leq i \leq q-1$) as shares, the 'shot-secret' is computed using $(2q-1, 2q-1)$-secret sharing. The q is number of key frames in a shot. We use the interpolating sequence $K_1, S_1^k, K_2, S_2^k, ..., K_{q-1}, S_{q-1}^k, K_q$ as shown in Fig. 5 and obtain the 'shot-secret' by following the steps similar to what used in computing S_i^k-frame (in previous section). In Fig. 5, the x coordinate indicates the locations of the K-frames and S^k-frames. The ith K-frame and ith S^k-frame are positioned at $2(i-1)+1$ and $2(i-1)+2$ x-positions, respectively. Since last K-frame does not have corresponding S^k-frame so the last K-frame is positioned at x-coordinate $2(q-1)+1$.
- *Updating weight-frame*: The weight frames W_i^K, $1 \leq i \leq q$, computed at key-frame level are used to workout W_j^S, $1 \leq j \leq p$, where W_j^S is the weight frame for jth shot and p is number of shots in the video. The W_j^S is computed as follows:

$$W_j^S = \bigcup_{i=1}^{q} W_i^k \tag{5}$$

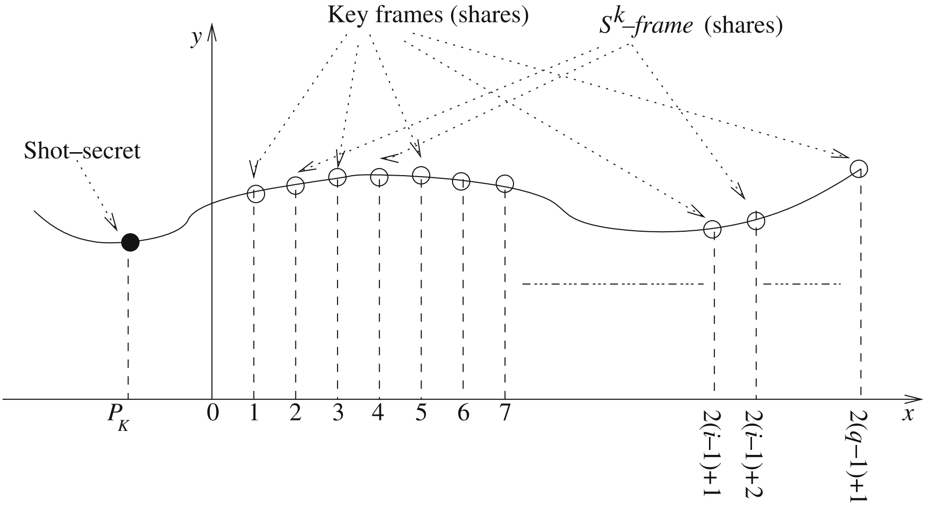


Fig. 5 Secret frame extrapolation at shot level

where, $\bigcup$ is an union operation which computes the weight of kth block of W_j^S by choosing the maximum of weights of all the kth blocks of W_i^K, $1 \le i \le q$.

Authentication at video level At the top level of the hierarchical structure of our scheme, we follow these two steps:

- *Master-secret computation*: At the video level, we utilize the shot-secrets obtained for all the p shots of the video to compute the 'master-secret' using the (p, p)-secret sharing in the similar way as described in previous two subsections. This 'master-secret' is treated as the signature for the video.
- *Updating weight-frame*: The weight frame W^M at master level is computed as:

$$W^M = \bigcup_{j=1}^{p} W_j^S \tag{6}$$

where, $\bigcup$ is again an union operation which computes the weight of kth block of W^M by choosing the maximum of weights of all the kth blocks of W_j^S, $1 \le j \le p$.

The 'master-secret' frame, 'weight-frame' (WM) and the configured inputs (Differential energy factor D and Weight factor W) are encrypted and provided with the video to the verifier.

4.3 Signature verification

We verify the authenticity of a given video as follows. We follow the steps described in Section 4.2 to compute the new signature frame (say $Sign_{new}$) using the same configurable inputs (D, W) and private key (P_K) with which original signature frame (say $Sign_{org}$) is computed. This $Sign_{new}$ frame is compared to the $Sign_{org}$ frame. If they match, then we are guaranteed that the video has not been tampered with, and its content is the same as that of the original, or else this video is not to be trusted. We measure the similarity between two master frames (or signature) block by block. We call this similarity measure as *sim* value.

The following steps are performed to compute the *sim* value:

```
1. The signature frames: $Sign_{new}$ and $Sign_{org}$, each consisting of $b$ blocks.
2. For each block $k$, $(1 \leq k \leq b)$, in $Sign_{new}$ and $Sign_{old}$
        If $abs(Sign_{new}(k) - Sign_{org}(k)) = 0$
            then $Count+ = w_k$.
3. Percentage of similarity between $Sign_{new}$ and $Sign_{org}$ i.e.
   $sim = (Count/\sum_{k=1}^{b}(k \times w_k)) \times 100\%$.
```

The term w_k is the weight of kth block and *Count* indicates the number of blocks which are matched in original and the new signature frames.

The *sim* value lies in the range [0,100] and if $sim = 100$, the two master frames are the same, however if $sim = 0$, the two master frames are quite different. The *sim* value is used for judging the authenticity of any video. If the *sim* value is high, then the video has undergone benign transformations. But if the *sim* value is low, then the video must have undergone some significant tampering.

To judge whether or not a given video is tampered, the threshold on the *sim* value is learned empirically for each type of tampering operation. For the specified values of input parameters (D and W), we obtain the *sim* data values for a specific type of tampering and model it as a Gaussian distribution. The mean μ and variance σ of this distribution are estimated. The threshold is determined in the range of $\mu - 2\sigma$ to $\mu + 2\sigma$.

