Mitigation of H.264 and H.265 Video Compression for Reliable PRNU Estimation

Enes Altinisik, Kasim Tasdemir, and Husrev Taha Sencar

***Abstract*— The photo-response non-uniformity (PRNU) is a distinctive image sensor characteristic, and an imaging device inadvertently introduces its sensor's PRNU into all media it captures. Therefore, the PRNU can be regarded as a camera fingerprint and used for source attribution. The imaging pipeline in a camera, however, involves various processing steps that are detrimental to PRNU estimation. In the context of photographic images, these challenges are successfully addressed and the method for estimating a sensor's PRNU pattern is well established. However, various additional challenges related to generation of videos remain largely untackled. With this perspective, this work introduces methods to mitigate disruptive effects of widely deployed H.264 and H.265 video compression standards on PRNU estimation. Our approach involves an intervention in the decoding process to eliminate a filtering procedure applied at the decoder to reduce blockiness. It also utilizes decoding parameters to develop a weighting scheme and adjust the contribution of video frames at the macroblock level to PRNU estimation process. Results obtained on videos captured by 28 cameras show that our approach increases the PRNU matching metric up to more than five times over the conventional estimation method tailored for photos. Tests on a public dataset also verify that the proposed method improves the attribution performance by increasing the accuracy and allowing the use of smaller length videos to perform attribution.**

***Index Terms*— Photo-response non-uniformity (PRNU), video source attribution, H.264/H.265 encoding & decoding.**

I. Introduction

SOURCE attribution is a crucial task in digital forensics that deals with the problem of establishing reasonable certainty about the source of an evidence. This is a very challenging task as it not only rests on uniqueness of source objects, but also requires methods that can distinguish traces left by a particular source object from traces left by every other object. Among various kinds and sources of digital evidence, such unique characteristics are hard to find. Fortunately in digital imaging domain, this challenge has been successfully met by demonstrating that photo-response non-uniformity (PRNU) of an imaging sensor can be used for attribution purposes.

Manuscript received March 17, 2019; revised July 16, 2019 and September 20, 2019; accepted September 23, 2019. Date of publication October 2, 2019; date of current version January 16, 2020. This work was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant 116E273. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Luisa Verdoliva. *(Corresponding author: Husrev Taha Sencar.)*

E. Altinisik is with the TOBB University of Economics and Technology, 06560 Ankara, Turkey.

K. Tasdemir is with Abdullah Gül University, 38080 Kayseri, Turkey.

H. T. Sencar is with the Qatar Computing Research Institute, Hamad Bin Khalifa University (HBKU), Doha, Qatar, and also with the TOBB University of Economics and Technology, 06560 Ankara, Turkey (e-mail: htsencar@etu.edu.tr).

Digital Object Identifier 10.1109/TIFS.2019.2945190

Today, it is well established that this unique and stable noise-like pattern arising due to manufacturing variations in imaging sensors can be used to reliably identify an imaging sensor. In practice, this means that multimedia content captured by a digital camera, such as photos and videos, can be attributed to their source cameras. The development of this capability has so far been focused on photographic images with limited interest in videos. Given that videos are becoming increasingly prevalent on the Internet and digital forensic practitioners are just as likely to encounter videos as much as photos in user devices, it is very important that the same attribution capabilities are also available for videos.

In effect, the PRNU of a sensor superimposes a distinctive noise-like pattern that is modulated with the light intensity on to the sensor output image. Hence, the ability to attribute a media to its source essentially relies on correct estimation of this pattern. In any camera, however, raw image data captured by the sensor must be processed in steps through an imaging pipeline before a photo or video is created. Therefore, PRNU noise pattern of a sensor needs to be extracted from the camera output media which is crucially a processed and encoded version of the raw sensor output. These processing steps in a digital camera pipeline have important implications on how extraction should be performed and on the reliability of PRNU pattern matching. Not only they may cause significant estimation errors in the PRNU noise pattern, but they may also introduce artefacts that yield spurious similarities between distinct PRNU patterns. Although the steps involved in generation of a photo greatly overlap with those of a video, video generation steps are much more disruptive to PRNU noise pattern estimation.

When recording a video, a camera measures the visible light for a period of time and transforms it to a digital data stream. During this transformation, there are several stages that could suppress the PRNU pattern overlaid in each sensor image. At the image acquisition stage, an indispensable processing step is the downsizing of the full-frame sensor output. To reduce the amount of data that needs processing, cameras deploy downscaling, cropping, or pixel binning as mechanisms for resolution reduction. Another key processing step employed by all modern day cameras is the image stabilization which compensates for the effects of camera shake. This typically involves application of global and local geometric transformations to align consecutive images [1]. These are followed by processing steps, such as white-balancing, demosaicing, noise-reduction, and color transformation which collectively make up the imaging pipeline and are utilized during acquisition of

both photos and videos. Involved processing steps at this stage are expanded with more sophisticated capabilities as cameras' computational power increases.

The second stage is the video compression. A raw video is a sequence of images; therefore, its size is impractically large to store or transfer. Video compression exploits the fact that frames in a video sequence are highly correlated in time and aims at reducing the spatial and temporal redundancy so that as few bits as possible are used to represent the video sequence. Consequently, compression related information loss causes much more severe artefacts in videos as compared to those in photos. Currently, the most prevalent and popular video compression standards are H.264/AVC and its recent descendent H.265/HEVC.

In essence, successful estimation of the PRNU of an imaging sensor from a given video requires taking these additional processing steps into account. With this motivation, in this work, we focus on the impact of video compression on source camera identification and introduce how to deal with adverse effects of H.264 and H.265 compression standards. We note here that among all the processing steps in the video pipeline, electronic stabilization is the most detrimental to PRNU estimation. Nevertheless, there are still many cases where stabilization is not deployed when capturing a video. For example, smartphones running Android and iOS do not support stabilization in their front cameras. In addition, Android operating system supports video stabilization when the video resolution is under 1920×1080 pixels and does not guarantee its deployment if the frame rate is above 30 frames/second [2]. Similarly, iOS allows stabilization only at certain frame resolutions and frame rates depending on the device [3]. Regardless of whether stabilization is performed or not, the ability to effectively estimate PRNU from any video requires dealing with compression related artefacts as it is always performed.

