J. Vis. Commun. Image R. 40 (2016) 502–515

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

# Multi-layer authentication scheme for HEVC video based on embedded statistics ☆

Yiqi Tew [a], KokSheik Wong [a,*], Raphael C.-W. Phan [b], King Ngi Ngan [c]

[a] Department of Computer System and Technology, Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia
[b] University Multimedia, Malaysia
[c] The Chinese University of Hong Kong, Hong Kong

ARTICLE INFO

*Article history:*
Received 4 January 2016
Revised 5 July 2016
Accepted 21 July 2016
Available online 26 July 2016

*Keywords:*
Authentication
Multi-layer
Information hiding
Compressed video
HEVC
Statistical properties

ABSTRACT

A multi-layer authentication scheme for HEVC compressed video is proposed. The combination of CU sizes, which is unique to HEVC and sensitive to video manipulation, is considered along with other elements in the HEVC coding standard to generate the authentication tag. Temporal dependency was enforced, where the authentication tag generated in one slice is embedded into its subsequent slice. By design, the authentication tag is repeatedly but selectively embedded into various elements in a HEVC video, including nonzero DCT coefficients, QP parameter values, and prediction modes, depending on the bit segment in the generated tag. Our scheme offers three layers of authentication to detect and localize the tampered regions in a HEVC video, as well as verifying the source/sender of the video using a shared secret key. Video sequences from various classes (resolutions) are considered to verify the performance of the proposed multi-layer authentication scheme. Results show that, at the expense of slight degradation in perceptual quality, the proposed scheme is robust against several common attacks. A functional comparison is performed between the proposed multi-layer authentication scheme and the conventional schemes.

## 1. Introduction

Digital video has become an important part of our daily life thanks to the widely accepted standardization of video coding formats and their successful deployments in various applications. We watch movies over the Internet, record video using car DVR (digital video recorder), establish video conference across heterogeneous network environments, etc. However, these videos can be easily manipulated (e.g., trimmed, cropped, re-compressed) due to the availability of high performance personal computer at affordable prices and user-friendly yet powerful video editing software [1]. As a result, the integrity of digital video and its origin become implausible. Hence, a video needs to be authenticated so that its source can be confirmed to be someone trustworthy and its content can be verified to be genuine prior to consumption or broadcasting [2].

Unlike its success in providing entertainment [3], the viability of digital video as evidence in the judicial process has been largely unprecedented. And yet we see an increasing number of videos from personal cameras or mobile phones being released in social media corresponding to incidents, e.g., road bullying. Digital evidence is often ruled inadmissible by courts because it is owned without authentication or its authenticity cannot be verified [4]. According to the guideline stipulated in [5], it is necessary to demonstrate how evidence is authenticated and to show the integrity of each process through which the evidence was obtained. The evidence should be preserved from any third party who is able to repeat the same process and attain the same result as that presented to the court. Therefore, implementing a secure authentication scheme to confirm the authenticity of viable video evidence is imperative. It is also important to prevent any digital video tailored for causing hatred or benefiting a certain party.

When handling compressed video, authentication is commonly achieved in two ways, namely, video labeling and data embedding [6]. Specifically, in video labeling, the authentication code is appended to the video for integrity verification. This code can be

☆ This paper has been recommended for acceptance by Zicheng Liu.
* Corresponding author.
*E-mail addresses:* koksheik@um.edu.my, KokSheik.Wong.MY@ieee.org (K. Wong).
*URL:* http://mspih.fsktm.um.edu.my (K. Wong).

http://dx.doi.org/10.1016/j.jvcir.2016.07.017
1047-3203/© 2016 Elsevier Inc. All rights reserved.

generated by a hash function using a set of features, such as those extracted or derived from the video, as the input. On the other hand, for data embedding based approach, the authentication code is imperceptibly embedded into the video stream rather than appended to it.

Specifically, Tartary et al. proposed an labeling based authentication scheme for any digital content by utilizing the Reed-Solomon code [7]. They demonstrated the design based on the list of recoverable codes in network stream distribution. An cryptographic function is introduced along with digital signature and hash function to ensure the robustness of the designed scheme. Later, Ren et al. introduced a video labeling based authentication scheme with loss-tolerant feature [8,9]. They combined a cryptographic fingerprint and video to achieve video authentication. However, this authentication scheme compromises on minor latency (i.e., access to the fingerprint) and video file size increment in the processed video. Baek et al. then designed a labeling based authentication framework through public key infrastructure for video broadcasting network [10]. They introduced an identity-based signature to authenticate online and offline broadcasted videos, and improved the performance of [11]. With the same spirit, Song et al. put forward an interactive content based authentication scheme using labeling for video streaming [12]. Their design generates levels of signature in the chosen video slices, multiple authentication paths as well as authenticating information on network packets. It is reported that their scheme is of high tolerance against packet loss.

While offering attractive performances, video labeling based authentication can hardly provide the security feature to authenticate video due to the nature of the code appending process, which fails to prevent the code from being copied, manipulated or counterfeited. This shortcoming can be overcome by means of data embedding. Roy et al. realized a video authentication approach in hardware by using field programmable get arrays [13], where the authentication information is embedded to resist against cover-up and cropping segment attacks. An authenticated video under Roy et al.'s scheme can be easily adapted in common video standards with minor quality degradation, but may not be viable for video of higher resolution due to the high computational complexity. On the other hand, Wei et al. proposed an authentication scheme in the scalable video code streams, where the authentication codes are encapsulated in the network abstraction layer unit [14]. Their proposed scheme is efficient in detecting content-preserving manipulation attack (e.g., recompression), but vulnerable to content-changing manipulation (e.g., color or luminance) attack.

Along with the similar direction, some researchers utilize semi-fragile watermarking scheme to randomly embed authentication code into coefficients in spatial domain [15]. These schemes detect and localize tampering (e.g., cropping, insertion) region, but not realized in temporal domain (i.e., video). Later, Qi et al. presented a similar scheme by utilizing the singular-value-based and private-key-based sequences to generate content-dependent authentication codes [16]. These codes are embedded through an adaptive quantization method under discrete wavelet transformation in spacial domain, and it can be easily extended to color domain. Furthermore, researchers then proposed data embedding based authentication by utilizing histogram shifting technique in the spatial domain [17], or manipulating motion vector [18], coefficients [19] and macroblock [20] in the compressed domain (e.g., MPEG2, H.264/AVC) [21]. However, these schemes are not implemented in the latest video standard, i.e., HEVC (High Efficiency Video Coding), which is anticipated to replace H.264/AVC standard especially when more high resolution (e.g., 4 K) cameras, display devices and video contents are available.

To the best of our knowledge, the conventional schemes (e.g., [8–11]) authenticate one video slice at a time, which is essentially the same as handling sequence of still images. The temporal dependency, on which all video compression techniques exploit, is not considered. Therefore, in this work, we propose an authentication scheme that exploits the dependency between two consecutive slices, where statistical information in the previous slice is utilized to guide the embedding process in the current slice. We focus on multi-layer data embedding based authentication for detecting and localizing tampered regions, as well as verifying the origin of the HEVC video. By considering the capability of current HEVC decoder, the proposed first and second layers of authentication can be conveniently implemented to verify genuineness of a video since the operations involved are of low complexity. On the other hand, the third layer of authentication, in which case the secret key is required, can be deployed to verify the legitimacy of a video, e.g., relating to the evidence in court case.

In the proposed scheme, we emphasize on the utilization of video content statistics, where all the coding elements in each video slice are considered to generate the authentication information. During encoding, the video content (i.e., visual information) is very sensitive to the compression mechanism under the HEVC standard. Specifically, encoding at different bitrate yields significantly different set of statistics. In particular, the act of manipulating or modifying any video element changes the video statistics drastically, and hence the sensitivity of these statistics is exploited for tamper detection in this work, which is the significant part of our proposed authentication scheme.

The remainder of this paper is organized as follows: Section 2.1 briefly reviews the coding structure of HEVC. Section 3 puts forward our multi-layer authentication scheme. Section 4 discusses the performance of the proposed multi-layer authentication scheme in terms of video quality, robustness and sensitivity when considering various classes of test video and presents the functional comparison against conventional schemes. Section 5 concludes this paper.

## 2. Preliminaries

### 2.1. HEVC standard

HEVC is the latest video coding standard published by ITU-T VCEG and ISO/IEC MPEG [22]. The main achievement of the HEVC standard is its significant improvement in compression performance when compared to the previous state-of-the-art standard, i.e., H.264/AVC [23], with at least 50% reduction in bitrate for producing video of similar perceptual quality [22]. HEVC standard is designed to address essentially all existing applications of H.264/AVC. It achieves two additional major achievements, namely: (a) handle video of higher resolution by introducing larger coding unit sizes, and (b) capitalize on parallel processing architecture in the video encoder design for suppressing encoding time.

HEVC introduced several novel features to achieve higher video compression performance, including various coding unit sizes, more intra prediction modes, residual quad tree, sample adaptive offset, tiles and wavefront processing [24]. Among these novel features, the implementation of variable coding unit sizes, quantization parameters and prediction modes are exploited in this paper for authentication purposes by means of data embedding.

In HEVC encoder, a video is treated as a sequence of pictures, where these pictures are labeled as I-(intra), P-(predicted) and B-(bi-directionally predicted) slices, depending on the order in which they appear. Each slice consists of certain number of CTU (coding tree units), while each CTU consists of some number of CU (coding units) with size of $64 \times 64, 32 \times 32, 16 \times 16$ or $8 \times 8$ pixels. Each $8 \times 8$ CU can be further split into $4 \times 4$ in the prediction process. The availability of CU in various sizes allows the video encoder to encode each part of the video slice based on its local characteristic (i.e., spatial activity). The encoder decides the CU size and the quantization value in each CTU based on the desired bitrate. In particular, due to the quantization process, a region with high spatial activity (e.g., water waves) requires smaller CU size to precisely capture the variation in pixel intensity values. On the other hand, a smooth region (e.g., background or cloudless sky) can be encoded by using larger CU size. In the case of low bitrate (e.g., 10 kbps), a large quantization value (e.g., QP = 40) is utilized to encode every CTU with larger CU sizes, which leads to quality degradation and smaller video file size. On the other hand, for high bitrate (e.g., 100 Mbps), small quantization value (e.g., QP = 12) is utilized and most CTUs are coded in smaller CU sizes for representing the region without compromising on perceptual video quality, but at the expense of larger video file size.

The prediction and transformation processes utilize the CU structure to perform intra/inter prediction, DCT (Discrete Cosine Transformation) and quantization. The CU utilized in the prediction and transformation processes are called PU (prediction unit) and TU (transform unit), respectively. Specifically, in I-slices, CU can only be coded by using squares, which include $64 \times 64, 32 \times 32, \ldots, 4 \times 4$ pixels. On the other hand, in P and B-slices, CU can be encoded by using all possible arrangements, including $2N \times 2N$ , $2N \times N$ , $N \times 2N$ , $N \times N$ for $N \in 4, 8, 16, 32$ and AMP (Asymmetry Motion Partition), which can assume the dimension of $2N \times nU, 2N \times nD, 2N \times nL$, or $2N \times nR$. The implementation of AMP in HEVC provides better prediction reference and less bitstream size overhead for PU that contains slight movement at either the upper, lower, left or right part of a CU in P- or B-slice. Here, each CU is encoded with a depth value $d$, to indicate the $N$ value in CU size definition. The depth value, $\delta \in \{0, 1, 2, 3\}$ signifies that $N = 64/2^{\delta}$ (e.g., CU of size $(64/2^{\delta}) \times (64/2^{\delta}) = 32 \times 32$ are considered for $\delta = 1$), except for $\delta = 3$ where both $8 \times 8$ and $4 \times 4$ blocks are included.

Each PU defines a slice region that shares the same prediction mode (i.e., intra, inter, skip and merge) [25]. Intra prediction defined in HEVC allows 33 angular prediction directions (i.e., modes) and two non-angular modes, namely DC and planar. Current PU's intra prediction is obtained through the extrapolation of values derived from the reference pixels of the neighboring PU's, a process which requires numerous arithmetic operations per predicted pixel value. On the other hand, inter prediction encodes PU by storing the motion vector, which points to the position of the matching PU in the reference slice, as well as the residual values, which are the differences (prediction errors) between the reference PU and current PU.