4.4 Scalability of the our scheme

As described in the previous section, the proposed method allows authentication and verification at three hierarchical levels. This offers the method to be used in a scalable manner in terms of number of hierarchical levels of signature computation. For example, in an ‘offline-authentication: online-verification scenario’ (such as advertisement monitoring scenario), one may like to verify the authenticity of video in real-time. This could be feasible if authentication and verification are done at the key-frame level. In this case, the piece of video (i.e. video frames between two key frames) along with their ‘key-secret’ must be available to the verifier. This suggests that the proposed scheme can be scaled depending upon the application requirements. In other scenario of streaming video, where robustness against frame dropping is an important issue, we can afford to loose non-key frames in case of network congestion. In such scenario, the computation of the secret at shot level could be more useful.

In an ‘offline-authentication:offline-verification’ scenario where authentication as well as verification does not require computing in real time, the ‘master-secret’ is used to verify the authenticity of a video.

4.5 Analysis

Security analysis Our algorithm ensures the authenticity of video by computing secret (or signature) from the video data based on hash key P'_K and the private key

P_K which are known to the authenticator (the owner) and to the verifier (the user) of the video. By having different sets of keys for different users, the owner of the video can individualize the signature to each user.

We analyze the security of our scheme as follow. In our scheme, the video V, its secret S and the keys P'_K and P_K can be made public. As described in Section 4.2, our scheme essentially does the following:

1. Computes the polynomial $F(x) = \sum_{u=0}^{r} a_u x^u$.
2. Finds the coefficients $b_u = Hash_{P'_K}(a_u)$, for $\forall u$.
3. Compute the new polynomial $G(x) = \sum_{u=0}^{r} b_u x^u$.
4. Extrapolate the $G(x)$ at $x = P_K$, i.e. $S = G(P_K)$.

Note that, it is easy for an attacker to find (in step 3) another polynomial $G'(x) = \sum_{u=0}^{r} b'_u x^u$ which (in step 4) extrapolates to the same secret S using the same key P_K. The b'_u, for $0 \leq u \leq r$ are the coefficients of $G'(x)$. However, due to the one-way property of a hash function, it is computationally hard to find the coefficients a'_u, $0 \leq u \leq r$, such that $\forall u$, $Hash_{P'_K}(a'_u) = b'_u$. Therefore, it is computationally hard to find another polynomial $F(x) = \sum_{u=0}^{r} a_u x^u$ and the corresponding video for which an attacker can obtain the same secret S using the same keys. Hence, our scheme is computationally secure.

In the subsequent paragraphs, we further analyze how secure is our algorithm against various types of tampering:

- *Sensitivity against spatial tampering*: The hierarchical structure of our scheme ensures security right to the block level spatially and to the level of non-key frames temporally. As we compute 'key-secret' using the non-key frames, any alteration in the content of non-key frames is reflected in the signature.
  In our method, since the key frame selection is based on how much a frame differs from its previous key frame, a change in the content of a frame can lead to its emergence as a new key frame or the elimination of an existing key frame. Any change in key frame position affects interpolating polynomial; hence, a different secret frame is generated. This change is propagated from one level to another in our hierarchical framework, which eventually changes the secrets at other levels also.
  The proposed method computes the differential energy at block level. If the differential energy between the blocks (having more significance or where the 'object of interest' exists) of two frames is noticeable, the proposed method updates the 'Weight-frame' accordingly by assigning a higher weight to the region of interest. Computing differential energy at block level ensures that the even small changes are highlighted. This is very useful in the cases such as tampering with the face of a person. Since a face may occupy several blocks in the frame, it is infeasible to replace it with a face that occupies the same number of blocks having the similar content. We provide experimental evidence for this in the next section.
- *Sensitivity against temporal tampering*: The proposed algorithm computes the secrets at each level by extrapolating the polynomial which is generated by interpolating the given shares (or frames) in a specific order. If any of the frame in the sequence is altered (content-wise or position-wise), it is reflected in the secret frame because the interpolation sequence changes. For instance, as we consider all non-key frames as shares to compute 'key-secret' S^k-frame, any alteration such

as dropping and reordering (or re-indexing) of non-key frames is reflected in S^k-frame. That eventually changes the 'shot-secret' frame and the 'master-secret' frame. Also, any alteration in the shot sequence of a video can be noticed in its 'master-secret.'

Also, if a frame is replaced by some extraneous frame or some extra frames are added, the first of these extra frames emerges as an extra key frame because the differential energy (between the first new frame and the old frame just before it) rises above the threshold level D_T . Having few more key frames in shot secret computation, we get an entirely different 'shot-secret,' hence the different 'master-secret.' So our algorithm is very secure against all the temporal attacks.

Robustness analysis The proposed method achieves the robustness against frame dropping by exploiting the temporal redundancy in the video data. In the video streaming scenarios where frame dropping is inevitable, the proposed method is scaled down to shot-level secret computation so that the secret computation does not depend on non-key frames. It allows us to drop non-key frames in case of network bottleneck.

Our method is also robust against frame reordering if it is performed with a group of non-key frames. The robustness can be maintained till the key frames sequences are preserved. If the frames that lie between two different groups of non-key frames are reordered, the key frame position changes and hence the signature also changes drastically.

Table 4 shows how the configurable inputs to our algorithm can be varied appropriately to tailor for the requirement of security and robustness, where '↑' and '↓' indicates the increase and decrease, respectively. We further show in results section, the effect of D on robustness against frame dropping (in Section 5.2) and the effect of W on security (in Section 5.4).

More specifically, higher the value of D higher the differential energy D_T would be. More the value of D_T lesser the number of key frames would be. In other words, the number of non-key frames would be more as D_T increases. Hence we can afford to loose more number of (non-key) frames. However, loosing too many frames affects the security. Therefore, we can have a trade-off between security and robustness by setting appropriate value of D. As shown in Fig. 6a, number of key frames (out of 121 frames shot of a video 'road.mpg') decreases as D increases.