The approach taken in this work to mitigate compression effects differs significantly from previous approaches in the literature. Essentially, prior work mainly tried to deal with this problem in post-processing either by suppressing compression artefacts [4], [5] or by selectively using video frames [6] or frame data [7], [8]. This overall led to solutions that are suboptimal in performance or that underutilize available video data when obtaining sensors' PRNU patterns. In contrast, our approach exploits the fact that the video decoder is at hand and incorporates available encoding information to PRNU estimation. The proposed solution deconstructs the problem into two parts. The first part addresses disruptive effects of a filtering function deployed by video codecs to suppress coding related visual artefacts. For this, we describe how one should intervene in the decoding process to minimize these effects rather than seeking to compensate them in post-processing. The second part of the problem concerns with compression related information loss. To cope with this, we introduce methods that utilize block-level encoding information by either masking out heavily compressed blocks or weighting PRNU contribution of each block in accordance with the amount of compression applied to each block.

In this work, we essentially build on ideas initially introduced in [7] to cope with video compression. More specifically, this work introduces a new method which utilizes a fine-grained weighting scheme instead of making a binary evaluation on each macroblock when incorporating them to PRNU estimation. The newly proposed method not only performs better, but it can also be used to estimate PRNU patterns from stabilized and highly compressed videos. This work also examines how different types of frames affect reliability of PRNU estimates and demonstrates that utilizing block level information yields better results than a crude categorization based on frame type. Moreover, to ensure broad applicability of the proposed methods, this work extends the basic capability built considering H.264 [9] encoding standard to H.265 coding standard [10], [11] as well. Finally, the effectiveness of the proposed method in PRNU estimation is validated using two separate datasets in terms of both improvement in the matching metric and the minimum number of frames needed to obtain a reliable estimate. The videos in the first dataset are acquired by 28 smartphone devices using a custom built camera application for Android smartphones [12] to capture videos under controlled settings (*e.g.,* stabilization, electronic zoom, frame resolution, and compression bitrate) to control for factors that contribute to weakening of PRNU pattern. The second dataset is the publicly available VISION dataset [13].

The remainder of this paper is organized as follows. In the next section, we describe the PRNU estimation process tailored mainly for photographic images and review the current approaches towards extending this capability to videos. Section III discusses critical steps of H.264 and H.265 video coding from a perspective of how they interfere with PRNU estimation. Section IV describes our approach and provides details of the proposed method in mitigating unwanted effects of video coding. The experiments conducted to validate the proposed method and our conclusions on the results are given in Sections V and VI, respectively.

II. PRNU Estimation From Still Images and Extension to Videos

The PRNU is essentially a systematic noise caused by the variation among pixels of an imaging sensor in their sensitivity to the light. Chen *et al.* [14] provided a mathematical model to characterize this intrinsic variability. According to this model, the raw sensor output can be expressed in matrix notation as

$$I = I^{(0)} + I^{(0)} \times K + \theta \tag{1}$$

where $I^{(0)}$ is the intensity of incident light on the sensor, K is a constant matrix encompassing pixel sensitivities with values distributed around 0, $I^{(0)} \times K$ is the componentwise multiplication corresponding to the PRNU, and θ is a combination of all other random noise components. Since K is the multiplicative factor that gives rise to PRNU overlaid to all sensor outputs, it also serves as a unique identifier for the sensor. The factor K is estimated through a maximum likelihood procedure using a set of images, $I_1, \ldots, I_N$, as

$$K = \frac{\sum_{i=1}^{N} I_i \times W_i}{\sum_{i=1}^{N} (I_i)^2} \tag{2}$$

where W_i represents the noise residue obtained after denoising image I_i to suppress image content's interference on the estimation.

To determine whether a given image I_q is captured by a sensor with an estimated PRNU factor K, a detection statistic based on normalized correlation of the noise residue W_q and the estimated PRNU $I_q \times K$ is used. It has been, however, empirically observed that sensors yield PRNU noise patterns of varying strength which in turn requires adjusting a detection statistic for each sensor. To address this problem, Goljan [15] introduced the use of peak-to-correlation energy (PCE) as a measure. The PCE essentially measures the percentage of the total power in the correlation plane that is concentrated in the correlation peak, and due to this normalization it yields a largely sensor-independent detection statistic.

This estimation and detection process obviously leaves out many details related to subsequent processing in the camera pipeline. In reality, the sensor output I undergoes several processing steps, such as color interpolation, filtering, color correction, and lossy compression, before it can be analyzed. Further the design and operation of the sensor at the hardware level may introduce additional complexities. The most critical aspect of these out-of-model factors relates to whether they introduce any systematic artefacts. Although any such structured pattern can be used to identify class-level characteristics of a camera, as shown in [16], [17], they will lead to false correlations when performing individual device level attribution. Therefore, in the presence of such artefacts, the estimated PRNU has to be further processed to eliminate them.

In the case of photos, the estimation process has been adapted to mitigate processing related artefacts in two main ways [14]. The first one concerns with the removal of biases introduced to PRNU estimate mainly by the demosaicing operation. The differences in offset gains applied to each color component at the sensor output and the periodic nature of color filter arrays together superimpose a periodic pattern onto interpolated color values. To eliminate this pattern, row and column averages of the estimated PRNU factor K are successively removed from each element of K. The second way involves removal of less deterministic but nevertheless structured artefacts such as the blockiness artefact caused by the JPEG compression To suppress this, Fourier domain representation of K is Wiener filtered, and the noise component that remains after the removal of all structural noise patterns in K is used as the sensor identifier.

As the sensor operation during acquisition of still photos and videos remains unchanged, the basic model in Eq. (1) is valid for both types of media. However the processing steps involved in a video pipeline are more complex and add further complications to reliable attribution of a video to its source. Therefore, the process for estimating the PRNU factor K has to be adapted to tackle additional artefacts related to video acquisition. Several approaches have been proposed to extend this capability to videos. However, reliable estimation of PRNU factor K from videos remained largely an unmet challenge.

The majority of the work in this domain has been concerned with video compression as it is typically more lossy than image compression, and hence more detrimental to PRNU estimation. The earliest work, [4], exploring this problem targeted artefacts, such as blocking and ringing noises, that are more apparent in video frames due to macroblock level operation of video coding. Assuming most codecs use 16×16 sized macroblocks, a frequency domain method is proposed for removing signal components that leak into the estimated K exhibiting this periodicity pattern. A significant limitation of this approach is that, when coding a video frame, codecs use adaptive macroblock sizes that start from 4×4 and go up to 64×64 while also supporting rectangular sized blocks. This versatility not only makes identifying a periodicity pattern difficult but the subsequent filtering step results with further weakening of the PRNU pattern. To better suppress compression artefacts, [5] proposed an alternative approach utilizing a minimum average correlation energy (MACE) filter which essentially minimizes the average correlation plane energy, thereby yielding higher PCE values. By applying MACE filter to the estimated PRNU factor K, 10% improvement in identification accuracy is obtained over the approach of [4].