## 3. Authentication scheme

Our proposed scheme aims to detect and localize the tampered region(s) in a HEVC compressed video by means of data embedding and exploiting the dependency in the temporal axis. Specifically, the authentication code (hereinafter referred to as tag) is generated and embedded into the video. Fig. 1 shows the process flow of the proposed multi-layer authentication scheme, which consists of the following four steps: Tag Generation, Tag Implantation, Tag Alteration and Tag Verification. In tag generation, the extracted features from the video and the secret key are combined then fed into a hash function as detailed in Section 3.1. The tag implantation process using data embedding technology is detailed in Section 3.2. Tag alteration by means of masking or skipping, as well as the embedding schemes is described in Section 3.3. To validate a video, the tag is verified in three layers of authentication as detailed in Section 3.4 (see Fig. 2).

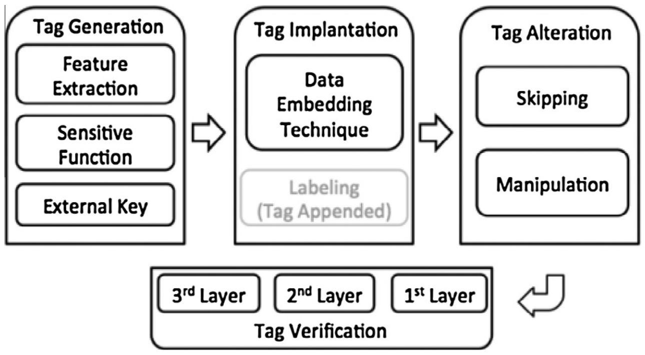


**Fig. 1.** Proposed multi-layer authentication scheme.

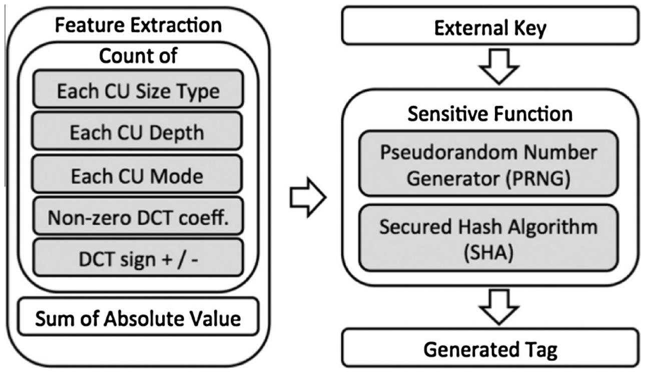


**Fig. 2.** Tag generation.

### 3.1. Tag generation

In video authentication, a generated tag must be unique as well as sensitive to its input, and its genuineness must be verifiable by anyone who has the secret key. To fulfill these requirements, we exploit the unique statistical features of the video content and a hash function (e.g., SHA2) to generate the tag, which is in turn embedded into the video.

#### 3.1.1. Feature extraction

Several video features are considered to serve as the input for tag generation. These features, including the size types, depths and modes in every CU, as well as non-zero DCT coefficient values, are extracted from each CTU in every video slice. Recall from Section 2.1 that in every video slice, HEVC divides each CTU into some combination of CUs in different sizes. To facilitate the discussion, let $\gamma_m \in \{2N \times 2N, 2N \times N, N \times 2N, N \times N, 2N \times nU, 2N \times nD, nL \times 2N, nR \times 2N\}$ refer the category of CU size, $\delta_m \in \{0, 1, 2, 3\}$ refers to the depth of quad tree decomposition, and $\pi_m \in \{\text{intra, inter}\}$ refers to the prediction mode in the $m$-th CTU, where $m \in \{1, 2, \ldots, M\}$ for

$$M = \lceil (width/64) \rceil * \lceil (height/64) \rceil. \quad (1)$$

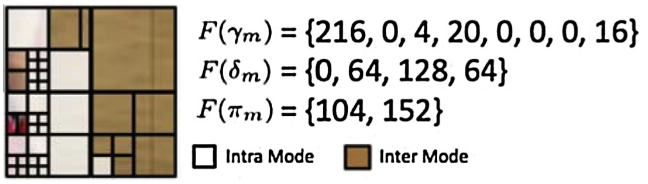

$F(\gamma_m) = \{216, 0, 4, 20, 0, 0, 0, 16\}$
$F(\delta_m) = \{0, 64, 128, 64\}$
$F(\pi_m) = \{104, 152\}$
Intra Mode
Inter Mode

**Fig. 3.** Illustration of feature extraction.

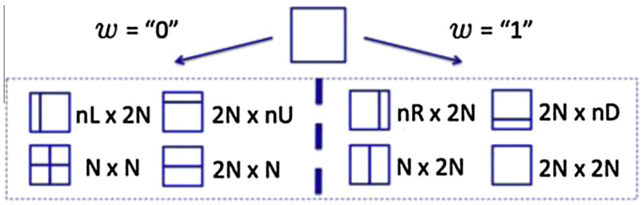


**Fig. 4.** Mapping rules for data embedding using CU size.

The frequency of occurrences for $\gamma_m, \delta_m$ and $\pi_m$ in the $m$th CTU are computed and referred to as $F(\gamma_m), F(\delta_m)$ and $F(\pi_m)$, respectively.

In this work, the number of $4 \times 4$ pixel blocks is considered, i.e., the block dimension divided by 16. Suppose the 3-rd CTU is being processed (i.e., $m = 3$). Given one CU of size $32 \times 32$, when $\delta_3 = 1$, the corresponding frequencies of occurrences are $F(\gamma_3 = 2N \times 2N) = F(\delta_3 = 1) = 64$ since there are exactly 64 units of $4 \times 4$ pixel block within it. Similarly, for a CU of size $16 \times 16$ with intra mode, the frequencies of occurrence $F(\pi_3 = \text{intra}) = 16$ since there are exactly 16 units of $4 \times 4$ pixels within it. For further illustration, frequencies of occurrences for $F(\gamma_m), F(\delta_m)$ and $F(\pi_m)$ are calculated based on the example given in Fig. 3. Here, $F(\gamma_m = 2N \times 2N) = 216$ since there are 1 CU of size $32 \times 32$ (i.e., 64 units of $4 \times 4$), 7 CUs of size $16 \times 16$ (i.e., 112 units of $4 \times 4$) and 10 CUs of size $8 \times 8$ (i.e., 40 units of $4 \times 4$). On the other hand, $F(\pi_m = \text{intra}) = 104$ since there are 104 units of $4 \times 4$ block coded in intra mode while $F(\pi_m = \text{inter}) = 152$ because there are 152 units of $4 \times 4$ block coded in inter mode.

For features extraction, let $\gamma_m^{\max_1}$ and $\gamma_m^{\max_2}$ be the two most frequently occurring CU categories in the $m$th CTU. The difference in frequency of occurrences between them, denoted by $\Gamma(m)$, is computed as $\Gamma(m) = F(\gamma_m^{\max_1}) - F(\gamma_m^{\max_2})$. Similarly, let $\delta_m^{\max_1}$ and $\delta_m^{\max_2}$ be the two most frequently occurring depths in the $m$-th CTU, and the difference in frequency, denoted by $\Delta(m)$, is computed as $\Delta(m) = F(\delta_m^{\max_1}) - F(\delta_m^{\max_1})$. Similarly, for prediction mode, the difference between frequency of using intra and inter in the $m$-th CTU, denoted by $\Pi(m)$, is computed as $\Pi(m) = |F(\pi = \text{intra}, m) - F(\pi = \text{inter}, m)|$. In addition, in each CTU, the count of non-zero DCT coefficient $cnz(m)$, the sum of absolute value of non-zero DCT coefficient $sav(m)$, and the difference between the frequency of occurrences for positive and negative signs $s(m)$ are computed. Note that these entities highly sensitive to re-compression and only available in the HEVC standard (i.e., $\gamma_m$ and $\delta_m$), which will change drastically when encoded in different bitrate or when different content is encoded.

### 3.1.2. Secret key

The (secret) key $K$ with a specific length is required to verify the origin (i.e., sender) of a video. This key must be owned by both parties (i.e., sender and receiver) to generate the same tag for verification purpose. Note that the secret key is not revealed during verification because only the generated tag is compared against the embedded tag. In case the origin of the video need not be verified, the secret key can be conveniently replaced by any value such as DCT coefficient values and motion vectors.

### 3.1.3. Sensitive function

A sensitive function (e.g., hash function, pseudo-random number generator) is required to generate a unique tag from the extracted features and shared secret key $K$. The tag generated by this function should differ significantly even when the inputs (e.g., statistics of video) are similar but not identical. It should be practically impossible to analyze this tag for inverting the mapping process, that is, to obtain the input value from the tag. In this work, a cryptographic hash function $H$, namely, SHA2 [26], is utilized to meet the aforementioned requirements. For each video slice, the extracted feature values, viz., $\Gamma(m), \Delta(m), \Pi(m), cnz(m), sav(m), s(m)$ for all CTUs ($M$ in total) as well as the shared secret key $K$ are concatenated to form the input for the hash function $H$ for generating the tag $w$. The hash function $H$, input and output $w$ are related as expressed in Eq. (2), where $\theta||\zeta$ concatenates $\theta$ and $\zeta$ together. Note that the length of the tag $x$ depends on the applied hash function and in our case, the output tag is 32 bytes since SHA256 [26] is considered.

$$w = H\Big((\Gamma(m))_{m=1}^{M}||(\Delta(m))_{m=1}^{M}||(\Pi(m))_{m=1}^{M}||(cnz(m))_{m=1}^{M}|| \times (sav(m))_{m=1}^{M}||(s(m))_{m=1}^{M}||K\Big) \quad (2)$$

## 3.2. Tag implantation

Four data embedding techniques are deployed to achieve high imperceptibility for embedding tag into the HEVC compressed video. These techniques utilize different elements in the HEVC encoded video, including the CU type, non-zero DCT coefficient, quantization parameter, prediction type and motion vector, to selectively and repeatedly embed the generated tag.

### 3.2.1. Coding unit size

In HEVC encoder, the RDO (rate distortion optimizer) decides the CU sizes to achieve the best compression ratio based on the desired bitrate. In our proposed multi-layer authentication scheme, instead of using the size determined by RDO, we force the size of CUs in slice $S_{n+1}$ to embed the tag $w$, which is computed from the previous slice $S_n$ based on a predefined mapping rule. An example of the mapping rule is shown in Fig. 4 [27].

The CU sizes are divided into two categories, where one encodes '0' and the other encodes '1'. In particular, category '0' includes $2N \times N, 2N \times nU, nL \times 2N$, and $N \times N$ pixels, while category '1' includes $N \times 2N, 2N \times nD, nR \times 2N$, and $2N \times 2N$. In other words, the CU size in $S_{n+1}$ can be $N \times 2N, 2N \times nD, nR \times 2N$, or $2N \times 2N$ for $w_l = 1$ where $l = 1, 2, \ldots, 256$, and vice versa, as depicted in Fig. 4. For instance, if the CU size decided by RDO is $16 \times 8$ and $w_l = 1$, then our proposed scheme will force the RDO to recalculate the required bitrate (i.e., cost) for $8 \times 16, 16 \times 16$, and two AMP's (i.e., $2N \times nD, nR \times 2N$), then choose the CU size that results in the lowest cost. For CU with larger size (e.g., $32 \times 32$), it is justifiable to encode it by using some combination of blocks with smaller sizes (e.g., two $32 \times 16$, four $16 \times 16$, etc.), because in this case, a smooth block is merely decomposed into combination of smaller blocks, which are conventionally considered for encoding region of higher spatial activity. The tag $w$ is sequentially and repeatedly embedded into all CUs, following the order from top-left to bottom-right (i.e., Z-scanning) as stipulated in the HEVC standard [22]. This approach maintains the video quality at the expense of slight increment in bitstream size. It should be noted that, in this

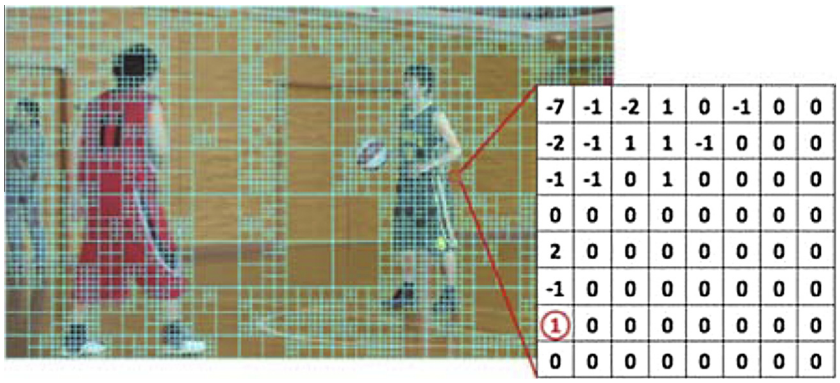


**Fig. 5.** Modified LSB of the non-zero DCT coefficient.

work, smaller blocks are not combined into a larger block to maintain the video quality at the expense of slight file size increment.