By increasing the value of W, we increase the level of security at block level. This is used to detect tampering in a particular spatial region of a video. We show in Fig. 6b how the increase in W affects the *sim* (i.e. similarity between two signatures) value. The five plots are shown with varying amount (10 to 50%) of change (tampering) in spatial region of a video. The graph shows that the *sim* value is inversely proportional to amount of region tempering in a video. The *sim* value is also inversely proportional

Table 4 Effect of inputs on security and robustness

Input parameter	Approximate range	Effect of security and robustness
Differential energy factor D	1-10	$D \uparrow \Rightarrow$ Security↓, Robustness ↑
Weight factor W	1-10	$W \uparrow \Rightarrow$ Block level security↑

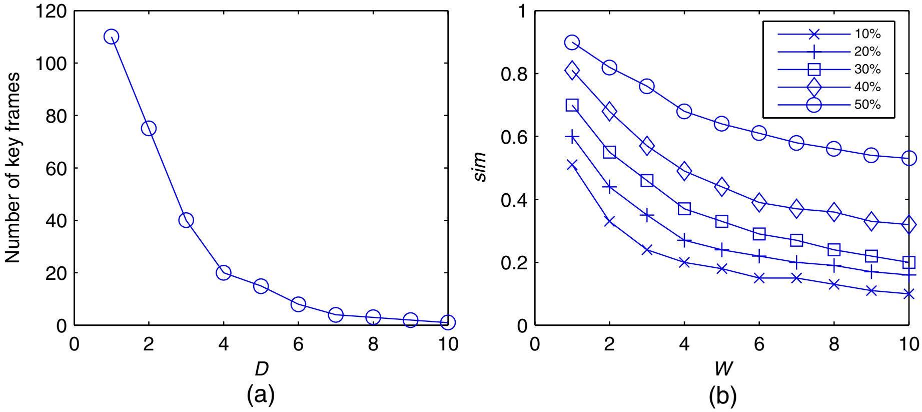


Fig. 6 Effect of **a** D and **b** W on security and robustness

to the W. By having $W = 1$, we assign equal importance to all blocks. For $W = 1$, *sim* value is exactly equal to spatial similarity between the two video. For instance, if only 10% spatial tampering is performed, the *sim* value will be 90%. However, this could be misleading when an important region such as a face which usually covers a small spatial area is tampered. Therefore, it is reasonable to assign a higher weight to such regions so that any tampering in these regions is reflected significantly in sim value during verification. For instance, as shown in Fig. 6b, at $W = 10$ *sim* value is noticed as 0.47 and 0.29 with the region tampering of only 10% and 20%, respectively.

Complexity analysis Consider a video of resolution $width \times height$ and of length N frames, the worst time complexity of our method would be $O(width \times height \times N^2)$ assuming zero temporal redundancy in the video. Since, in practice, a video has lot of temporal redundancy, therefore the number of frames used for interpolation would be small compared to N. Note that we use the parameter T_p to utilize the temporal redundancy due to which number of frames to be interpolated decreases significantly.

In real-time streaming video scenarios, since we use only the key frames of a video; the time complexity would be $O(width \times height \times Q)$, where Q is the total number of key frames in the video. Since the total number Q of key-frames are usually significantly less than the total number N of frames in a video, our scheme meets the real-time computing requirements.

Assuming that a block is a very small spatial region to perceive visually, the time-complexity can further be improved if the secrets are computed at block level instead of pixel level.

We analyze the space complexity of our scheme as follows. A buffer of size $M \times width \times height$ in pixels is used to read the video in real-time, where M is the total number of frames in buffer. Another buffer is used to store all the key frames and key-secrets. Assuming that there are Q key-frames in whole video, the space complexity is given by $O(max(M, Q) \times width \times height)$.

5 Results

To substantiate the theory, we have performed test for several video sequences. We show the utility of our method in the following three different applications:

1. Streaming video scenario where robustness against frame dropping is required.
2. Video identification (e.g. advertisement monitoring) scenario where dropping of frames is treated as tampering.
3. Face tampering scenario.

For the first scenario, we compute signature at shot level. By doing this, we afford to loose non-key frames to achieve robustness against frame dropping. We provide the corresponding results in Section 5.2.

For second scenario, we ensure better security by computing the 'key-secrets' using non-key frames. In this case, we compute secrets at all three hierarchical levels—key-frame level, shot level and the master level. We provide the results of temporal and spatial tampering in Sections 5.3 and 5.4, respectively.

The results for face tampering are provided in Section 5.4.