In [18], the authors studied the effectiveness of the PRNU estimation and detection process on singly and multiply compressed videos. For this purpose, videos captured by webcams are compressed using a variety of codecs at varying resolutions and matched with the reference PRNU pattern estimated from raw videos. Their results highlight the dependency on the knowledge of encoding settings for successful attribution and the non-linear relation between quality and the detection statistic. Later, [6] explored the reliability of the use of different types of frames in PRNU estimation. They found that intra-coded frames (*i.e.*, I frames) yield better results as compared to predicted frames (*i.e.*, P and B frames) and suggested giving a higher weight to contribution of intracoded frames during estimation. This finding was essentially a consequence of the fact that I frames typically undergo a lighter compression than other types of frames.

To better deal with compression artefacts, [7] proposed the idea of modifying the operation of the video decoder and utilizing the macroblock level information. This represents a more active approach to suppressing artefacts than solely relying on post-estimation processing. Results of this work showed that turning-off the deblocking filter and masking out heavily compressed blocks promises significant improvement in the accuracy of source attribution. More recently, Kouokam and Dirik [8] proposed a similar masking approach based on the premise that lack of high frequency content in the prediction error of a block is a sign that compression removed the PRNU pattern. With their method, essentially, all blocks with all-zero frequency components (known as AC coefficients) in the prediction error are eliminated from PRNU estimation. This way of elimination of blocks has both positive and negative aspects. On the positive side, since for highly compressed blocks quantization will remove all the high-frequency content, the method works as a predictor for highly compressed blocks. On the negative side, even the well predicted blocks, which yielded no prediction error

(*i.e.,* those blocks where the prediction error was so small that quantization rounded it to zero), are eliminated from estimation.

Unlike the video compression aspect, only few works addressed the problem of source camera attribution for stabilized videos [19]–[21]. Assuming video frames have undergone some global affine transformation during electronic stabilization, these works perform a brute force search for the unknown parameters (*i.e.*, for scaling, rotation, and shift) that enable proper alignment of PRNU estimates of individual frames. In [19] and [21], authors proposed methods that essentially align PRNU patterns estimated from I frames with respect to a reference frame prior to estimating the PRNU factor K of the camera sensor. In contrast, [20] proposed first evaluating all pair-wise matches and identifying group of frames that are transformed similarly and then re-aligning the PRNU estimates from each group with respect to each other to obtain the factor K. These works and others pointed out that downsizing operation in camera pipeline needs to be taken into account when verifying the source of a video and considered coping with it through search and re-scaling. To address this gap, in [12], the downsizing behavior of more than 20 cameras is examined, and the effects of in-camera downsizing on accuracy of source attribution is quantified. This work also introduced a systematic approach for matching PRNU estimates obtained from media acquired at different resolutions. A similar approach was also taken by [22] with a focus on line-skipping and pixel-binning based downsizing methods towards the same goal.

In the next section, we provide an overview of H.264 and H.265 video coding standards with an emphasis on steps that are most disruptive to PRNU estimation before introducing elements of our method.

III. H.264 and H.265 Video Coding Standards

Video and image compressions are fundamentally different. A video is a sequence of highly correlated pictures. This redundancy is reduced by transferring an image region once, then letting the receiver construct other similar image regions using the received reference image and the prediction parameters.

Similar to JPEG image compression standard, H.264 and H.265 video compression standards operate by dividing an image or frame into smaller blocks. Each coding standard has different naming and size limits for those blocks. In H.264, the largest blocks are called macroblocks and they are of size 16×16 pixels. A macroblock can be further divided into smaller sub-macroblock regions as small as 4×4 pixels. (Figure 1 shows partitioning of a sample video frame in H.264 format.) H.265 format is an improved version of H.264 and it is designed with parallel processing in mind. The sizes of container blocks in H.265 can vary from 64×64 to 16×16. They might contain smaller sub-blocks as small as 4×4 pixels. Unlike the previous format, H.265 allows sub-blocks to have their own sub-blocks which results in a block tree. Moreover, there are several block types such as Coded Block (CB), Transfer Block (TB) and Prediction Block (PB). Block definitions and limitations in H.265 is more intricate than the previous version. However, those details are unrelated to PRNU estimation and skipped for the sake of brevity. Throughout this paper, all block types in H.265 are referred to as macroblocks even though the term macroblock does no longer exist in the H.265 standard.

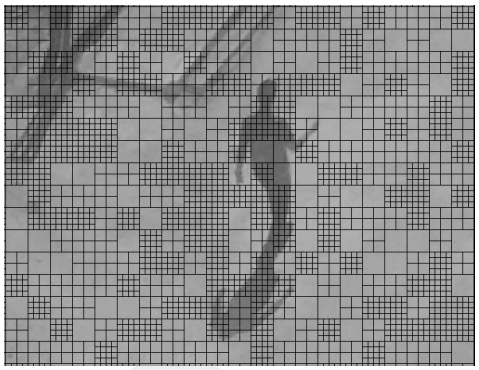

Fig. 1. A typical partitioning of an H.264 video frame obtained using Elecard software tool.

Blocks are compressed independently in a JPEG image[1] However, In H.264, a macroblock can be predicted from adjacent macroblocks of the same frame which makes the block an intra-coded block, or, it can be predicted from macroblocks of previous and/or future frames, which makes it an inter-coded block. The difference between the original block and its prediction is called residual. It is the residual that is being transformed, quantized, entropy coded and transferred alongside the prediction parameters. In this way, the receiver can construct the estimate of the original block using the reference block, residual and prediction parameters.

A frame in an H.264 or H.265 video can have three types: I, P, and B. An I frame is self contained and temporally independent. It can be likened to a JPEG image in this sense. However, they are quite different in many aspects including variable block sizes, use of prediction, integer transformation derived from a form of discrete sine transform (DST), variable quantization, and deployment of a loop filter. The macroblocks in P frame can use previous I or P frame macroblocks to find the best prediction. Encoding of a B frame provides the same flexibility. Moreover, future I and P frames can be utilized when predicting B frame macroblocks.

In a similar manner, macroblocks can be categorized into three types, namely, I, P, and B. An I macroblock is intra-coded with no dependence on future or previous frames, and I frames only contain this type of blocks. A P macroblock can be predicted from a previous P or I frames, and P frames can contain both types of blocks. Finally, a B macroblock can be predicted using one or two frames of I or P types. A B frame can contain all three types of blocks. If a block happens to have the same motion vectors as its preceding block and little or no residuals, this block is skipped since it can be constructed in the decoder side using the preceding block information. This type of blocks have inherent properties of its preceding

[1]Only exception to that is zero frequency coefficients (known as DC values) which are predicted from the previous block.

blocks and they are called skipped blocks. Skipped blocks might appear in P and B frames only.