### *3.2.2. Non-zero DCT coefficient*

Here, we utilize LSB of non-zero DCT coefficients to embed the tag without causing significant quality degradation. To minimize distortion, we choose the last non-zero DCT coefficient (with respect to the scanning order in use) of each CU in every CTU, as shown in Fig. 5. The selected DCT coefficient $c$ is modified to an even integer for embedding $w_l = 0$, and vice versa.

### *3.2.3. QP (Quantization Parameter)*

During encoding, RDO utilizes the QP value to achieve the desired bitrate. In order words, it determines the quality of video, where smaller QP value leads to higher video quality, and vice versa. HEVC encodes each CTU with different QP value based on the predefined QP value range as stipulated in the configuration file. Here, we force the QP of each CTU to embed $w_l$ by modifying the offset range during the encoding process. The RDO calculates the cost of each CTU (i.e., total bit requires to code the CTU) based on the QP value with the selected offset. Odd QP values will be utilized in the calculation when embedding $w_l = 1$, and vice versa. Fig. 6 shows the possible QP values for embedding $w = 11111000\cdots$.

### *3.2.4. Prediction type*

Video compression is closely tied with the implementation of various prediction methods, which can be coarsely divided into two approaches: prediction within the video slice itself (intra) and among few neighboring slices (inter). We exploit these two

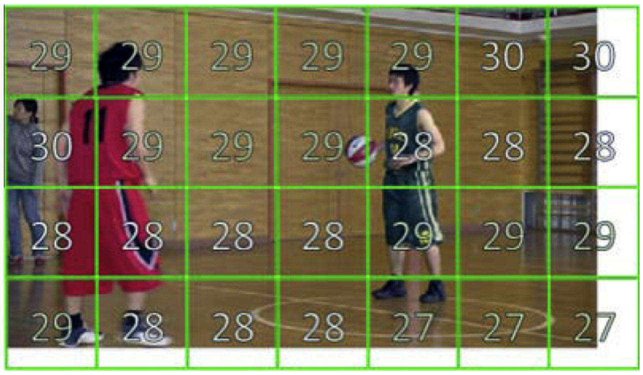


**Fig. 6.** Quantization value for each CTU in a slice.

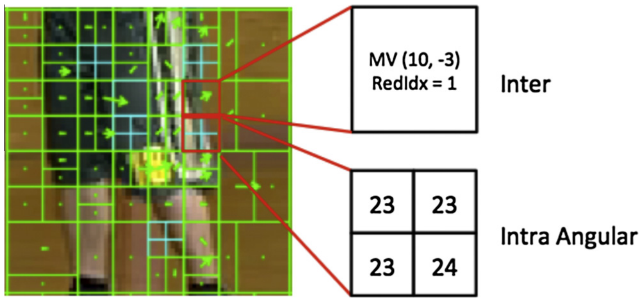


**Fig. 7.** Intra and inter prediction mode decision in a slice.

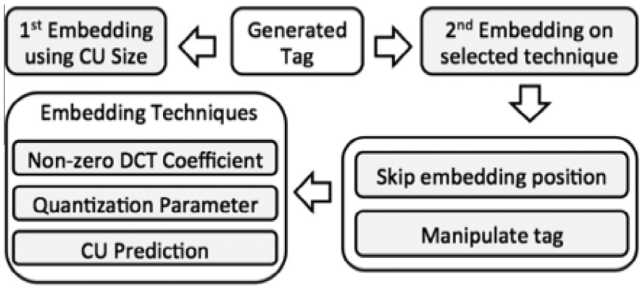


**Fig. 8.** Tag implantation and alteration process.

approaches of prediction to represent the tag $w_l \in \{1, 0\}$. Again, RDO is set to consider only the CTU cost for all 34 types of intra prediction (see Section 2.1) while ignoring those for inter prediction when $w_l = 0$. On the other hand, only the costs for inter prediction are considered when $w_l = 1$. Fig. 7 shows the selected CU to embed $w_l \in \{0, 1\}$ in inter prediction mode with $MV = (10, -3)$ using RedIdx (i.e., reference slice index) = 1, and intra prediction mode using mode 23 and 24.

## *3.3. Tag alteration*

In our proposed multi-layer authentication scheme, the first slice $S_1$ of the video is utilized for generating the tag $w$. This tag is conveyed to the next slice $S_2$ via two embedding steps. The first step utilizes the CU size embedding technique detailed in Section 3.2.1 to embed the tag by modifying the CU size in $S_{n+1}$. The second step embeds the tag by using three other embedding techniques by considering the bit segment in each byte of the tag. Fig. 8

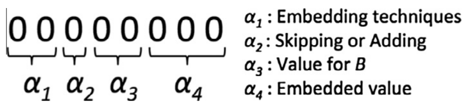

0 0 0 0 0 0 0 0
$\alpha_1$ $\alpha_2$ $\alpha_3$ $\alpha_4$
$\alpha_1$ : Embedding techniques
$\alpha_2$ : Skipping or Adding
$\alpha_3$ : Value for B
$\alpha_4$ : Embedded value

**Fig. 9.** Bit segment of every byte in tag.

**Table 1**
Syntax of bit pattern in every byte of tag.

| Bits | $\alpha_1$: embedding technique | $\alpha_2$: mode | $\alpha_3$: value |
|---|---|---|---|
| 00 | No embedding | Skipping | 0 |
| 01 | Coefficient | Adding | 1 |
| 10 | QP | – | 2 |
| 11 | Prediction mode | – | 3 |

shows the aforementioned embedding steps. The purposes of having two embedding processes are to: (a) enable a quick way to check the authenticity of a given video, and (b) localize the tampered regions, with precision up to the CU size. Note that these processes can be performed without using the secret key *K*.

The tag is divided into non-overlapping segments where each segment is processed and embedded one at a time. As an illustration, Fig. 9 shows an 8-bit segment of the tag, which will be processed by the second embedding process. Specifically, the second embedding process determines the technique to be applied (for embedding), the skipping of positions, and the manipulation on (i.e., masking) the tag itself. These processes are included to complicate the act of mimicking.

### 3.3.1. Selection

Given a segment, the first two bits (denoted by $\alpha_1$) determine the embedding technique to deploy. All four possible combinations are listed in Table 1. Specifically, when $\alpha_1 = 01, \alpha_4$ is embedded into the last non-zero DCT coefficient of the next three CUs in the next CTU(s). In the case of $\alpha_1 = 10, \alpha_4$ is embedded into the quantization parameter of the next three CTUs. For $\alpha_1 = 11$, prediction mode for the next three CUs are utilized to encode $\alpha_4$. For $\alpha_1 = 00$, no embedding takes place.

### 3.3.2. Manipulation

We complicate the embedding process to discourage mimicking of the tag by skipping selected embedding locations (synchronization) or adding the value $\alpha_3$ to the bit segment $\alpha_4$ (masking) prior to actual data embedding. For instance, once the type of embedding technique (venue) has been decided (signaled by $\alpha_1$), when $\alpha_2 = 1, \alpha_3$ stipulates how many of the eligible embedding venues (of the decided type) will be skipped before $\alpha_4$ is embedded. For example, when $(\alpha_1, \alpha_2, \alpha_3) = (01, 0, 10)$, the following 2 non-zero coefficients will be skipped, and $\alpha_4$ will be embedded by using the last nonzero coefficients in the 3rd, 4th, and 5th non-zero coefficients. On the other hand, when $\alpha_2 = 1, \alpha_4$ is added to $\alpha_3$ before being embedded into the selected location. Due to the problem of overflow, $\text{mod}(\alpha_4 + \alpha_3, 2)$ will be embedded. The embedding process continues until all bit segments in the tag *w* are processed.

The pseudo-code of our proposed multi-layer authentication scheme is presented as Algorithm 1, which includes the tag generation, implantation and alteration processes in the HEVC encoder. The tag is generated in the *n*-th slice, i.e., $S_n$, and embedded into $S_{n+1}$.

**Algorithm 1.** Pseudo-code for tag generation, implantation and alteration

```
Input: $K$
Output: $w$
initialization ; $m \leftarrow 0$ ; $n \leftarrow 0$ ; $i \leftarrow 0$ ; $j \leftarrow 0$ ;
repeat
    $\Gamma(m) \leftarrow 0$ ; $\Delta(m) \leftarrow 0$ ; $\Pi(m) \leftarrow 0$ ;
    if $n \neq 0$ then
        Embed $w$ via CU size embedding technique ;
        foreach byte $(w_i)$ in $w$ do
            switch $\alpha_1$ of $w_i$ do
                case 1 : set as prediction modes embedding ;
                case 2 : set as CTU QP value embedding ;
                case 3 : set as non-zero DCT coeff. embedding ;
                otherwise : not embedding ;
            endsw
            if $\alpha_2 = 0$ then
                Skip $\alpha_3$ time(s) on selected embedding technique ;
            else
                $\alpha_4 = \alpha_4 + \alpha_3$ ;
            end
            Embed $\alpha_4$ using $\alpha_1$ technique;
        end
    end
    foreach CTU in $S_n$ do
        $m = m + 1$ ;
        foreach coefficient$(c_j)$ in m-th CTU do
            if $c_j \neq 0$ then
                $cnz(m) \leftarrow cnz(m) + 1$ ;
                $sav(m) \leftarrow sav(m) + |c_j|$ ;
                if $c_j > 0$ then
                    $s_{+,m} \leftarrow s_{+,m} + 1$ ;
                else
                    $s_{-,m} \leftarrow s_{-,m} + 1$ ;
                end
            end
        end
        foreach $4 \times 4$ pixels in m-th CTU do
            check CU sizes and add count on $\{F(\gamma_m)\}$ ;
            check CU depths and add count on $\{F(\delta_m)\}$ ;
            check CU modes and add count on $\{F(\pi_m)\}$ ;
        end
        $\Gamma_m^{\max_1}, \Gamma_m^{\max_2} \leftarrow$ max and second max of $\{F(\gamma_m)\}$;
        $\Delta_m^{\max_1}, \Delta_m^{\max_2} \leftarrow$ max and second max of $\{F(\delta_m)\}$;
        $\Gamma(m) \leftarrow |\Gamma_m^{\max_1} - \Gamma_m^{\max_2}|$ ;
        $\Delta(m) \leftarrow |\Delta_m^{\max_1} - \Delta_m^{\max_2}|$ ;
        $\Pi(m) \leftarrow |\Pi_{intra,m} - \Pi_{inter,m}|$ ;
        $s(m) \leftarrow |s_{+,m} - s_{-,m}|$ ;
    end
    $w \leftarrow H((\Gamma(m))_{m=1}^{M} || (\Delta(m))_{m=1}^{M} || (\Pi(m))_{m=1}^{M} ||$
    $(cnz(m))_{m=1}^{M} || (sav(m))_{m=1}^{M} || (s(m))_{m=1}^{M} || K)$ ;
    $m \leftarrow 0$ ;
    $n \leftarrow n + 1$ ;
until end of slices;
```

### 3.4. Tag verification

The embedded tag is verified during video decoding. Three layers of authentication are achieved in the proposed multi-layer authentication scheme, namely: the conveniently applicable layer without the need of the secret key (first layer); the dedicated layer to localize tampered region (second layer), and; the sophisticated layer which extracts the video features for computation of the hash value (third layer). Algorithm 2 shows the extraction and verification of tag during decoding. Here, $v_1, v_2$ and $v_3$ show the first, second and third layer authentication statuses, respectively. Status 1 indicates the video is authenticated, and status 0 indicates a failed authentication in that particular slice.