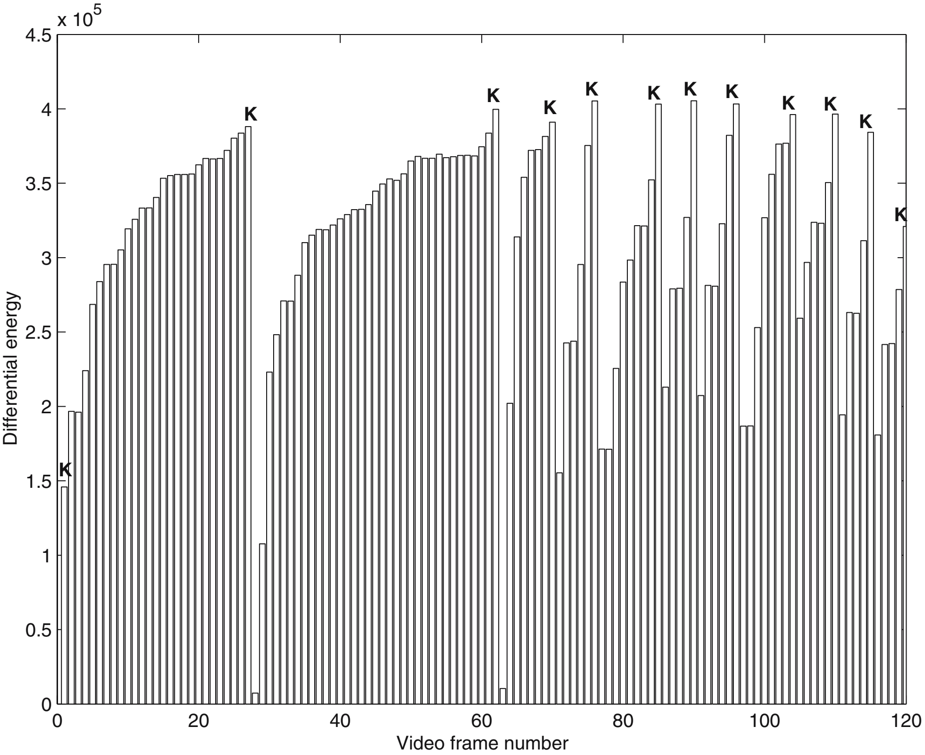


Fig. 7 Differential energy between the frames of 'road.mpg' video

5.1 Preprocessing

For these test, three MPEG videos—‘road.mpg’, ‘bus.mpg’ and ‘face.mpg’, each of resolution 320 × 240, are used. The input videos are decoded using a MPEG-decoder. We used SHA-1 as a hash function. The configurable input parameters are set as follows: $T_p = 10$, $D = 5$, and $W = 10$.

From the video ‘road.mpg’, we extracted a shot of 121 frames. Next, we extract the key frames from this shot using the method described in Section 4.2. Figure 7 shows the differential energy between the frames. The ‘**K**’ indicates the key frames. The extracted key frames are shown in Fig. 8.

5.2 Robustness against frame dropping

As mentioned earlier, we do not consider non-key frames in the secret computation. Only key frames are used to compute the secret for the shot. That allows us to drop the non-key frames in a scenario where frame-dropping is common.

For frame dropping, we perform an exhaustive test by dropping non-key frames and key frames. Table 5 shows how the *sim* value (i.e. similarity value between the original and new signatures) and the key frame sequence change with respect to dropping of frames. The results show that dropping of non-key frames (such as frames 1 to 25, or frames 28 to 61) does not affect the *sim* value, however if a key frame (27) is dropped, the *sim* value sharply goes down to 38%, and also the frame

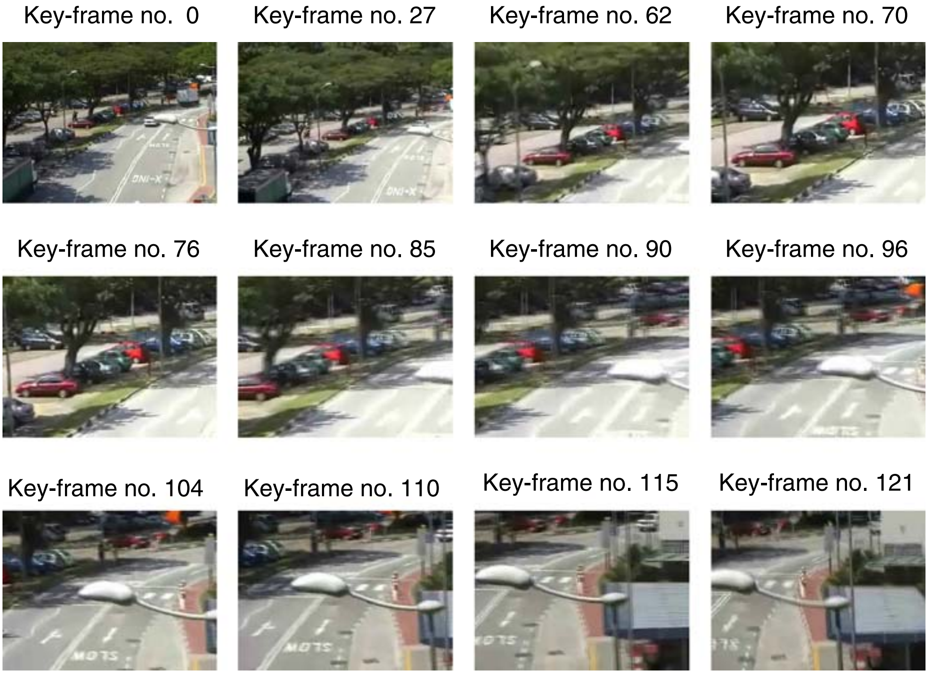


Fig. 8 The key frames of ‘road.mpg’ video

Table 5 Robustness against frame dropping in 'network congestion' case

Frames dropped	Key frame sequences	*sim* (%)
1 to 25	0,27,62,70,76,85,90,96,104,110,115,121	100
28 to 61	0,27,62,70,76,85,90,96,104,110,115,121	100
27	0,28,62,70,76,85,90,96,104,110,115,121	38
27, 62	0,28,63,70,76,85,90,96,104,110,115,121	33
27, 62, 70, 76	0,28,63,71,77,85,90,96,104,110,115,121	18
27, 62, 70, 76, 85, 90	0,28,63,71,77,86,91,97,105,112,119,121	7

number 28 emerges as a new key frame. The decrease in the *sim* value continues as more key frames are dropped. This shows that our scheme is robust to frame dropping if only non-key frames are dropped.

5.3 Temporal tampering

In the temporal domain, we performed test for dropping, replacing, adding and reordering of video frames/shots. The tests are performed by tampering with one to ten frames in the video shot 'road.mpg'. Due to the randomness in selection of frames for tampering, we considered the average of ten instances of each test. We discuss the result of each test as follows.