A video is composed of a sequence of frames that exhibits a pattern, *e.g.*, IBBPBBPBB, known as group of pictures (GOP). This pattern starts with an I frame and typically repeats itself over the whole video. For the general case, however, the GOP size and structure can be either fixed or variable.

A. Quantization

In H.264, the transform and quantization steps are designed to minimize computational complexity so that it can be deployed by devices using limited-precision integer arithmetic. For this purpose, instead of using discrete cosine transform (DCT) as used in JPEG, H.264 uses an integer transform. Moreover, some parts of the integer transform are combined with quantization into a single step as follows. Integer transform uses a scaled integer approximation of a DCT transform matrix. However, unlike a DCT matrix, an integer transformation matrix is not orthogonal. For orthogonalization, it is multiplied by a scaling matrix. Later, the transformed matrix is divided by a quantization step. Instead of having a multiplication for scaling and then a division for quantization, H.264 standard offers merging these two operations into one multiplication step which can be realized in hardware very fast. The H.265 standard also uses the same quantization approach.

Because of this in H.264 standard the actual quantization step size cannot be selected by the user but rather controlled indirectly as a function of a quantization parameter. In practice, user decides on the bitrate of coded video and the quantization parameters are varied accordingly so that the intended rate can be achieved. Therefore, unlike a single quantization table as deployed in JPEG, the quantization parameters might change several times when coding a video. Also, in contrast to JPEG, a uniform quantizer step is applied to every coefficient in a 4×4 or 8×8 block by default. However, frequency dependent quantization can also be performed for different compression profiles.

B. Loop Filtering

Block-wise quantization performed during encoding yields a blockiness effect on the output images, H.264 decoder uses loop filters to compensate this effect it by applying a spatially variant low pass filter to smoothen the block boundaries. The filter is applied up to 3 pixels from the boundary of 4×4 blocks. The strength of the filter and the number of pixels affected from filtering depends on several constraints such as being at the boundary of a macroblock, the amount of quantization applied, and the gradient of image samples across the boundary.

Similar to H.264, an in-loop filtering process is applied in H.265. However, loop filtering in H.265 has two steps. In the first step, a deblocking filter (DBF), which is similar to the loop filter of H.264 format, is applied. The DBF is simplified in H.265 such as there are three Boundary Strength (BS) values instead of five. Also, it is only applied to the edges that are aligned on an 8×8 sample grid rather than 4×4 sample grid. In the second step, DBF is followed by a Sample Adaptive Offset (SAO) filtering which does not exist in H.264 standard. The purpose of SAO is reducing the ringing artefacts and applying an additional refinement to the reconstructed image. Applying SAO after DBF improves both the objective and subjective quality around edges and smooth areas even further [23].

It must be noted that the loop filter is also deployed during encoding to ensure synchronization between encoder and decoder. In essence, the decoder reconstructs each macroblock by adding a residual signal to a reference block identified by the encoder during prediction. At the decoder, however, all the previously reconstructed blocks would have been filtered. To not create such a discrepancy, the encoder also needs to perform prediction assuming filtered block data rather than using original macroblocks. The only exception to this is the intra-frame prediction where extrapolated neighboring pixels are taken from an unfiltered block.

IV. Mitigation of Video Coding Artefacts

The ability to effectively cope with video coding related information loss essentially requires an intervention on the decoding stage during which a compressed video file is turned into a sequence of frames. To realize this we take a two-pronged approach. First, we eliminate the loop filtering step at the decoder while ensuring the decoder stays in lockstep with the encoder. Second, we retain decoding information at the macroblock level, including macroblock sizes, locations, and quantization parameters, for each video frame and utilize it to perform more reliable estimation. Details of our approach are elaborated in the following subsections.

A. Compensation of Loop Filtering

The strength of blocking effect at block boundaries of a video frame is crucially determined by the level of compression. In high quality videos, such an artefact will be less apparent. Therefore, from a PRNU estimation point of view, presence of loop filtering will be of lesser concern. However for low quality videos, application of loop filtering will result with removal of a significant portion of PRNU pattern, thereby making source attribution less reliable [24].

A simplified decoding schema for H.264 and H.265 codecs is given in Fig. 2. In this model, the coded bit sequence is run through entropy decoding, dequantization, and inverse transformation steps before the residual block and motion vectors are obtained. Finally, video frames are reconstructed through the recursive motion compensation loop. The loop filtering is the last processing step before visual presentation of a frame. Despite the fact that loop filter is primarily related to decoder side, it is also deployed in encoder side to be coherent with the decoder. In the encoder side, decoded picture buffer is used for storing reference frames and those pictures are all loop filtered. Inter-prediction is carried out on loop filtered pictures. In contrast, unfiltered reconstructed macroblocks are used in intra-prediction.

Due to its weakening effect on the PRNU pattern, avoiding loop filtering will yield more reliable PRNU estimates. It can be seen in Fig. 2 that for intra-prediction, unfiltered blocks

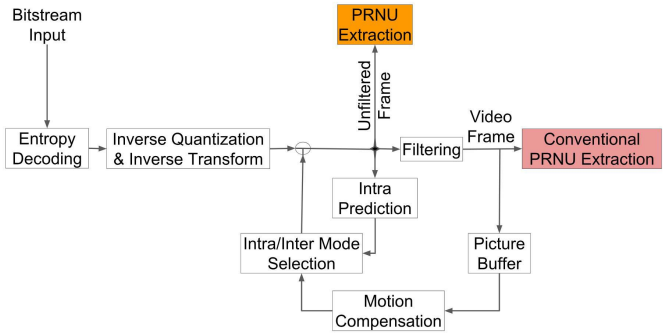

Fig. 2. Simplified schema of decoding for H.264 and H.265 codecs. In the block diagram, for H.264, the filtering block represents the deblocking loop filter and for H.265, it involves a similar deblocking filter cascaded with the sample adaptive offset filter.

are used. Since I type macroblocks that appear in all frames are coded using intra-frame prediction, bypassing the loop filter during decoding will not introduce any complications. However for P and B type macroblocks since encoder performs prediction assuming filtered blocks, simply removing the loop filter at the decoder will not yield a correct reconstruction. To compensate for this behavior that affects P and B frames, decoding process must be modified to reconstruct both filtered and non-filtered versions of each macroblock. Filtered macroblocks must be used for reconstructing future macroblocks, and non-filtered ones need to be used for PRNU estimation. To realize this, we modified the operation of H.264 and H.265 decoding modules of open source libraries developed under the FFMPEG project with required additional steps.