### 3.4.1. First layer of authentication

Recall that the tag is sequentially embedded by using the size of all CUs in each slice. This tag is repeatedly embedded until all CUs within the same video slice are exhausted. Therefore, the first layer of authentication checks for uniformity of the embedded tag throughout the slice under consideration. The embedded tag can be extracted during the decoding process by examining the CU size based on Fig. 4. The tags are extracted following the Z-scanning order in every CTU in a slice [22]. The first instance of the tag (32 bytes in length) is extracted and stored as $w^0$, while the following instances within the same slice are stored as $w^i$ for $i = 1, 2, \ldots$. If $\exists i$ such that $w_l^i \neq w_l^{i+1}$ at any bit location $l$, then the video is termed *tampered*. Specifically, the group of CTUs encoding the instance of the tag that differs from the majority are marked as the tampered group. On the other hand, when $w_l^i = w_l^{i+1}$ for all $i$ and all $l$, the video is authenticated with respect to the first layer.

### 3.4.2. Second layer authentication

Since the first layer of authentication depends only on the CU sizes, it is possible that some elements such as coefficients and QP are tampered, while maintaining the CU sizes. Therefore, the second layer is invoked to further verify the video at the byte-level of the extracted tag. Specifically, the last non-zero DCT coefficient in a CU, the CU prediction mode or the quantization parameter in the previous slice is considered, depending on the value $\alpha_1$ extracted from the tag. Then, $\alpha_2, \alpha_3$ and $\alpha_4$ are also obtained from the extracted byte segment. Next, the derived $\alpha_4$ from the previous slice is compared with the embedded $\alpha'_4$ based on the embedding technique as stipulated by $\alpha_1$. When $\alpha'_4 \neq \alpha_4$, it implies that tampering occurs at the region(s) under investigation. Note that when $\alpha_1 = 0$, no verification is performed because the tag is not embedded into any coefficient, QP or prediction mode, for that particular byte segment of the tag.

### 3.4.3. Third layer authentication

When a video passes the first and second layers of authentication, the video is merely verified to be neither modified nor tampered, but its source (i.e., sender) is not verified. To verify the video source, the same secret key $K$ (supplied during the encoding process) is required (see Section 3.1.2). Specifically, the values $\Gamma(m), \Delta(m), \Pi(m), cnz(m), sav(m)$ and $sm(m)$ in $S_n$ are computed to generate $w$ (see Eq. (2)). Next, the source of the video can be verified by comparing the generated tag $w$ against the embedded tag $w^i$. In case the tags match (i.e., $w = w^i$), the video is authenticated to be originating from a known (reliable) source, otherwise the source cannot be verified and hence the video cannot be trusted.

**Table 2**
Results of embedding tags in various classes of standard test video.

| Class (Test Video) | QP | Random access | | | | Low delay P | | | | Low delay B | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Original | | Processed | | Original | | Processed | | Original | | Processed | |
| | | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| Class A (PeopleOnStreet) | 8 | 50.5264 | 0.9999 | 50.3907 | 0.9999 | 51.7031 | 0.9999 | 51.6017 | 0.9999 | 51.6246 | 0.9999 | 51.5039 | 0.9999 |
| | 16 | 44.0813 | 0.9997 | 44.0174 | 0.9997 | 45.0852 | 0.9998 | 44.9942 | 0.9998 | 45.1223 | 0.9998 | 45.0277 | 0.9998 |
| | 24 | 38.9427 | 0.9985 | 38.8355 | 0.9985 | 39.3133 | 0.9986 | 39.2417 | 0.9986 | 39.4622 | 0.9987 | 39.3945 | 0.9987 |
| | 32 | 34.2105 | 0.9941 | 34.0721 | 0.9940 | 32.2255 | 0.9936 | 34.1066 | 0.9936 | 34.3092 | 0.9936 | 34.2095 | 0.9935 |
| | 40 | 29.7969 | 0.9799 | 29.6221 | 0.9795 | 29.7152 | 0.9786 | 29.5799 | 0.9786 | 29.7694 | 0.9783 | 29.6609 | 0.9783 |
| | 48 | 25.3706 | 0.9399 | 25.2184 | 0.9388 | 25.0522 | 0.9334 | 24.9315 | 0.9345 | 25.0255 | 0.9317 | 24.8976 | 0.9318 |
| Class B (Tennis) | 8 | 49.8782 | 0.9996 | 49.7813 | 0.9996 | 51.2741 | 0.9997 | 51.1930 | 0.9997 | 51.2695 | 0.9997 | 51.1816 | 0.9997 |
| | 16 | 44.2705 | 0.9979 | 44.2204 | 0.9978 | 44.5771 | 0.9982 | 44.5325 | 0.9982 | 44.6120 | 0.9982 | 44.5604 | 0.9982 |
| | 24 | 41.7317 | 0.9939 | 41.6404 | 0.9937 | 41.8567 | 0.9942 | 41.7730 | 0.9940 | 41.9911 | 0.9944 | 41.9232 | 0.9943 |
| | 32 | 38.2451 | 0.9799 | 38.1230 | 0.9791 | 38.3561 | 0.9804 | 38.2316 | 0.9798 | 38.5466 | 0.9813 | 38.4612 | 0.9808 |
| | 40 | 34.5212 | 0.9454 | 34.4460 | 0.9438 | 34.6043 | 0.9671 | 36.3681 | 0.9660 | 34.8057 | 0.9489 | 34.7805 | 0.9484 |
| | 48 | 30.3875 | 0.8689 | 30.4880 | 0.8705 | 30.5435 | 0.8707 | 30.6879 | 0.8720 | 30.6313 | 0.8733 | 30.8077 | 0.8766 |
| Class C (PartyScene) | 8 | 50.1873 | 0.9977 | 50.0968 | 0.9977 | 51.7037 | 0.9983 | 51.6155 | 0.9983 | 51.6403 | 0.9983 | 51.5349 | 0.9983 |
| | 16 | 42.4209 | 0.9925 | 42.3561 | 0.9924 | 43.2778 | 0.9929 | 43.2236 | 0.9928 | 43.2807 | 0.9930 | 43.2236 | 0.9930 |
| | 24 | 36.8693 | 0.9799 | 36.7610 | 0.9795 | 36.6196 | 0.9781 | 36.5374 | 0.9778 | 36.7842 | 0.9785 | 36.7004 | 0.9782 |
| | 32 | 31.6608 | 0.9495 | 31.4896 | 0.9483 | 31.0041 | 0.9428 | 30.9262 | 0.9420 | 31.0513 | 0.9431 | 30.9694 | 0.9422 |
| | 40 | 27.0375 | 0.8898 | 26.8283 | 0.8860 | 26.0257 | 0.8728 | 25.9443 | 0.8703 | 26.0159 | 0.8714 | 25.9227 | 0.8691 |
| | 48 | 22.9707 | 0.7619 | 22.8111 | 0.7549 | 22.2403 | 0.7402 | 22.2189 | 0.7385 | 22.1968 | 0.7371 | 22.2086 | 0.7364 |
| Class D (BasketballPass) | 8 | 50.3374 | 0.9954 | 50.2384 | 0.9952 | 50.8117 | 0.9956 | 50.7380 | 0.9955 | 50.9240 | 0.9958 | 50.8318 | 0.9957 |
| | 16 | 44.9885 | 0.9876 | 44.8839 | 0.9873 | 45.0952 | 0.9868 | 45.0297 | 0.9866 | 45.2152 | 0.9872 | 45.1418 | 0.9870 |
| | 24 | 39.1638 | 0.9629 | 39.0220 | 0.9616 | 39.2302 | 0.9600 | 39.1576 | 0.9594 | 39.3147 | 0.9604 | 39.2333 | 0.9597 |
| | 32 | 33.5886 | 0.8986 | 33.4273 | 0.8954 | 33.4390 | 0.8894 | 33.3368 | 0.8874 | 33.4943 | 0.8899 | 33.3970 | 0.8881 |
| | 40 | 29.2736 | 0.7943 | 29.1272 | 0.7889 | 28.9012 | 0.7785 | 28.8067 | 0.7765 | 28.9669 | 0.7804 | 28.8853 | 0.7781 |
| | 48 | 25.5622 | 0.6816 | 25.5205 | 0.6789 | 25.0662 | 0.6642 | 25.0752 | 0.6639 | 25.0087 | 0.6666 | 25.1259 | 0.6683 |
| Class E (FourPeople) | 8 | 50.0288 | 0.9994 | 49.9057 | 0.9994 | 51.2102 | 0.9996 | 51.1023 | 0.9996 | 51.1498 | 0.9996 | 51.0191 | 0.9996 |
| | 16 | 44.6663 | 0.9974 | 44.5989 | 0.9974 | 45.0461 | 0.9977 | 44.9736 | 0.9977 | 45.1308 | 0.9977 | 45.0451 | 0.9977 |
| | 24 | 41.9733 | 0.9956 | 41.8807 | 0.9955 | 41.5085 | 0.9951 | 41.4503 | 0.9951 | 41.6532 | 0.9951 | 41.5962 | 0.9951 |
| | 32 | 38.3121 | 0.9896 | 38.1473 | 0.9890 | 37.5055 | 0.9873 | 37.4642 | 0.9873 | 37.5704 | 0.9873 | 37.5439 | 0.9873 |
| | 40 | 33.6861 | 0.9679 | 33.4551 | 0.9655 | 32.7283 | 0.9597 | 32.7010 | 0.9599 | 32.7436 | 0.9594 | 32.7497 | 0.9596 |
| | 48 | 28.8121 | 0.8989 | 28.5426 | 0.8914 | 27.9325 | 0.8772 | 27.9864 | 0.8788 | 27.9577 | 0.8774 | 28.0315 | 0.8801 |
| Class F (ChinaSpeed) | 8 | 52.5190 | 0.9996 | 52.3205 | 0.9996 | 53.0841 | 0.9996 | 52.9650 | 0.9996 | 53.1060 | 0.9996 | 52.9716 | 0.9996 |
| | 16 | 47.1133 | 0.9983 | 46.9471 | 0.9983 | 47.5657 | 0.9986 | 47.4487 | 0.9985 | 47.6250 | 0.9986 | 47.4983 | 0.9985 |
| | 24 | 41.1222 | 0.9925 | 40.8611 | 0.9922 | 41.4770 | 0.9938 | 41.3542 | 0.9936 | 41.5332 | 0.9938 | 41.4090 | 0.9937 |
| | 32 | 35.0748 | 0.9717 | 34.7306 | 0.9708 | 35.3295 | 0.9736 | 35.1068 | 0.9729 | 35.3495 | 0.9739 | 35.1260 | 0.9732 |
| | 40 | 29.9592 | 0.9362 | 29.6215 | 0.9339 | 29.8973 | 0.9322 | 29.7294 | 0.9306 | 29.9264 | 0.9335 | 29.7613 | 0.9328 |
| | 48 | 25.6314 | 0.8690 | 25.1712 | 0.8625 | 25.1290 | 0.8478 | 25.1002 | 0.8481 | 25.1377 | 0.8509 | 25.1218 | 0.8524 |

**Algorithm 2.** Pseudo-code for tag verification, $w^i$

```
Input: $K$
Output: $v_1, v_2, v_3$
initialization ; $n \leftarrow 0, w \leftarrow 0, w^i \leftarrow 0$;
repeat
    if $n \neq 0$ then
        $w^0 \leftarrow 0$ ;
        foreach $x$ bytes of CU in $S_n$ do
            if $w^0 = 0$ then
                $w^0 \leftarrow x$ bytes of $w$ based on in Fig. 4 ;
            else
                $w^i \leftarrow x$ bytes of $w$ based on in Fig. 4 ;
                if $w^i = w^0$ then $v_1 \leftarrow 1$;
                else $v_1 \leftarrow 0$;
            end
        end
        foreach $w^i$ do
            check $\alpha_1$ in $w^i$ ;
            apply tag alteration based on $\alpha_2$,$\alpha_3$ in $w^i$ ;
            if $\alpha'_4 = \alpha_4$ then
                $v_2 \leftarrow 1$
            else $v_2 \leftarrow 0$;
        end
        if $w^i = w$ then $v_3 \leftarrow 1$;
        else $v_3 \leftarrow 0$;
    end
    Algorithm 1 step 21 - 47 to obtain $w$;
    $n \leftarrow n + 1$ ;
until end of slices;
```

## 4. Analysis

The HM10.0 reference software model is modified to implement the proposed multi-layer authentication scheme. Video in class A ($2560 \times 1600$), B ($1920 \times 1080$), C ($832 \times 480$), D ($416 \times 240$), E ($1280 \times 720$) and F ($1024 \times 768$) are utilized as our test video sequences. Three configurations, namely, RA (random access), LDP (low delay P) and LDB (low delay B), consisting of P/B slices are selected to collect results using QP in the range of $[8, 48]$. The results are recorded in Table 2.