Frame dropping In the scenario such as advertisement monitoring where a single frame dropping is considered as tampering, our method can be tuned to achieve that level of security by computing secrets at all three hierarchical levels. We tested the security of our method by dropping few frames as shown in Table 6. The results show that the *sim* value sharply goes down to 50% even if a single frame is dropped, and this fall continues as we drop more frames. This is also noticed from the results that even if there is no change in the key frame sequence, the *sim* value is affected by frame drops. This is because we use non-key frames in computing the secret at the key-frame level. Any change in non-key frames is reflected in the *sim* value.

Table 6 Sensitivity to frame dropping in a video in 'advertisement monitoring' scenario

Frames dropped	Key frame sequences	*sim* (%)
Zero	0,27,62,70,76,85,90,96,104,110,115,121	100
25	No change	50
25, 60	No change	13
25, 60, 72	No change	6
25, 60, 72, 116	No change	2
25, 60, 72, 116, 86	No change	1

Table 7 Sensitivity to frame replacing in a video

No. of frames replaced	Key frame sequences	*sim* (%)
0	0,27,62,70,76,85,90,96,104,110,115,121	100
1	0,27,62,70,76,79,80,87,92,99,105,112,119,121	2
2	0,27,62,70,76,85,90,92,93,97,98,105,112,119,121	1
3	0,14,15,51,64,65,75,85,90,96,104,105,106,112,119,121	1
4	0,27,62,70,73,74,82,86,87,92,99,105,109,110,115,121	0
5	0,12,13,49,64,74,75,76,77,85,90,96,104,110,113,114,120,121	0

Frame replacing To test whether our algorithm detects the tampered frames, we replace few video frames by some random frames. By random frames, we mean those frames whose luminance values are generated from a uniform random distribution. As shown in Table 7, when we replace a single frame (i.e. 79th frame in second row of Table), the *sim* value immediately drops down to 2%. This is due to the emergence of the tampered frame as a new key frame, and subsequently this change is propagated to the later frames which changes the whole key frame sequence; that eventually results in a entirely different signature. This shows that our method is very sensitive to frame replacement.

Frame adding We also tested the sensitivity of our method by inserting a real frame of ‘bus.mpg’ video in between 58th and 59th frames of ‘road.mpg’ video. The tampered sequence of frames is shown in Fig. 9. We computed signature for both the original and tampered video shots. The similarity between the two signatures is found to be zero. The inserted frame emerged as a new key frame which eventually changed the whole signature.

Frame reordering For frame reordering, we performed two types of test. First, the few frames are reordered randomly. Second, specific frame-pair are reordered to critically analyze the performance of our scheme.

Table 8 shows the results of random frame reordering. From Table 8, we observe that the *sim* value drops down to zero even with a single frame-reordering. Truly

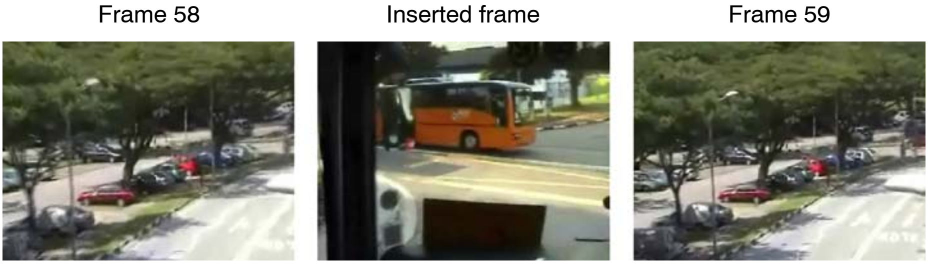


Fig. 9 Frame adding in ‘road.mpg’ video

Table 8 Sensitivity to frame reordering in a video

Frame reordered	Key frame sequences	*sim* (%)
Zero	0, 27, 62, 70, 76, 85, 90, 96, 104, 110, 115, 121	100
1	0, 27, 56, 57, 65, 75, 85, 90, 96, 104, 110, 112, 113, 119, 121	0
2	0, 6, 7, 41, 61, 62, 70, 76, 80, 81, 87, 92, 99, 105, 112, 117, 118, 121	0
5	0, 10, 11, 44, 45, 56, 57, 65, 75, 83, 84, 89, 93, 94, 98, 99, 105, 112, 113, 114, 118, 119, 121	0
10	0, 1, 5, 13, 14, 23, 24, 29, 30, 31, 32, 33, 52, 53, 63, 64 66, 68, 71, 72, 74, 75, 81, 82, 89, 94, 96, 97, 102, 103, 107, 108, 111 112 119 120 121	0

speaking the fall in *sim* value depends upon which frames are reordered. There are two possible cases. Case 1 is, if the reordering is limited to within a group of non-key frames, *sim* value is not affected much. However, in case 2, if the reordering takes place between the frames belonging to two different groups of non-key frames; it causes the change in the key frame sequence and hence the signature. The results shown in Table 8 are for the case 2.

To verify the case 1, we performed the test with specific frame-pair reordering. The results are given in Table 9. We achieved the expected results. The reordering of frame pairs 1-2, 1-10, 1-20 does not affect the *sim* value since these reorders are within a group of non-key frames. The bounding key frame numbers are 0 and 27. We also observed that reorder of 1-30 changed the *sim* value, since it belongs to case 2.

Shot reordering We also performed the test for shot reordering using 'bus.mpg' video. This video contains two shots. Using the original video, we tampered a new video by changing the shot sequence. We computed signatures for both videos and found them completely different (i.e. *sim* = 0). In Table 10, we show the key frame sequence for original and tampered videos. Some of these key frames are also shown in Fig. 10.