As a result of this modification, video frames will exhibit blocking artefacts resembling those introduced by JPEG coding with two significant differences: the size of blocks and locations of artefacts. In JPEG compressed images, blockiness artefact positions are well known because block sizes are constant over the image. In videos, by contrast, (sub-)macroblock sizes are not deterministic. Video coding standards leave encoding preferences including block sizes to the developers [11]. Also, block size decision involves a trade-off between picture quality and bitrate. Smaller block sizes produce finer pictures but cost more in bitrate. This decision is typically made through a rate-distortion optimization [10]. As a result, PRNU estimates containing blocking artefacts are very unlikely to exhibit a structure that is persistent across many consecutive frames. Therefore, we can rely on the estimation process, given in Eq. (2), itself to suppress blocking artefacts through averaging.

B. Coping With Quantization Related Information Loss

For both photos and videos, the most important factor that determines the reliability of the estimated PRNU factor is the loss of information caused by quantization. Measurements performed on diverse set of photos show that almost all camera models save their output as high quality JPEG coded images with a default quality factor in the range 85-95 [25]; hence, compression does not pose a significant obstacle to PRNU estimation in most cases. In contrast to photos, file size of a video is a major concern as it significantly increases the demand for storage and transmission resources. Therefore, cameras typically downsize high-resolution sensor output, further degrading the PRNU pattern [12], and perform a heavier compression.

To provide a relative comparison between effects of video and still image compression on PRNU estimation, we performed a test. With JPEG coding, the strength of compression is determined by the quality factor (QF), a number that varies between 1 and 100 such that at a quality of 100 no quantization (other than integer rounding) is performed and higher quantization levels are associated with lower QF values. Similarly, in video coding, the extent of compression is determined in terms of the quantization step size. In both H.264 and H.265 codecs, the value of quantization step size, $Qstep$, is indexed through a quantization parameter QP. The parameter QP takes values between 1 and 51 with lower values indicating that lesser amount of quantization noise is introduced to each transform coefficient. (The relation between the two parameters is such that $Qstep$ is 1 when QP is set to 4, and for each increment of 6 in QP, $Qstep$ value doubles.)

Tests are performed on a set of high resolution photos (4160×3120) acquired at a JPEG quality of 100 while slightly moving the camera in one direction, thereby simulating video acquisition. A reference PRNU pattern is estimated from 150 images and 120 images are set aside for comparison. This subset of images are first compressed at all quality values, yielding a total of 99×120 images. The same images are then H.264 encoded (without turning off the loop filter) at fixed QP values which resulted with 51 compressed videos. The mean squared error (MSE) between the original and compressed images are computed and averaged over all QF values. The same computation is performed for frames extracted from the videos to determine the average MSE corresponding to all QP values. Then, QP and QF values that yield similar MSE values are identified. The same process is repeated by first estimating the PRNU patterns from all resulting images and frames and then evaluating their match with the reference PRNU pattern in terms of the PCE metric. Average PCE values at all QP and QF values are subsequently computed and compared.

Table I identifies QP and QF pairs that yield similar MSE and PCE values. It can be seen that H.264 coding, in comparison to JPEG coding, maintains a relatively high image quality (measured in the MSE sense), despite rapidly weakening the inherent PRNU pattern. For example, compression with $QP = 21$ introduces an MSE distortion that is comparable to JPEG compression at $QF = 89$; however, PCE values match those of a higher compression, $QF = 66$. It is observed from these results that what would be considered medium level in video compression ($QP = 20$ to $QP = 30$) is equivalent to heavy JPEG compression ($QF = 66$ to $QF = 32$). This increasing gap essentially indicates why the conventional method of PRNU estimation is not very effective on videos.

In many video coding scenarios, however, user decides on the bitrate of coded video and the encoder changes the value of

TABLE I
A COMPARATIVE EVALUAITON OF JPEG AND H.264 ENCODING IN TERMS OF MSE AND PCE

Equalized MSE		Equalized PCE	
QF_{JPEG}	$QP_{H.264}$	QF_{JPEG}	$QP_{H.264}$
92	5	92	5
92	10	90	10
91	15	84	15
90	18	77	18
89	21	66	21
87	24	50	24
82	27	33	27
77	30	22	30
67	35	10	35
61	40	3	40
57	45	1	45
57	50	1	50

QP adaptively to attain the target bitrate. Further, unlike JPEG compression where a single quantization table is deployed for compression of all 8×8 image blocks, QP parameter might change for each block when coding a frame. As a result, the effect of compression on the reliability of the estimated PRNU pattern depends on the video content itself and the designated level of compression, and devising a parametric relation between the two is a challenging task.

The effect of compression on PRNU estimation can be empirically determined in terms of the quantization parameter QP and the PCE metric. Due to the exponential relation between QP and $Qstep$, a linear increase in QP will cause a similar degradation in quality measured in terms of peak signal-to-noise ratio. Hence, it can be expected that the strength of estimated PRNU will also decrease exponentially with increasing QP. To test the validity of this assumption, we performed tests on a set of videos captured by 28 different smartphone cameras. These videos were captured under controlled settings by turning off stabilization and electronic zoom and at the highest frame resolution supported by the camera with the best video bitrate possible (corresponding to a QP value of 1) using the camera app developed in [12]. All the test videos include indoor scenes captured under natural light by moving the cameras.

Using two videos from each camera, we performed the following tasks: *(i)* camera sensor's reference PRNU pattern is obtained using one of the videos; *(ii)* the other video is compressed using H.264 codec at all QP values, yielding a total of 51 compressed versions of the same video; *(iii)* resulting videos are decoded and individual PRNU patterns are estimated from all frames; *(iv)* extracted PRNU patterns are matched with the reference pattern and PCE values are computed for all QP values; and finally, *(v)* within camera PCE values are normalized with respect to the PCE value at $QP = 15$ to further reduce the variation among cameras. It must be noted that by fixing the QP value, the encoder is essentially forced to encode frames utilizing its choice of macroblocks (I, P, and, B) quantized with the designated QP value. However, the encoder can selectively skip the encoding of some macroblocks. In such cases, we use the block