### 4.1. Video quality

The results in Table 2 indicate that both the original and processed videos exhibit similar and steady growth in image quality when QP decreases. For video encoded with small QP (such as those in the range of $[8, 48]$), the degradation in quality with respect to SSIM (structural similarity) index [28] is hardly noticeable in all video classes. However, in terms of PSNR, the video quality drops, on average, <1 dB for QP in the range of $[8, 48]$ for all video classes considered.

To further examine the results, Fig. 11 shows the rate distortion curve for video sequences in Class A, B, C, D, E and F under RA, LDP and LDB configurations. Each graph is featured with a magnified region to show the detailed PSNR vs Bitrate performance between the original and processed video. In Class A, quality of the processed video drops $\sim$0.5 dB when considering the same bitrate (e.g., at 45 kbps, original and processed videos yield $\sim$41.5 and $\sim$41.0 dB, respectively). In other words, the processed video requires extra $\sim$5 kbps (e.g., at 41.5 dB, the original and processed bitrates are $\sim$40 and $\sim$45 kbps, respectively) to achieve the same quality as the original (compressed) video. Similar performances are observed in Class B (drop by $\sim$0.25 dB), Class C (drop by $\sim$1.5 dB), Class D (drop by $\sim$1.0 dB), Class E (drop by $\sim$0.25 dB) and Class F (drop by $\sim$1.2 dB).

Fig. 12 shows the SSIM results for all classes of video sequence under the RA, LDP and LDB configurations. Note that only the SSIM difference between the original and processed videos at lower bitrates (i.e., <20 kbps in Class A, <6 kbps in Class B, <50 kbps in Class C, <8 kbps in Class D, <5 kbps in Class E and <10 kbps in Class F) are reported here. It is because the processed video achieves similar perceptual quality (i.e., similar SSIM value) as that of the original video when encoding at higher bitrate, where the SSIM values exceed the boundary of the graphs shown in Fig. 12. Based on the numerical results recorded in Table 2, it is observed that the SSIM values of the original and processed videos for small QP values (e.g., QP < 16) are similar. This conclusion can also be drawn from Fig. 12 where the graphs of SSIM for both the original and processed videos converge as bitrate increases.

Overall, the quality of the processed videos degrade by <1% in terms of both SSIM and PSNR when compared to their original compressed counterparts. Perceptually, both original and processed videos appear to be identical by visual inspection. As a representative example, the 8-th slice from the original and processed video of Class D are shown in Fig. 10, which appear to be identical.

### 4.2. Robustness against forgery

In this section, the robustness of the proposed multi-layer authentication scheme is verified by considering the following attacks: slice tampering, CTU replacement, generic, VQ and slice re-compression attack.

#### 4.2.1. Slice tampering attack

During video transmission, video content are transmitted slice by slice. If any slice (e.g., $S_n$) is accidentally dropped or intentionally removed, then the following slice (i.e., $S_{n+1}$) will be authenticated in first and second layers, but not the third layer due to the dependency between two consecutive slices, where features from $S_n$ are required to generate the tag for verification in $S_{n+1}$.

For tampering across slices such as slice shuffling (reordering), insertion (see Fig. 13) and removal (see Fig. 14), the tampered slice can be detected due to the dependency between adjacent slices as part of our authentication design (see Section 8). For instance, if $n$ slices are removed and inserted at any other position, the positions of the removed slice as well as the starting and ending of the inserted slices can be detected. More precisely, as detailed in Section 8, the tag generated by using the features of $S_n$ is embedded in the subsequent slice $S_{n+1}$. Therefore, by checking the tag in the non-tampered slice immediately after the attacked slice, the act of removal or insertion of slice can be detected. For further investigation, we tamper the processed video by removing one video

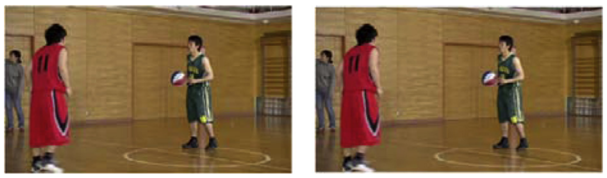
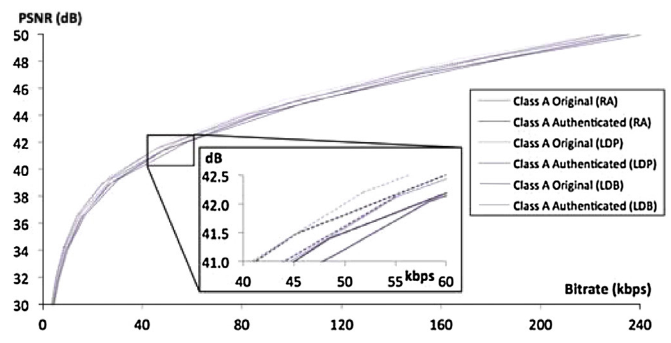
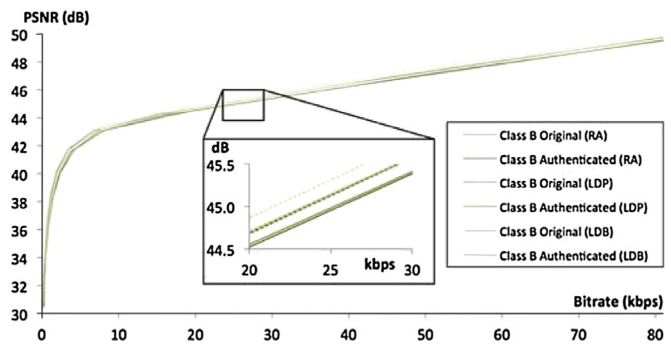


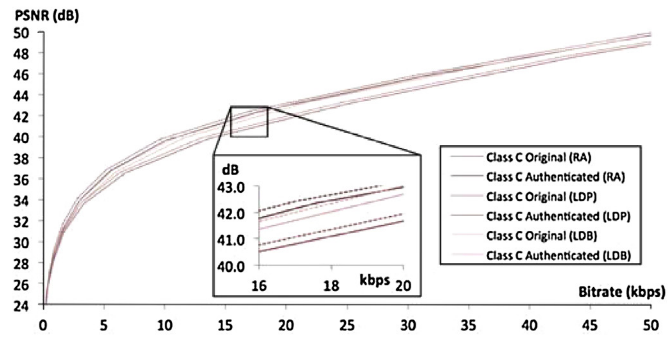
**Fig. 10.** Illustration of the 8th slice of the test video - *BasketballPass*.



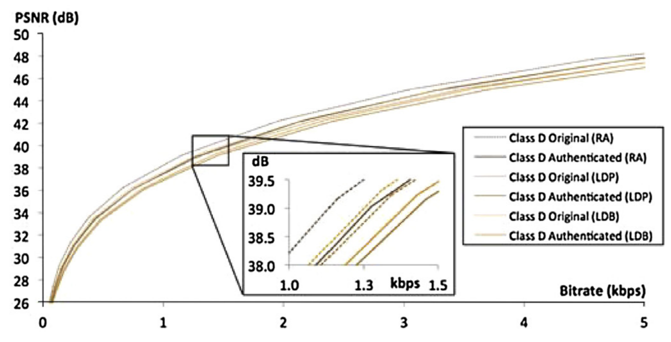
(a) Class A



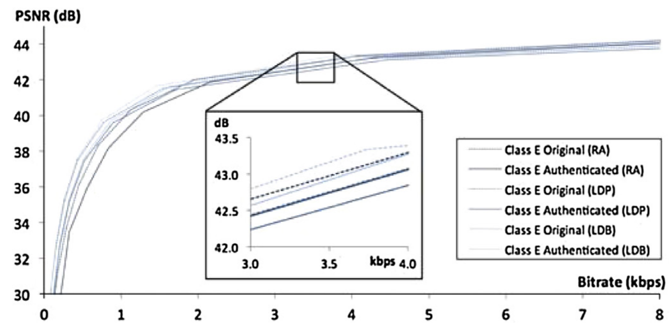
(b) Class B



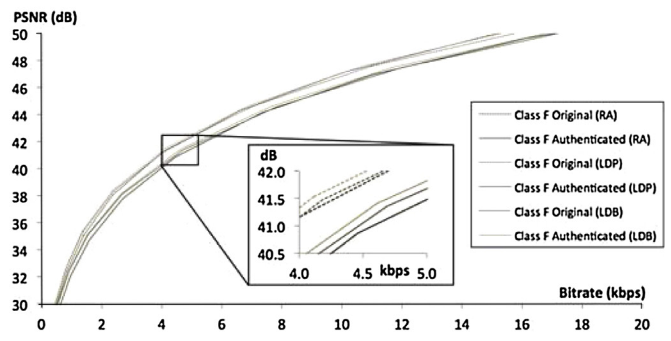
(c) Class C



(d) Class D



(e) Class E



(f) Class F

**Fig. 11.** PSNR vs Bitrate performance for original and processed video for various test video sequences.

slice. Then, we verify the authenticity of the manipulated video by checking the embedded tag. Fig. 15 shows the tag generated in the pervious slice (denoted by $w_g^i(n-1)$) as well as the tag extracted from the current slice (denoted by $w_e^i(n)$) in the processed video. The illustration shown in Fig. 15 is based on the video *BasketballPass* in Class D as the representative example. In the case of no tampering, the generated tag $w_g^i(n-1)$ in the previous slice $S_{n-1}$ is always identical to the extracted tag $w_e^i(n)$ from the current slice $S_n$ for $i = 1, 2, \ldots$ as illustrated in Fig. 15(a). Due to space limitation, only part of the tag for $i = 0$ (i.e., the first copy of the generated and extracted tags) are shown. In case of tampering, Fig. 15(b) illustrates an example of this attacking scenario, where the slice $S_{n+2}$ is removed. Since $w_g^i(n+1)$ completely disagrees with $w_e^i(n+2)$ (extracted from $S'_{n+2}$, which is $S_{n+3}$ of the original video), the act of tampering is detected in $S'_{n+2}$ of the video under consideration. The same verification process is viable regardless of the number of slices being copied, moved, inserted, or removed. Similar results are attained when other classes of processed video are tampered, and we omit the presentation here.

### *4.2.2. CTU replacement attack*

In this section, we consider the attacking scenario where the CU structure is tampered. Fig. 15(c) illustrates an example where $S_{n+2}$ is tampered by means of swapping the CU structures. Specifically, the CU structure of the 3rd CTU is swapped with the 4th CTU, which causes part of the extracted tag $w_e^i(n+2)$ to disagree with the generated tag $w_g^i(n+1)$, namely, 20, 50, ..., and *F*9 as illustrated in Fig. 15(c). Since the CU structure in $S_{n+2}$ is tampered, the tag $w_e^i(n+3)$ extracted from $S_{n+3}$ disagrees with the tag $w_g^i(n+2)$ generated in $S_{n+2}$. This mismatch will propagate due to the temporal dependency introduced in the design of the proposed authentication scheme. Note that this type of tampering can also be detected by checking the equality condition $w_e^i(n) = w_e^j(n)$, where any mismatch for any value of $i$ and $j$ renders the video as tampered. Similar results are attained when other classes of processed video are tampered, and we omit the presentation here.