5.4 Spatial tampering

For spatial tampering, we performed two types of tests. First, using an exhaustive approach, we tested our method by randomly varying the cropping region size and

Table 9 Specific frame-pair reordering in a video

Frame-pair reordered	Key frame sequences	*sim* (%)
Zero	0, 27, 62, 70, 76, 85, 90, 96, 104, 110, 115, 121	100
1-2	0, 27, 62, 70, 76, 85, 90, 96, 104, 110, 115, 121	100
1-10	0, 27, 62, 70, 76, 85, 90, 96, 104, 110, 115, 121	100
1-20	0, 27, 62, 70, 76, 85, 90, 96, 104, 110, 115, 121	100
1-30	0, 1, 30, 31, 62, 70, 76, 85, 90, 96, 104, 110, 115, 121	6

Table 10 Sensitivity to shot-reordering in a video

Shot sequence	Key frame sequences	*sim* (%)
1-2	0, 85, 96, 109, 114, 118, 164, 171, 177, 190, 208, 218, 227, 230, 235, 241, 245, 250, 257, 267, 280, 285, 292, 297, 305, 328, 331, 335, 357, 360, 363, 368, 375, 395	100
2-1	0, 10, 21, 33, 40, 45, 51, 55, 60, 67, 77, 90, 95, 102, 107, 115, 138, 141, 145, 167, 170, 173, 178, 185, 208, 294, 305, 317, 322, 326, 372, 380, 386, 395	0

the crop value. Second, by tampering a specific object (i.e. face) in video. We describe the results for both in the following subsections.

Random cropping We performed random cropping tests on 'road.mpg' video for different size of cropping regions (or windows) in variable number of frames. We

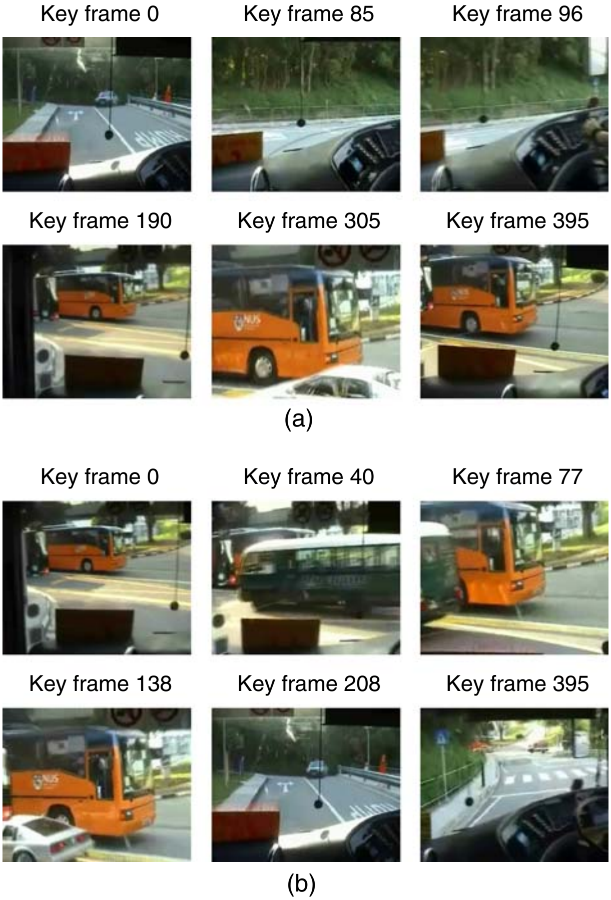


Fig. 10 Key frames in **a** Original video shot **b** Tampered video shot

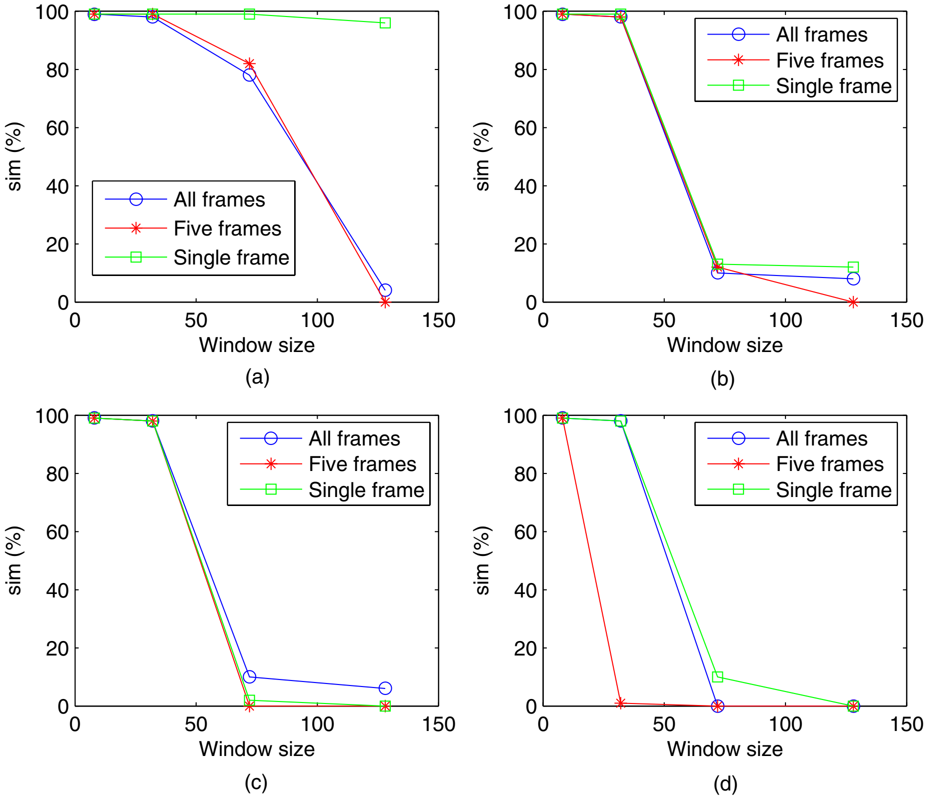
Table 11 Varying parameters for spatial tampering test

Window size	8 × 8, 32 × 32, 72 × 72, 128 × 128
Change in the luminance value of pixels	±100, ±50, ±20, ±10, or a random value.
Position of the cropping window	Beginning of any 8 × 8 block, or any random position.
Number of cropped frames	All, five, or single frame.
Cropping position	Same or different for all the frames.

performed test of cropping with many possible combinations. Some of the varying parameters in this test are given in Table 11.