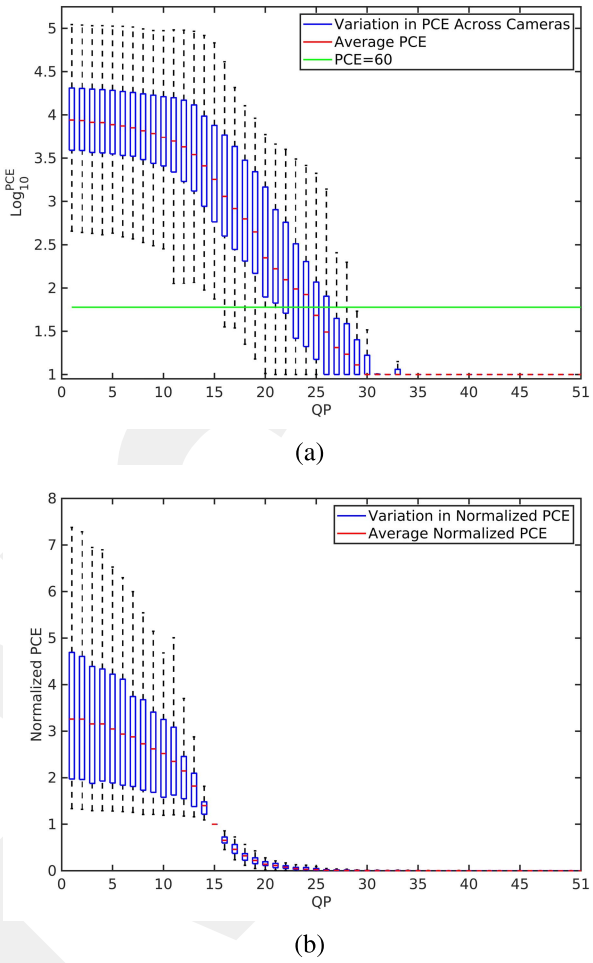

Fig. 3. Change in PCE values with respect to QP computed using 28 videos captured by different cameras: (a) PCE values plotted in logarithmic scale in comparison to the commonly accepted threshold of PCE = 60 and (b) PCE values normalized with respect to the PCE value at $QP = 15$ for more compact representation of inter-camera variation.

that substitute the skipped macroblock for estimation rather than discarding it. Since skipped-blocks are decided based on a rate-distortion criterion, the substituting block will not introduce an arbitrarily large distortion and, as shown in [26], even correctly estimating signs of PRNU components will still facilitate reliable attribution with only a slight reduction in the accuracy. Moreover, at lower bitrates blocks are more likely to be skipped; therefore, discarding those blocks will eliminate most of frames' content and attribution will not be possible.

Figures 3-a and 3-b display the change in PCE with respect to QP obtained at step *(iv)* and step *(v)* of the above procedure by combining results from 28 cameras in the box-plot format. It can be seen that PCE values show a slow, almost linear, initial decline until QP is around 10 followed by an exponential drop for increasing values with PRNU pattern getting almost completely removed when QP goes beyond 28. We consider two strategies to cope with this behavior.

1) PRNU Masking: Parts of PRNU pattern estimated from macroblocks that are compressed at higher QP values will only yield noise. Therefore, those macroblocks should be excluded from the estimation process. To incorporate this

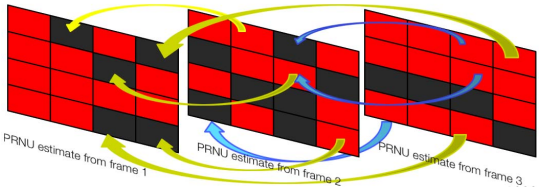

Fig. 4. Rearrangement of remaining macroblocks after binary masking of each frame. Black colored blocks represent the locations of masked out macroblocks and arrows show the order of splicing depending on factors such as QF, intensity, texture, etc.

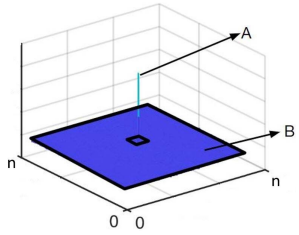

Fig. 5. PCE is calculated as the square of the ratio of cross-correlation term at A to the average of square terms in B *i.e.*, $PCE = \frac{A^2}{(avg(B^2))}$.

into the estimation process we create a pixel-wise mask, M, for each frame as it is being decoded. In the mask M, all pixel locations corresponding to macroblocks that underwent compression with QP higher than 28 are set to 0, and the rest of the pixel values are set to 1. The PRNU factor K is estimated using Eq. (2) by multiplying each contributing frame by this mask as

$$K = \frac{\sum_{i=1}^{N} I_i \times W_i \times M_i}{\sum_{i=1}^{N} (I_i)^2 \times M_i} \tag{3}$$

The use of a binary mask to eliminate highly compressed blocks can be further enhanced by incorporating macroblock level compression information. Since a lower QP value corresponds to a more reliable PRNU estimate, new frames can be spliced together from PRNU noise blocks obtained from macroblocks that have lower QP values [7]. In other words, to maximize PCE, we can fill up new frames by select PRNU noise blocks from other frames as shown in Fig. 4. It must be noted that in the newly generated frames, it is necessary for macroblocks to preserve their position in the original frames to prevent any geometric distortion. The rearrangement of macroblocks can be performed to take also into account content characteristics, such as intensity levels and texture, that are known to affect PRNU estimation. Resulting spliced-up frames can then either be used to match with the reference PRNU pattern or to create more reliable PRNU estimates.

The PRNU masking based frame splicing method has two shortcomings. First, for low quality videos, it ends up eliminating most blocks, thereby leaving only a few blocks for PRNU estimation. Further, it does not make a distinction between an uncompressed or moderately compressed block during estimation. Second, it cannot be applied to stabilized videos where frame-to-frame synchronization between PRNU patterns is distorted prior to video coding.

2) PRNU Weighting: An alternative approach to binary masking is to weight each macroblock depending on the level of compression so that blocks that underwent lighter compression contributes more strongly to estimating the PRNU pattern. This approach essentially corresponds to assigning each macroblock a continuous valued weight as opposed to a binary weight used by the PRNU masking approach. Therefore, it is expected to yield more reliable PRNU estimates. Further, due to the ability to use all macroblock data, PRNU weighting is better suited to performing attribution on stabilized videos.

Having empirically determined the relation between QP and PCE as shown in Fig. 3, we can utilize it to determine appropriate weighting factors. For the general case, given two PRNU factor estimates K and K^*, the PCE is defined as the ratio between the square of correlation between K and K^* and the total energy in the cross-correlation plane except around a spot around the peak correlation, and it is calculated as:

$$PCE(K, K^*) = \frac{c^2(0,0)}{\frac{1}{n^2 - \|\delta\|} \sum_{i,j,(i,j)\notin\delta} c^2(k,l)}, \quad \text{where} \tag{4}$$

$$c(k,l) = \frac{1}{n^2} \sum_{i=1,j=1}^{n,n} K_{i,j} \cdot K^*_{i\oplus k, j\oplus l}, \; k, \; j = 0, 1, \ldots n-1 \tag{5}$$

and $\oplus$ denotes a modulo n addition to compute cyclically shifted cross-correlation within a window of size $n \times n$ except for a δ region around the correlation peak. The terms involved in computation of PCE are depicted pictorially in Fig. 5.