On the other hand, by replacing one of the CTU contents by that of any other CTU with the same CU structure, the tampered slice will still be authenticated by the first layer. It is possible to change

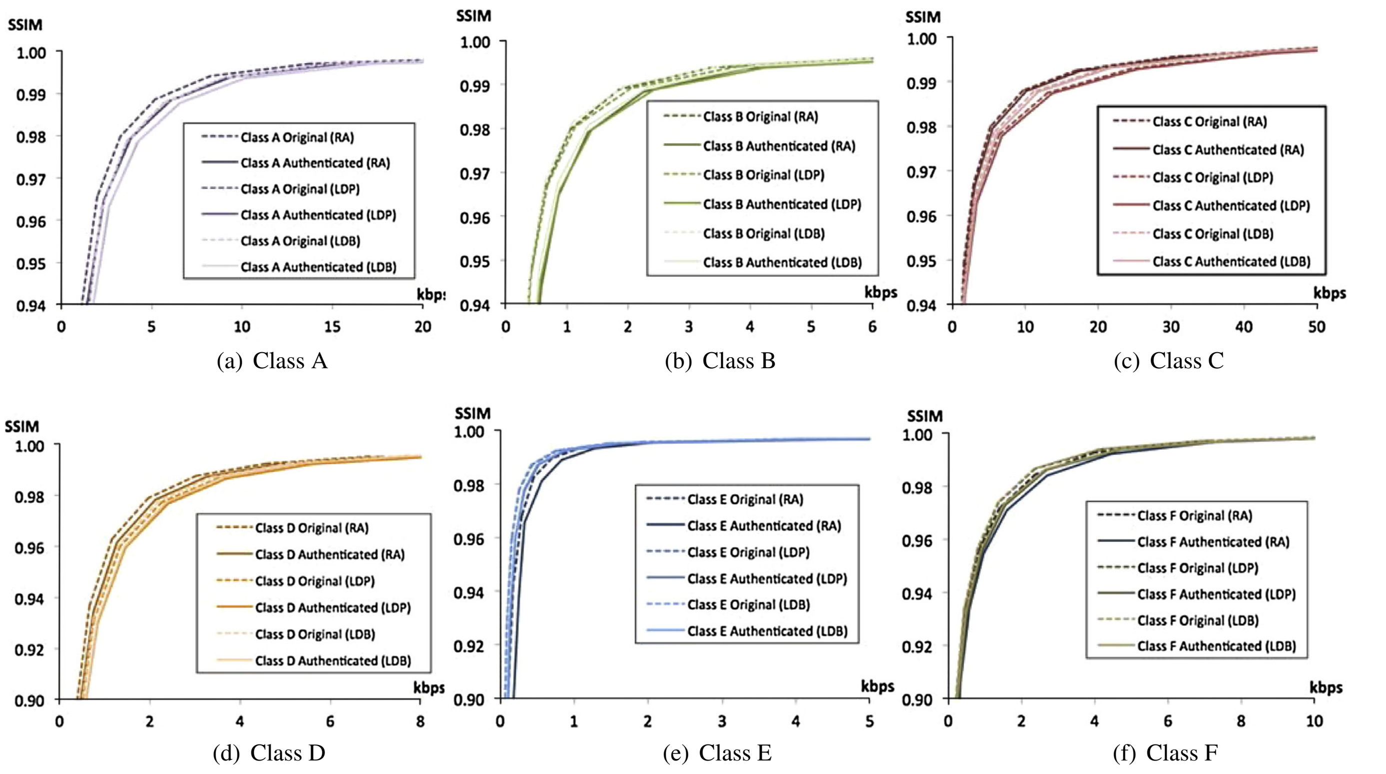


**Fig. 12.** SSIM vs Bitrate performance for the original and processed videos for various test video sequences.

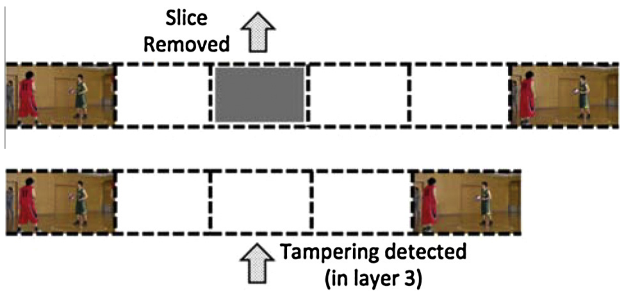


**Fig. 13.** Tampering detected when a slice is removed.

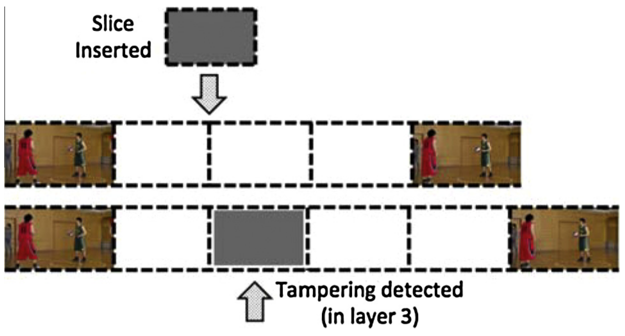


**Fig. 14.** Tampering detected when a slice is inserted.

the content of a CTU (e.g., coefficient value) while maintaining its size, which represents one bit of the embedded tag. However, in the second layer of authentication, the replaced CTU content will be examined by extracting the embedded information (viz., $\alpha'_4$) from the last non-zero coefficient, QP or prediction mode, depending on $\alpha_1$. If $\alpha'_4 \neq \alpha_4$, the mismatched CTU can be identified and utilized to pinpoint the modified CTU.

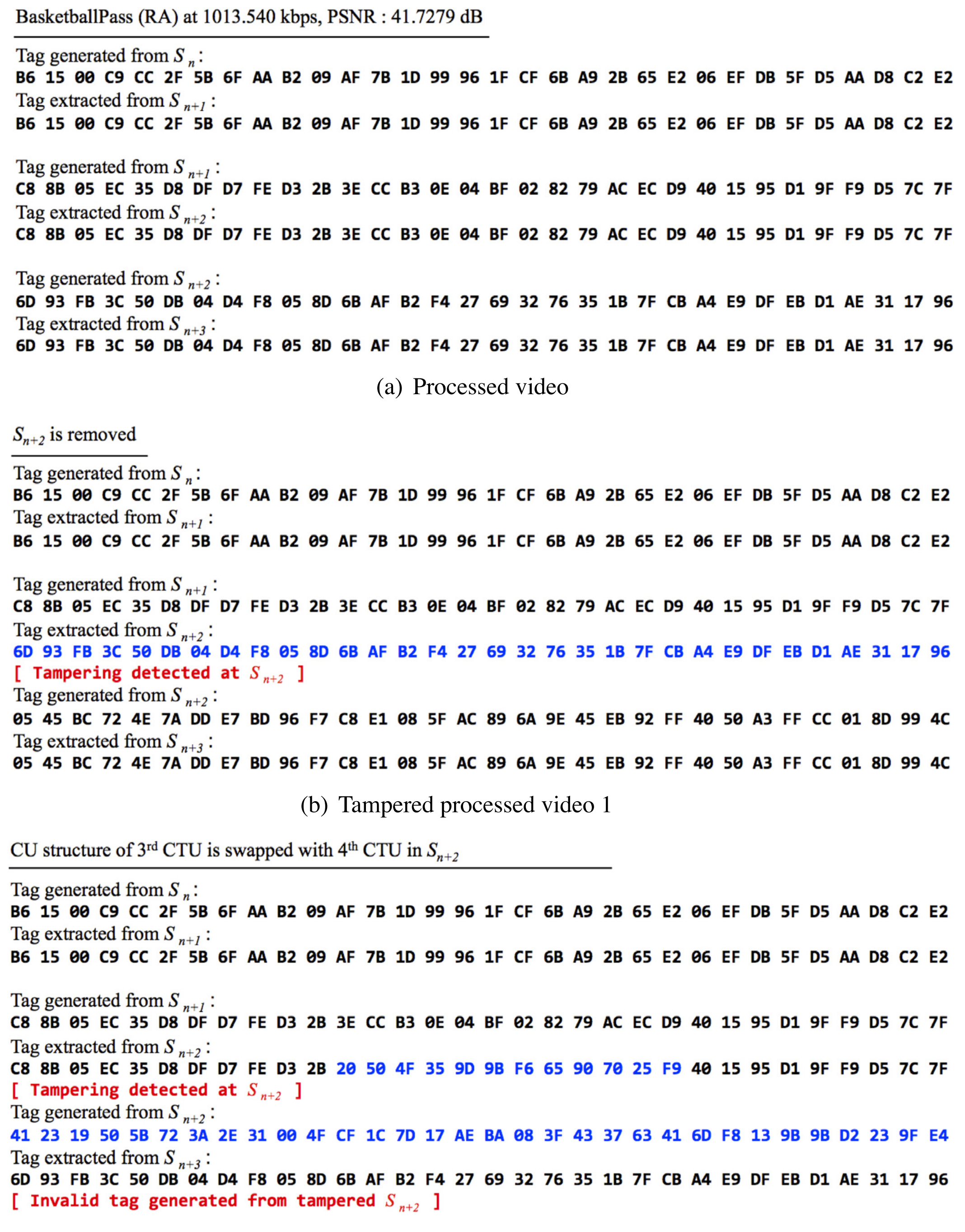
### 4.2.3. Generic attack

Lo et al. detailed a generic attack on tagged video stream by exploiting the coefficients of CU [29]. According to Lo et al., data embedding in LSB of coefficients, sign of coefficients and count of zero/non-zero coefficients are potentially attacked by changing the coefficients that are not involved in the authentication process so that the modified/tampered video will be authenticated at the decoder. However, this modification is infeasible under our proposed authentication scheme due to the unpredictable location of coefficients utilized for tag embedding. Recall that the tag is repeatedly embedded into selected nonzero coefficients, where nonzero coefficients are skipped in a non-regular manner as illustrated in Fig. 9 and Table 1. Hence, any discrepancies among multiple copies of the embedded information will be detected by the second layer of authentication in the proposed scheme (see Section 3.4.2).

In addition, due to the large number of possible combinations of CUs as well as other considered entities in HEVC video (including nonzero coefficients count and sign), although in theory other video content may have the same coarse features but it will be unlikely that these visually different contents (but producing the same authentication code) would be perceptually meaningful or having unnoticeable visual distortion. In other words, the distortion caused by tampering would be obvious to the naked eyes and the distortion may further propagate to other future slices due to motion compensation. Therefore, generic attack [29] is infeasible in attacking the proposed authentication scheme.

### 4.2.4. VQ attack

VQ (Vector Quantization) is a technique designed to retrieve the embedded information based on a constructed codebook obtained from a learning process using a huge quantity of authenticated



```
BasketballPass (RA) at 1013.540 kbps, PSNR : 41.7279 dB

Tag generated from S_n :
B6 15 00 C9 CC 2F 5B 6F AA B2 09 AF 7B 1D 99 96 1F CF 6B A9 2B 65 E2 06 EF DB 5F D5 AA D8 C2 E2
Tag extracted from S_n+1 :
B6 15 00 C9 CC 2F 5B 6F AA B2 09 AF 7B 1D 99 96 1F CF 6B A9 2B 65 E2 06 EF DB 5F D5 AA D8 C2 E2

Tag generated from S_n+1 :
C8 8B 05 EC 35 D8 DF D7 FE D3 2B 3E CC B3 0E 04 BF 02 82 79 AC EC D9 40 15 95 D1 9F F9 D5 7C 7F
Tag extracted from S_n+2 :
C8 8B 05 EC 35 D8 DF D7 FE D3 2B 3E CC B3 0E 04 BF 02 82 79 AC EC D9 40 15 95 D1 9F F9 D5 7C 7F

Tag generated from S_n+2 :
6D 93 FB 3C 50 DB 04 D4 F8 05 8D 6B AF B2 F4 27 69 32 76 35 1B 7F CB A4 E9 DF EB D1 AE 31 17 96
Tag extracted from S_n+3 :
6D 93 FB 3C 50 DB 04 D4 F8 05 8D 6B AF B2 F4 27 69 32 76 35 1B 7F CB A4 E9 DF EB D1 AE 31 17 96
```

(a) Processed video

```
S_n+2 is removed

Tag generated from S_n :
B6 15 00 C9 CC 2F 5B 6F AA B2 09 AF 7B 1D 99 96 1F CF 6B A9 2B 65 E2 06 EF DB 5F D5 AA D8 C2 E2
Tag extracted from S_n+1 :
B6 15 00 C9 CC 2F 5B 6F AA B2 09 AF 7B 1D 99 96 1F CF 6B A9 2B 65 E2 06 EF DB 5F D5 AA D8 C2 E2

Tag generated from S_n+1 :
C8 8B 05 EC 35 D8 DF D7 FE D3 2B 3E CC B3 0E 04 BF 02 82 79 AC EC D9 40 15 95 D1 9F F9 D5 7C 7F
Tag extracted from S_n+2 :
6D 93 FB 3C 50 DB 04 D4 F8 05 8D 6B AF B2 F4 27 69 32 76 35 1B 7F CB A4 E9 DF EB D1 AE 31 17 96
[ Tampering detected at S_n+2 ]
Tag generated from S_n+2 :
05 45 BC 72 4E 7A DD E7 BD 96 F7 C8 E1 08 5F AC 89 6A 9E 45 EB 92 FF 40 50 A3 FF CC 01 8D 99 4C
Tag extracted from S_n+3 :
05 45 BC 72 4E 7A DD E7 BD 96 F7 C8 E1 08 5F AC 89 6A 9E 45 EB 92 FF 40 50 A3 FF CC 01 8D 99 4C
```