We altered the pixel-luminance values by +10, +20, +50, +100, from block position (0,0) onwards in all frames, in five frames (Frame numbers 7, 40, 60, 79, 93) and in a single frame (10th). The results are shown in Fig. 11a–d. As expected, the results suggest that *sim* value decreases as the cropping window size and the crop value increases. The trends from these results are expressed as follows:

- $sim \propto \frac{1}{WindowSize}$
- $sim \propto \frac{1}{CropValue}$



(a)

(b)

(c)

(d)

Fig. 11 Spatial tampering with crop values **a** +10 **b** +20 **c** +50 **d** +100

As shown in Fig. 11a, the cropping a single frame with a crop value +10 does not affect the *sim* value. However, if it is performed for many frames, this alteration becomes significant in computing differential energy between the frames. Therefore, it changes the key frame sequence, and hence the signature. For a crop value of +20 and above (Fig. 11b–d), the drop in *sim* value is observed even for smaller cropping windows. From Fig. 11b–d, we also notice a drastic drop in the sim value after a particular crop size. It is due to the reason that larger change in pixel value disturbs the differential energy between the frames and hence the key frame sequence is also changed.

We illustrate it with an example. Suppose that two pixel values through temporal axis are 200 and 250. Since, the differential energy is computed by taking the square of difference in pixel values along temporal axis; therefore, before cropping, this

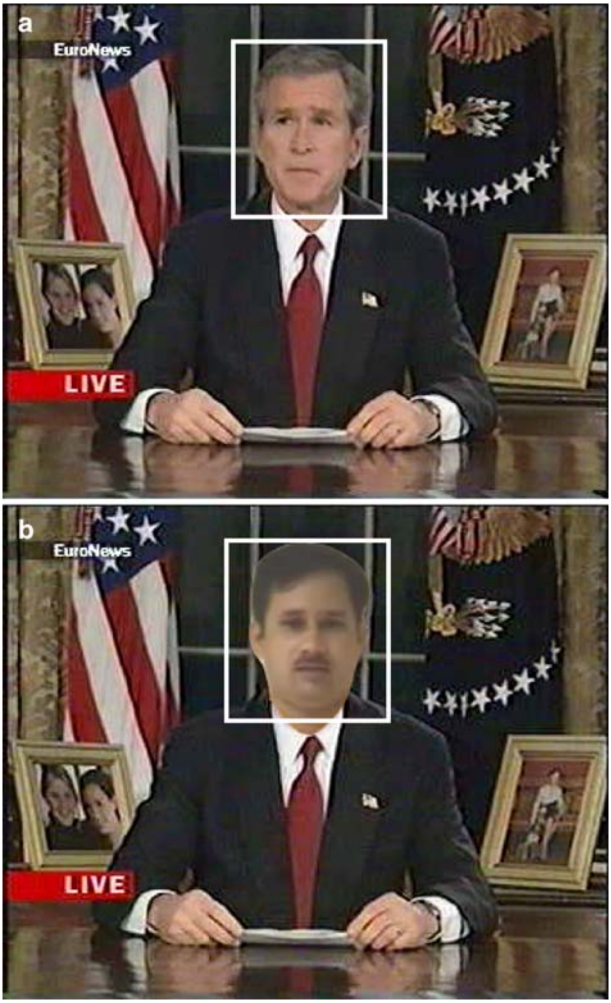


Fig. 12 Face tampering: **a** Original video frame [Source: EURONEWS website—http://www.euronews.com/] and **b** Tampered video frame

difference is $250 - 200 = 50$. The $50^2 = 2500$ contributes in differential energy. After the cropping takes place in form of increase of +50, these pixel values rise to $200 + 50 = 250$ and $250 + 50 = 300 \approx 255$ (since luminance value can be between 0 and 255). Now the difference of 255 and 250 is 5. The $5^2 = 25$ contributes to differential energy. That shows if this change is performed for many pixels, it may lead to major change in differential energy between frames and hence key frame sequence may also change.

Face tampering We also performed a specific test of face tampering by replacing only the face of a person with the face of another person in all the frames of 'face.mpg' video. This type of intentional tampering is often professionally done with a good or bad intent. We have tampered it for the algorithm testing purpose.[1]

We use a face detector to identify the face in video. Figure 12 shows the window where the faces are detected in two video clips. Our method generates a *Weight frame* which contains the weights for each block of the video frame. We assign a higher weight ($W = 10$) to the region where the face is detected. We compute and compare the signatures of two video clips. The test result shows the *sim* = 47 i.e. 53% tampering is found. Note that the region (rectangle) where the face is found is only 10% of the whole video frame. We also observed the change in the key frame sequences (0, 10, 46 in original video and 0, 8, 46 in tampered video). The change in key frame sequence results in different signatures.

6 Conclusion

Security and robustness are the two equally important issues in the video authentication problem. In this paper, we propose a novel technique for video authentication that ensures the integrity of the video based on the atypical use of cryptographic secret sharing. The experimental results show that the proposed method is effective against malicious spatial cropping and frame jittering which is highly needed for video identification. The proposed scheme utilizes three hierarchical levels of a video in a scalable manner. The proposed scheme suitably scales down the authentication process to shot level to achieve robustness against frame dropping in video streaming scenario. The algorithm can also detect specified region tampering and has been successfully tested over face tampering in a video.