When the PRNU estimates K and K^* correspond to the same sensor, the numerator in Eq. (4) can be interpreted as the signal, and the denominator as the noise part since PRNU pattern is known to be pixel-wise independent. Hence, given two PRNU instances K^x and K^y estimated from different media captured by a camera with reference pattern K^*, the ratio r of two PCE values, *i.e.*, $r = \frac{PCE(K^x, K^*)}{PCE(K^y, K^*)}$, will be determined by the numerator of Eq. (4) as the denominator term depends on average correlation of two uncorrelated signals, which will be the same for both K^x and K^y. Correspondingly,

$$\sqrt{r} = \frac{c_x(0,0)}{c_y(0,0)} \tag{6}$$

where $c_i(0,0)$ corresponds to correlation between K^i and K^* as defined in Eq. (5). Assuming a linear relation between K^x and K^y, this yields $K^x \approx \sqrt{r} K^y$. Although PRNU patterns estimated from media at multiple compression levels will not be linearly related, this model can nevertheless be used to model the change in PCE with respect to the compression level. This also means that given the PCE-QP relation, the relative weighting factors for each QP level can be determined through the square-root of the underlying PCE-QP curve normalized with respect to a base PCE value. Hence, the estimation process can be changed by modifying the binary values in mask M introduced in Eq. (3) by the weighting factors corresponding to QP of each macroblock. When such a curve is not available for a camera, the general characteristic given in Fig. 3 can be utilized for this purpose. Figure 6 displays the weight function corresponding to average

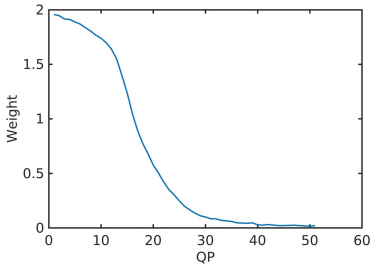

Fig. 6. The weight curve as a function of QP obtained through averaging PCE-QP curves of 28 cameras given in Fig. 3.

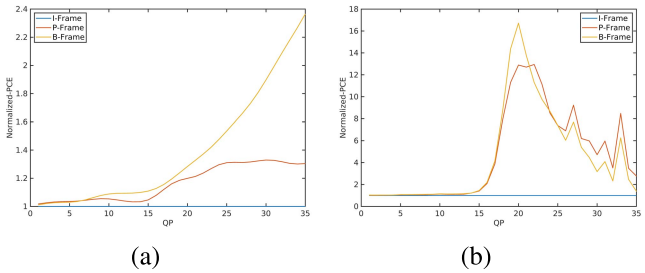

Fig. 7. PCE-QP curves for three picture types obtained from videos encoded using a fixed GOP structure of type (a) IBP and (b) IBBPBBPBBPBBPBBPBBPBBPBBP (GOP size 25). PCE values are normalized with respect to mean PCE values of I frames at each compression level.

PCE-QP curves of 28 cameras. Accordingly, macroblocks that underwent compression at $QP = 10$ and $QP = 25$ will be weighted by factors of 1.74 and 0, 25, respectively, indicating that the former macroblocks will contribute 7 times more strongly to estimation process than the latter ones. It must be noted that the general PCE-QP characteristic is obtained under H.264 encoding. The H.264 and H.265 coding standards perform quantization in the same manner. Hence, the inflicted information loss due to compression will be similar, and the same characteristic is expected to apply to both compression methods.

C. Choice of Frame Type

The type of encoded picture (I, P, or B) is another attribute that needs to be considered when performing estimation. Earlier work observed that use of I frames alone for PRNU estimation yields more reliable results [6]. This was a reflection of the fact that I frames are compressed at lower compression levels than P and B frames. In fact, I frames are encoded by intra-frame prediction where blocks are only predicted from edge pixels of neighboring blocks within the same frame. The edge pixels of the left or top neighbouring block are extrapolated (extended) in one of the several possible directions. This is used as the prediction block. While this method might predict lines or one directional straight patterns, it suffers if the pattern to be predicted has non-straight content such as curves, corners, *etc*. On the contrary, P and B frames are coded with inter-prediction which leverages a set of reference frames to perform block predictions. In the implementation, the previously decoded frames are stored in the Decoded Picture Buffer (DPB), P or B blocks can use any frame in their DPB to find the best match of the current block. The size of the DPB is decided by the encoder. As a result of this search space that extends over several frames, P and B frames block predictions result with better matching blocks. In other words, the prediction error that will be subject to quantization will be of lesser strength for P and B frames. Therefore, at the same compression level with I frames, P and B frames should be expected to produce more reliable PRNU patterns.

To test the dependency of PRNU estimation on the choice of frame type, we used raw (almost uncompressed) videos captured by different number of cameras. These videos are compressed at all QP levels while using two different GOP structures. In the first case, two videos are created by setting the GOP size to 3, thereby creating a repeating sequence of I, P, and B frames. For the second case, we used five videos that are encoded using a typical GOP structure where each I or P frame is followed by two B frames with a GOP size of 25 frames. Following compression, frames are extracted and grouped based on the picture type. For all QP levels and picture types, average PCE values were calculated and resulting values were normalized with respect to average PCE value of I frames.

Figure 7 shows the resulting PCE-QP curves for the three frame types for both encoding settings. When the GOP size is small, as shown in Fig. 7-a, B frames yield the best estimates with PCE values up to a few times higher than those of I frames, and P frames are slightly better than I frames at all compression levels. However, the difference becomes more apparent only at high compression levels where PRNU estimates are in general not very reliable as found in Fig. 3. With the increased GOP size, we observed a similar trend up to QP values of 13-14 where all picture types yield comparable estimates as shown in Fig. 7-b. In contrast, however, for QP values in the range of 15-25, PRNU estimates from P and B frames result with significantly higher PCE values, up to 16 times higher than those of I frames, with B frames yielding slightly better results than P frames. This difference in PCE values start dropping when QP values go beyond 22-23, which is mostly likely caused by the weakening of the PRNU patterns by increased quantization levels.