(b) Tampered processed video 1

```
CU structure of 3rd CTU is swapped with 4th CTU in S_n+2

Tag generated from S_n :
B6 15 00 C9 CC 2F 5B 6F AA B2 09 AF 7B 1D 99 96 1F CF 6B A9 2B 65 E2 06 EF DB 5F D5 AA D8 C2 E2
Tag extracted from S_n+1 :
B6 15 00 C9 CC 2F 5B 6F AA B2 09 AF 7B 1D 99 96 1F CF 6B A9 2B 65 E2 06 EF DB 5F D5 AA D8 C2 E2

Tag generated from S_n+1 :
C8 8B 05 EC 35 D8 DF D7 FE D3 2B 3E CC B3 0E 04 BF 02 82 79 AC EC D9 40 15 95 D1 9F F9 D5 7C 7F
Tag extracted from S_n+2 :
C8 8B 05 EC 35 D8 DF D7 FE D3 2B 20 50 4F 35 9D 9B F6 65 90 70 25 F9 40 15 95 D1 9F F9 D5 7C 7F
[ Tampering detected at S_n+2 ]
Tag generated from S_n+2 :
41 23 19 50 5B 72 3A 2E 31 00 4F CF 1C 7D 17 AE BA 08 3F 43 37 63 41 6D F8 13 9B 9B D2 23 9F E4
Tag extracted from S_n+3 :
6D 93 FB 3C 50 DB 04 D4 F8 05 8D 6B AF B2 F4 27 69 32 76 35 1B 7F CB A4 E9 DF EB D1 AE 31 17 96
[ Invalid tag generated from tampered S_n+2 ]
```

(c) Tampered video 2

**Fig. 15.** Slice tampering detection based on tag verification.

videos with the same embedded tag. When a VQ style attack [30] is attempted, the proposed multi-layer authentication scheme is able to localize the attacked area. It is because our scheme requires the exact sequence of CU sizes to generate and match the tag $w^i$. Specifically, the tag $w^i$ of length 32 bytes may be copied from one part of the slice and pasted onto another part in the same slice (similar to copy-move attack in image forgery). Considering the HEVC encoding structure, the large number of possible combinations of CU sizes in any CTU (i.e., $> 2^{256}$) suggests that this attack is practically infeasible. In other words, it may be possible to fabricate a perceptually meaningless (i.e., noise-like) video to deceive the proposed multi-layer authentication scheme, but the fabricated video can be easily identified by visual inspection or non-reference image quality assessment [31]. Hence, the proposed scheme is robust against VQ style attack.

### 4.2.5. Slice re-compression attack

The proposed multi-layer authentication scheme is sensitive against modification. For instance, any modification in the video content (e.g., pixel values, DCT coefficients) causes $cnz(m), sav(m)$ and $s(m)$ to change, where the third layer authentication will fail (see Section 3.4) since the correct tag $w$ cannot be reproduced. During (re-) compression, the video encoder decides the CU structure of each video slice to achieve the desired bitrate. In the $n$th video

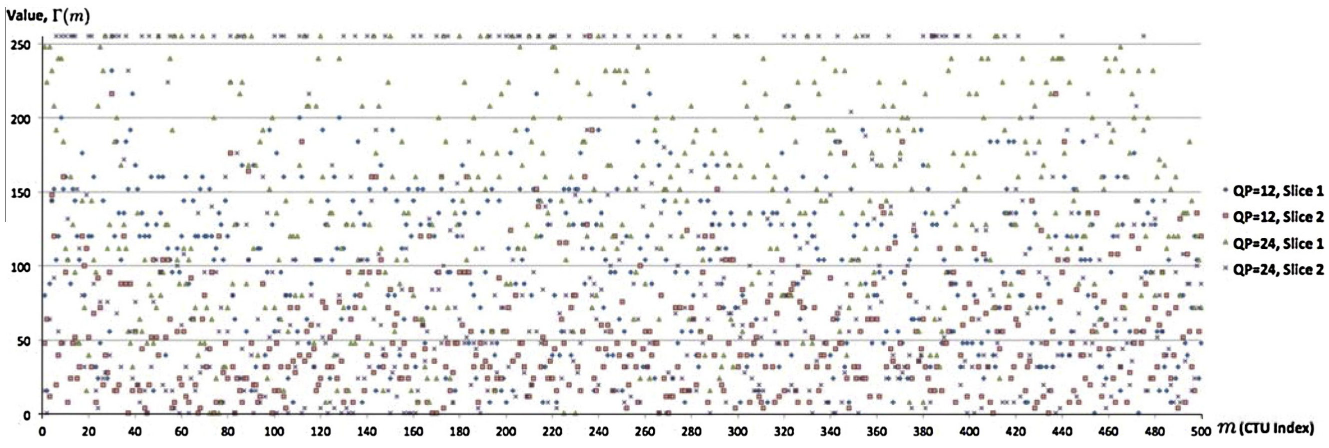


**Fig. 16.** Graph of $\Gamma(m)$ vs $m$ (CTU index) in slice 1 and 2 for Class B compressed with QP = 12 and 24.

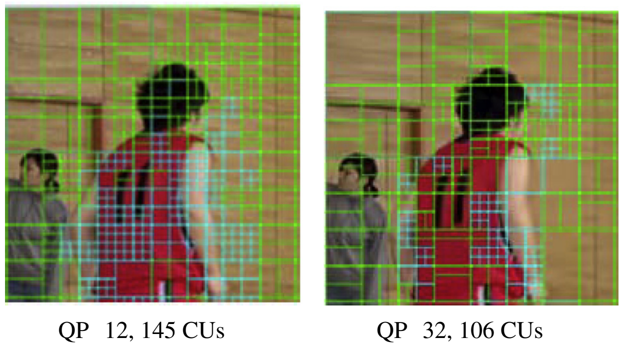


**Fig. 17.** Re-compression result of 4 CTUs with QP = 12 and QP = 32.

slice, $\gamma_m$, $\delta_m$ and $\delta_m$ are subjected to the CU structure decision when encoding at a targeted bitrate. The tag generated from $S_n$ is embedded into $S_{n+1}$, which causes the CU structure in $S_{n+1}$ (and hence the statistics) to change. Therefore, the embedded tags in the processed video are sensitive against re-compression (re-encoding) at different bitrates or different QP values. For example, Fig. 17 (a) and (b) shows the same slice compressed with QP = 12 and 32, respectively. It is apparent that the CTU sizes are different, and hence the same tag cannot be regenerated for authentication purpose. For further illustration, Fig. 16 shows $\Gamma(m)$, which represents the difference in number of occurrences for two most frequently occurring CU categories in each CTU. Here, $\Gamma(m)$ for video compressed with QP = 12 and 24 are shown. The $x$-axis is the CTU index throughout the entire test video sequence while the $y$-axis represents the value of $\Gamma(m)$. Results suggest that $\Gamma(m)$ for QP = 12 and 24 are significantly different, which confirm that the same tag cannot be regenerated. Furthermore, for slice re-sizing, cropping or rotation, the encoding process is required to generate the format compliant HEVC video. These modifications will further eradicate the embedded tags, hence failing the authentication process and hence indicating the sign of tampering.

### *4.3. Computational cost*

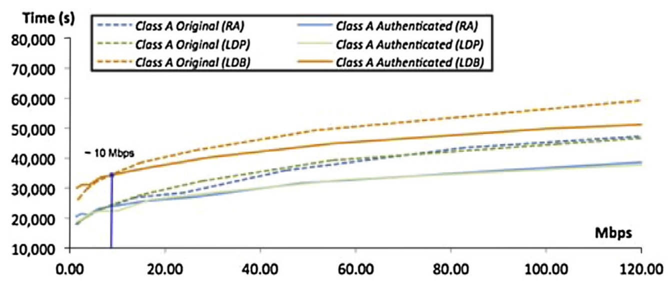
Fig. 18 shows the graphs of total time needed for the original and modified encoders versus bitrate for various classes of test video sequence. It is observed that the modified encoder requires lower computational time when encoding at higher bitrates (e.g., for LDB configuration, >10 Mbps in Class A, and >5 Mbps in Class B). In particular, by utilizing the proposed tag embedding technique [27], RDO of the modified encoder is restricted to choose one of the 4 types of CU to embed the authentication tag as described in Section 3.2.1, in contrast to the original encoder that considers all 8 cases. Note that the time spent on computing the cryptographic function is less than the time saved by restricting the choices of CU type due to tag embedding. However, the opposite situation is observed at lower bitrates (e.g., for LDB case, $<\sim 10$ Mbps in Class A, and $<\sim 5$ Mbps in Class B), where the modified encoder needs longer time when compared to the original video in all classes of video. It is because, at lower bitrate, the video sequence is encoded with CUs of larger sizes (e.g., mostly $32 \times 32$ and larger). In other words, the number of CU is reduced, and the chances to embed tag (i.e., time saving) are also reduced at lower bitrate.

Table 3 shows the percentage of increment in computational time, where the maximum and minimum increment among all considered QP values are recorded. To facilitate the presentation, let O denote the time needed to decode the original video using the original decoder, OA denote the time needed to decode the processed video (i.e., video with tag) using the original decoder, and A denote the time needed to decode and authenticate the processed video using the modified decoder. Positive percentage indicates an increment of computational time, and vice versa. Results for OA vs. O (i.e., Column 2 to 5) suggest that some of the test video sequences yield negative percentage of time increment. That is, the time needed to decode the processed video is shorter than that of the original video. Here, the decoding time is reduced due to the differences in the encoded CU structure in the original and processed videos. Specifically, this happens when a more complex CU structure is reduced to a simpler one due to tag embedding. One of the many possible scenarios is as follows: a CTU originally encoded with four $32 \times 32$ blocks is modified to be encoded by just one $64 \times 64$ block to embed the tag.

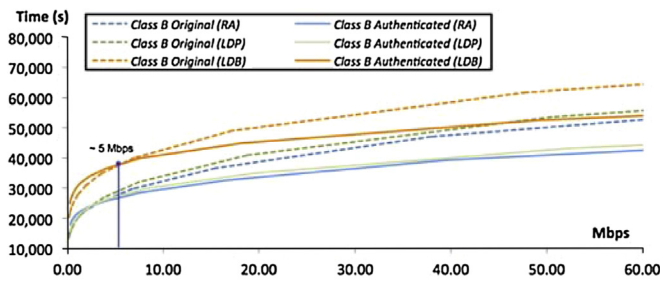
The negative percentages recorded in Column 6 to 9 of Table 3 suggest that the time needed for decoding and authenticating the processed video using the modified decoder is shorter than the time needed to decode the original video using the original decoder. In contrast, the opposite situations are captured by the positive percentages in Table 3. All in all, the proposed authentication scheme takes an additional computational time of −9.08% and +12.69% for decoding as well as authenticating the processed video in the best and worse scenarios, respectively.