The future work is to further analyze the algorithm to localize the tampering, and also to recover the losses due to tampering into the secret sharing framework with the aid of forward error correction techniques. Other issues like robustness against video processing and geometric operations will also be addressed in the future work.

[1] The authors would like to explicitly state that the image used in this experiment has solely been used for the academic research purpose without any commercial or malicious intent.

References

1. Atrey PK, Yan WQ, Chang EC, Kankanhalli MS (2004) A hierarchical signature scheme for robust video authentication using secret sharing. In: International multi-media modelling conference, Brisbane, Australia, pp 330–337
2. Celik MU, Sharma G, Tekalp AM, Saber ES (2002) Video authentication with self-recovery. In: SPIE: security and watermarking of multimedia contents IV, vol 4675. San Jose, CA, pp 531–541
3. Daniel C, Bijan M (2002) Watermarking for self-authentication of compressed video. In: IEEE international conference on image processing, vol II. Rochester, NY, pp 913–916
4. Dittman J, Steinmetz A, Steinmetz R (1999) Content based digital signature for motion pictures authentication and content-fragile watermarking. In: IEEE international conference on multimedia computing and systems, vol 2. Firenze, Italy, p 204
5. Divakaran A, Radhakrishnan R, Peker KA (2002) Motion activity based extraction of key frames from video shots. In: IEEE international conference on image processing, vol I. Rochester, NY, pp 932–935
6. Du R, Fridrich J (2002) Lossless authentication of MPEG-2 video. In: IEEE international conference on image processing, vol II. Rochester, NY, pp 893–896
7. Eskicioglu AM (2001) A prepositioned secret sharing scheme for message authentication in broadcast networks. In: The 5th joint working conference on communications and multimedia security, Darmstadt, Germany, pp 363–373
8. He D, Sun Q, Tian Q (2003) A semi-fragile object based video authentication system. In: IEEE international symposium on circuits and systems, vol III. Bangkok, Thailand, pp 814–817
9. Lang ST, Peticolos FA (2003) Authentication of MPEG-4 data: risks and solutions. In: SPIE international conference on security and watermarking of multimedia contents V, vol 5020. Santa Clara, CA, pp 452–461
10. Lin CY, Chang SF (1999) Issues and solutions for authenticating MPEG video. In: IEEE international conference on acoustics, speech and signal processing, pp 54–65
11. Mobasseri BG, Sieffert MJ, Simard RJ (2000) Content authentication and tamper detection in digital video. In: IEEE international conference on image processing, vol I. Vancouver, BC, Canada, pp 458–461
12. Qi Y, Liu T, Hauptmann A (2004) Supervised classification of video shot segmentation. In: IEEE international conference on multimedia and expo, vol II. Baltimore, MD, pp 689–692
13. Quisquater JJ, Marc J, Macq B, Degand N, Bernard A (1997) Practical solutions to authentication of images with a secure camera. In: SPIE international conference on storage and retrieval for image and video databases V, vol 3022. San Jose, CA, pp 290–297
14. Radhakrishnan R, Xiong Z, Memon ND (2003) On the security of visual hash function. In: SPIE international conference annual symposium on electronic imaging: Science and technology, vol 5020. Santa Clara, CA, pp 644–652
15. Shamir A (1979) How to share a secret. Commun ACM 22(11):612–613
16. Stinson DR (1995) Cryptography: Theory and Practice, 1st edn, Chap 11. CRC Press, Boca Raton
17. Tzeng CH, Tsai WH (2001) A new technique for authentication of image/video for multimedia applications. In: Multimedia and security workshop at ACM multimedia, Ottawa, Canada
18. Yan WQ, Kankanhalli MS (2003) Motion trajectory based video authentication. In: IEEE international symposium on circuits and systems, vol III. Bangkok, Thailand, pp 810–813
19. Yin P, Yu HH (2002) A semi-fragile watermarking system for MPEG video authentication. In: International conference on acoustic speech and signal processing, vol IV. Orlando, FL, pp 3461–3464
20. Zhang HJ, Wu J, Zhong D, Smoliar S (1997) An integrated system for content-based video retrieval and browsing. Pattern Recogn 30(4):643–658

Pradeep Kumar Atrey received B.Tech. from Harcourt Butler Technological Institute, Kanpur, India in 1991 and M.S. from Birla Institute of Technology and Sciences, Pilani, India in 1999. He has worked as a lecturer at Delhi College of Engineering, University of Delhi and at C. R. State College of Engineering, Murthal (Haryana), India from 1992 to 2002. He pursued his PhD at School of Computing, National University of Singapore from July 2002 to July 2006. Currently, he is a Research Fellow at School of Information Technology and Engineering, University of Ottawa, Ottawa, Canada. His research interest includes Multimedia security and Surveillance, and Video and Audio Processing.

Wei-Qi Yan was a Research Fellow in School of Computing, National University of Singapore from 2001 to 2005. His research interests are multimedia systems and media security. Yan has a Ph.D. in computer engineering from Academia Sinica.

Mohan Kankanhalli obtained his BTech (Electrical Engineering) from the Indian Institute of Technology, Kharagpur and his MS/PhD (Computer and Systems Engineering) from the Rensselaer Polytechnic Institute. He is a Professor at the School of Computing at the National University of Singapore. He is on the editorial boards of several journals including the ACM Transactions on Multimedia Computing, Communications, and Applications, IEEE Transactions on Multimedia, ACM/Springer Multimedia Systems Journal and the IEEE Transactions on Information Forensics and Security. His current research interests are in Multimedia Systems (content processing, retrieval) and Multimedia Security (surveillance, authentication and digital rights management).