### *4.4. Comparison*

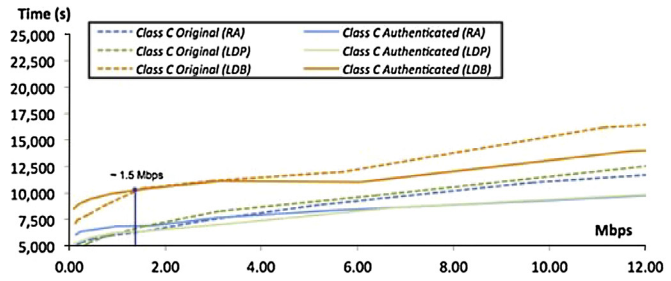
The conventional schemes may be applicable to all video coding standards (e.g., MPEG-2, H.264/AVC), but they are not specifically



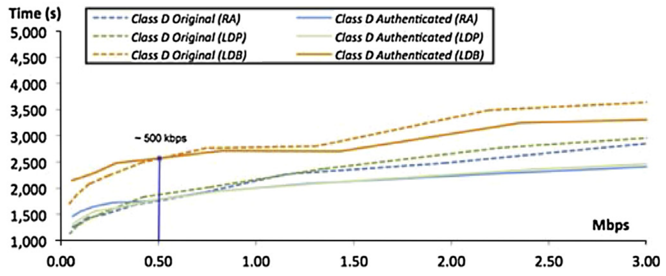
(a) Class A



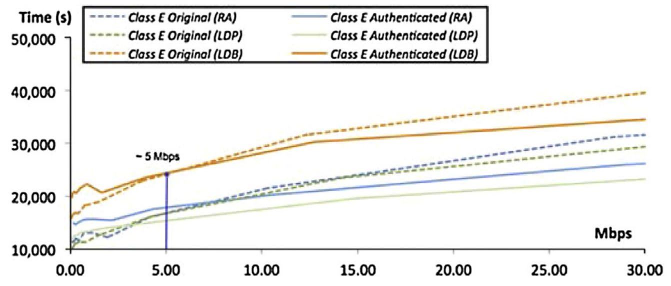
(b) Class B



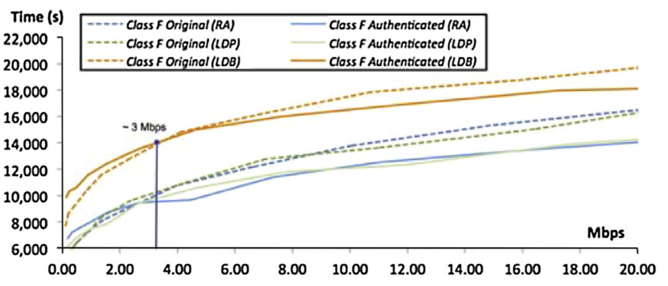
(c) Class C



(d) Class D



(e) Class E



(f) Class F

**Fig. 18.** Graphs of encoding time vs bitrate for original and processed video for various classes of test sequence.

**Table 3**
Percentage (%) of decoding time increment for original decoder and decoder with authentication.

| Video Class | OA vs. O | | | | A vs. O | | | |
|---|---|---|---|---|---|---|---|---|
| | LDP | | LDB | | LDP | | LDB | |
| | min. | max. | min. | max. | min. | max. | min. | max. |
| A | 1.44 | 6.08 | 1.55 | 2.21 | −4.27 | 4.82 | −3.20 | 4.27 |
| B | 1.06 | 7.73 | −0.60 | 4.51 | 0.03 | 10.77 | −1.30 | 9,23 |
| C | −1.30 | 3.44 | −1.48 | 6.20 | −6.72 | 2.65 | −5.92 | 6.71 |
| D | −2.39 | 9.27 | −4.17 | 5.11 | −6.23 | 6.74 | −7.88 | 5.11 |
| E | −6.18 | 4.24 | −3.67 | 4.95 | −9.08 | 5.05 | −6.91 | 12.69 |
| F | −4.65 | −1.57 | 0.46 | 5.14 | −8.04 | 3.18 | −3.65 | 10.91 |

O = Original decoder on original video.
OA = Original decoder on processed video.
A = Modified decoder on processed video.

implemented on or experimented with the HEVC coding standard, and hence it is not clear whether they are suitable for deployment on HEVC. To the best of our knowledge, there is no authentication scheme specifically designed to exploit/adapt to the coding structure of HEVC. As such, we compare our proposed multi-layer authentication scheme to four conventional schemes under different video standard. The first scheme is proposed by Roy et al. [13], where the authentication tags are embedded in the mid-frequency range of the non-zero DCT coefficients through hardware implementation. However, this process is only performed for I-frame under the H.264/AVC standard. The second scheme is proposed by Wei et al. [14], where tags are embedded into the Supplement Enhancement Information in Network Abstract Layer Unit for both the base- and enhancement-layers in SVC (Scalable Video Coding) of H.264. The third scheme is proposed by Upadhyay and Singh [32]. They utilize a non-linear classifier to compute the statistical local information (i.e., absolute difference) between every two consecutive slices and exploit this feature to determine whether a slice is tampered or genuine. The fourth scheme is proposed by Ren and Gorman [8], where the digital signature architecture is considered. Local video features were calculated from slices to form a concise fingerprint sequence, which is in turns appended to the video signal for authentication purpose.

Table 4 functionally compares the proposed and conventional authentication schemes for compressed video using a scale from 0 to 2. Here, “2” implies that the scheme is completely in line with the function, “1” indicates that the scheme achieves part of the function and “0” signifies that the scheme does not have the function. All schemes are robust to video slices dropping but only our proposed scheme exploits the dependency of all video slices in the temporal axis, i.e., a tampered content in current slice will be detected by the following slice. Also, our scheme extracts and utilizes the video features to verify the integrity of every slice without the need of the secret key *K*, which is only required to verify the origin of the video.

**Table 4**
Comparison among authentication scheme.

| Function | [8] | [13] | [14] | [32] | ✠ |
|---|---|---|---|---|---|
| Require feature extraction | 2 | 1 | 1 | 2 | 2 |
| Apply on all slices | 1 | 1 | 2 | 1 | 2 |
| Robust to slice dropping | 1 | 1 | 2 | 2 | 2 |
| Require key for verification | 2 | 2 | 0 | 2 | 1 |
| Localize tampered region | 1 | 0 | 0 | 1 | 1 |
| Source identification | 2 | 2 | 2 | 2 | 2 |
| Exploit temporal axis dependency | 0 | 1 | 0 | 1 | 2 |

2: fully functional, 1: partially functional, 0: no function.
✠: Proposed authentication scheme.

## 5. Conclusion

A multi-layer authentication scheme for HEVC compressed video is put forward. The temporal dependency was enforced and exploited, where authentication tag generated based on the statistics of the current slice was embedded into the subsequent slice. The video slices were verified by three layers of authentication: first layer provides an surface verification without utilizing the shared secret key; second layer localizes tampered region, if any, and; third layer verifies the source/sender by comparing the hash value of the combination of the shared secret key as well as the statistics from the video against the extracted tag. Results suggest that proposed multi-layer authentication scheme generates output video with high perceptual quality. Robustness of the proposed scheme against common attacks (e.g., CTU replacement, VQ attack) as well as its sensitivity against slice tampering and re-compression are analyzed and justified. The proposed scheme was also compared to the conventional video authentication schemes.

For further work, we aim to enhance our multi-layer authentication scheme for real-time applications such as live video streaming and video conferencing as well as handling the transcoding scenario.

## Acknowledgement

This research is supported by the Fundamental Research Grant Scheme (FRGS) MoHE Grant under project - Background/foreground separation in compressed domain (Grant No. FP063-2014A). R. Phan is supported by Telekom Malaysia (TM) under project - 2beAware (Grant No. RDTC/150879).

## References

[1] R. Waddilove, Best free video editing software 2015 uk <http://www.pcadvisor.co.uk/test-centre/photo-video/14-best-free-video-editing-software-2015-uk-3512375/> 2015 (accessed 1-Jan-2016).
[2] P.K. Atrey, A.E. Saddik, M. Kankanhalli, Digital Video Authentication, IGI Global, Hershey, United State of America, 2009.
[3] J. Maillard, Second emmy award for iso/iec mpeg-4 avc standard <http://www.iso.org/iso/home/news_index/news_archive/news.htm?refid=Ref1213> 2009 (accessed 1-Dec-2016).
[4] E. Casey (Ed.), Digital Evidence and Computer Crime: Forensic Sciences, Computers and The Internet, third ed., Elsevier Inc., United State of America, 2011.
[5] D.J.W. QPM, ACPO good practice guide for digital evidence, Tech. rep., Association of Chief Police Officers, England, Wales and Northern Ireland, 2012.
[6] B. Furht, D. Kirovski (Eds.), Multimedia Encryption and Authentication Techniques and Applications, Auerbach Publications, United State of America, 2006.
[7] C. Tartary, H. Wang, S. Ling, Authentication of digital streams, IEEE Trans. Inf. Theory 57 (9) (2011) 6285–6303.
[8] Y.J. Ren, L. O'Gorman, Accuracy of a high-level, loss-tolerant video fingerprint for surveillance authentication, in: Int. Conference on Pattern Recognition (ICPR), 2012, pp. 1088–1091.
[9] Y.J. Ren, L. O'Gorman, L.J. Wu, F. Chang, T.L. Wood, J.R. Zhang, Authenticating lossy surveillance video, IEEE Trans. Inf. Forensics Secur. 8 (10) (2013) 1678–1687.
[10] J. Baek, Y. ji Byon, E. Hableel, M. Al-Qutayri, An authentication framework for automatic dependent surveillance-broadcast based on online/offline identity-based signature, in: Int. Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 2013, pp. 358–363.
[11] J. Liu, J. Baek, J. Zhou, Y. Yang, J. Wong, Efficient online/offline identity-based signature for wireless sensor network, Int. J. Inform. Secur. 9 (4) (2010) 287–296.
[12] L. Song, J. Wang, L. Wang, J. Chen, A low-cost authenticating mechanism for interactive video streaming, in: Int. Conference on Software Engineering and Service Science, 2013, pp. 422–425.
[13] S. Roy, X. Li, Y. Shoshan, A. Fish, O. Yadid-Pecht, Hardware implementation of a digital watermarking system for video authentication, IEEE Trans. Circuits Syst. Video Technol. 23 (2) (2013) 289–301.
[14] Z. Wei, Y. Wu, R.H. Deng, X. Ding, A hybrid scheme for authenticating scalable video codestreams, IEEE Trans. Inf. Forensics Secur. 9 (4) (2014) 543–553.
[15] H.M. Al-Otum, Semi-fragile watermarking for grayscale image authentication and tamper detection based on an adjusted expanded-bit multiscale quantization-based technique, J. Vis. Commun. Image Represent. 25 (2014) 1064–1081.
[16] X. Qi, X. Xin, A singular-value-based semi-fragile watermarking scheme for image content authentication with tamper localization, J. Vis. Commun. Image Represent. 30 (2015) 312–327.
[17] I. Caciula, D. Coltuc, Improved control for low bit-rate reversible watermarking, in: IEEE Int. Conference on Acoust., Speech, and Signal Process. (ICASSP), 2014, pp. 7425–7429.
[18] A. Sharp, J. Devaney, A. Steiner, Digital video authentication with motion vector watermarking, in: 4th Int. Conference on Signal Process. and Commun. Syst. (ICSPCS), 2010, pp. 1–4.
[19] S. Ababneh, R. Ansari, A. Khokhar, Iterative compensation schemes for multimedia content authentication, J. Vis. Commun. Image Represent. 20 (2009) 303–311.
[20] J. Zhang, A.T.S. Ho, Efficient video authentication for H.264/AVC, First Int. Conference on Innovative Computing, Inform. and Control (ICICIC), vol. 3, 2006, pp. 46–49.
[21] Y. Tew, K. Wong, An overview of information hiding in H.264/AVC compressed video, IEEE Trans. Circuits Syst. Video Technol. 24 (2) (2014) 305–319.
[22] ISO, Information technology – high efficiency coding and media delivery in heterogeneous environments – part 2: High efficiency video coding* ISO/IEC 23008-2:2013, Int. Organization for Standardization, Geneva, Switzerland, 2013.
[23] ISO, Information technology – coding of audio-visual objects – Part 10: Advanced video coding* ISO/IEC 14496-10:2003, Int. Organization for Standardization, Geneva, Switzerland, 2003.
[24] G.J. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, Overview of the high efficiency video coding (HEVC) standard, IEEE Trans. Circuits Syst. Video Technol. 22 (12) (2012) 1649–1668.
[25] J. Vanne, M. Vitanen, T.D. Hamalainen, Efficient mode decision schemes for HEVC inter prediction, IEEE Trans. Circuits Syst. Video Technol. 24 (9) (2014) 1579–1593.
[26] NIST, Secure Hash Standard* FIPS PUB 180-2, National Institute of Standards and Technology, Gaithersburg, Maryland, United States, 2002.
[27] Y. Tew, K. Wong, Information hiding in HEVC standard using adaptive coding block size decision, in: 21th IEEE Int. Conference on Image Process. (ICIP), 2014, pp. 5502–5506.
[28] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
[29] S.-W. Lo, Z. Wei, X. Ding, R.H. Deng, Generic attacks on content-based video stream authentication, in: IEEE Int. Conference on Multimedia and Expo Workshops, 2014, pp. 1–6.
[30] J.-C. Chuang, Y.-C. Hu, An adaptive image authentication scheme for vector quantization compressed image, J. Vis. Commun. Image Represent. 22 (2011) 440–449.
[31] A.K. Moorthy, A.C. Bovik, Blind image quality assessment: from natural scene statistics to perceptual quality, IEEE Trans. Image Process. 20 (12) (2011) 3350–3364.
[32] S. Upadhyay, S.K. Singh, Learning based video authentication using statistical local information, in: Int. Conference on Image Inform. Process. (ICIIP), 2011, pp. 1–6.