When evaluated together, these findings verify the intuition that at the same compression level B and P frames yield slightly better estimations than I frames. Further, an increase in the GOP size translates to a more accurate prediction of macroblocks and, correspondingly, to a more pronounced difference. In terms of its implication on the weighting scheme, this implies P and B macroblocks must be weighted more heavily than I macroblocks. However, in typical coding scenarios I frames are rare and I macroblocks will be much far fewer P and B macroblocks especially at lower bitrates. Our measurements on the tested videos revealed that in videos compressed at QP =15 level, I macroblocks constitute only 0.8% of the overall number of blocks in P and B frames with further decreasing ratios at higher QP values. Therefore, we didn't incorporate macroblock types as another weighting factor when generating a mask for each frame.

V. Experimental Results

To determine the effectiveness of introduced approaches in countering video compression artefacts, we performed a number of tests using two datasets.[2]

A. Tests on Custom Dataset

The videos in this dataset are acquired under known compression settings by controlling other factors (such as stabilization, electronic zoom, in-camera downsizing) which may interfere with our evaluation. Using this dataset, we first compare the performance of basic PRNU estimation method with its improved versions by successively incorporating loop filter compensation and PRNU weighting methods into it. The dataset used for this test included videos captured by 28 Android phone cameras that support H.264 compression using our custom camera application with camera motion [12]. (See Appendix for the list of cameras used to acquire videos.) For each PRNU estimation method, cameras' reference PRNU patterns were obtained from 4 seconds long, high-quality videos captured at 25 frames per second. Another set of videos compressed at a relatively low bitrate of 2.2 Mbps were used for testing. The test videos were also limited to 4 seconds in duration, recorded at the same frame rate with resolutions of 2160×2160, 1920×1080 or 1440×1080. Stabilization and electronic zoom were turned off in all cases and GOP size was set to 25 frames with each camera using the Android default structure where an I frame is followed by only P frames.

Table II provides average PCE values computed between the reference pattern and estimated PRNU patterns from all video frames and the average QP values measured in each video. It can be seen from these results that turning off loop filter improves measured PCE values on average 2.6 times over the basic PRNU estimation method. More critically, for videos on which the basic method failed but loop filter compensation alone yielded a higher PCE value than the commonly accepted threshold of 60 (videos #8-17) the improvement is more notable with a 3 times average increase. Incorporation of masking further improves results in all videos (except for video #4) with the PRNU weighting yielding the best results as expected. It is surprising to note that with binary masking even when more than 95% of all blocks are eliminated (*i.e.,* $QP > 28$), its contribution still remains significant. Results show that the average improvement due to the PRNU weighting over loop filter compensation is 70% and it is 4.44 times over the basic estimation method. Similarly, on videos #6-17, where the proposed method achieved correct attribution but basic method did not, the improvement in PCE values is more significant with a 4.8 times average increase with respect to the basic method. Alternatively, when only PRNU weighting is applied (without loop filter compensations), a noticeable improvement is still observed over the basic method, with more visible effects on videos #14-17. Overall, however, the impact of loop filter compensation is observed to be more prominent than PRNU weighting alone.

[2]An implementation of our method can be obtained at https://github.com/VideoPRNUExtractor

The second test involved 36 videos captured at 480p and 720p resolutions by the same 28 cameras. (With some of the cameras videos in 720p resolution were not recorded; therefore, for those cameras only videos at 480p resolution are included.) The videos were captured under similar controlled settings with the lightest in-camera compression (*i.e.*, $QP = 1$). These videos were considered as raw and were re-encoded at 12 different bitrates using both H.264 and H.265 codecs. During encoding GOP size is fixed at 25 frames while using a variable GOP structure determined by each codec. PRNU estimation is performed as before using the four methods. Obtained average PCE values for the two codecs are given in Tables III and IV. It can be seen that loop filter compensation and PRNU weighting improves measured results by 53% for H.264 coding and by 68% for H.265 coding. The improvement is more visible at compression bitrates of 600-700 kbps where PRNU weighting yielded 60% to two times greater PCE values and made it possible to overcome the threshold for attribution. Tables V and VI provide similar results normalized with respect to frame resolution so that compression results can be evaluated in bits per pixel. It must be noted that, with binary masking, at low bitrates many macroblocks undergo heavy compression and get masked out; therefore, not enough macroblocks are left to perform reliable estimation.

Another important evaluation criterion is the length of the minimum duration video needed to reliably identify the source camera. This is a major concern in practice when large number of videos need to be processed and computational resources are limited. To determine this, we used 21, 30-second long videos captured at 15 Mbps bitrate under controlled settings. As earlier, videos are re-encoded at 12 bitrates using H.264 and H.265 codecs. Using all the methods other than the binary masking, we estimate the PRNU pattern from all accumulated frames at the end of each second until the match with the reference PRNU pattern yields a PCE value of 60, the typical threshold used for verifying the source of photos. For each method we obtained the average duration in seconds required to achieve the target PCE threshold value. Obtained average times are presented in Table VII and Table VIII for H.264 and H.265 encoding, respectively. It can be seen that for low bitrate videos (500-2,500 Kbps), the two methods combined reduce the required duration to 35% and 38% of what is needed by the basic method, respectively, for H.264 and H.265.

B. Tests on VISION Dataset

This dataset includes a collection of videos and photos taken by 35 camera models [13]. These videos are grouped into different sub-categories depending on encoding, camera motion, and content characteristics. All videos are about 70 seconds long and initially acquired using the native camera application under three camera motion models with different complexity (*i.e.,* still, move, or panrot). A very large portion of these videos are later re-encoded by sharing them through YouTube and WhatsApp. In terms of their content, videos are also categorized into three as having flat, indoor, or outdoor scenes. In our experiments, we utilized the 16 cameras that don't

TABLE II
AVERAGE PCE VALUES FOR H.264 CODED VIDEOS AT 2.2 Mbps

Video Number	Basic method	Loop filter compensation	Only PRNU weighting	Binary masking	PRNU weighting	Average QP	Ratio of QP>28(%)
1	0.3	7.4	8.5	13.5	13.5	45.7	95.4
2	3.6	19.9	10.2	20.3	21.8	26.8	37.8
3	4.8	22.0	18.4	20.5	23.0	39.6	91.2
4	7.9	30.0	9.5	21.9	23.1	26.8	40.0
5	11.3	36.3	13.0	52.6	52.9	38.7	95.4
6	12.7	37.4	15.4	58.6	67.5	37.1	92.0
7	15.9	50.5	22.8	71.8	78.6	39.0	96.2
8	16.5	61.9	35.4	89.1	94.3	34.2	83.6
9	21.6	73.7	20.5	95.5	95.5	43.0	98.6
10	25.8	76.0	28.7	109.0	109.9	38.9	85.5
11	30.2	94.7	31.5	123.2	120.3	35.5	98.4
12	37.7	94.7	36.5	128.0	127.1	36.0	93.2
13	39.0	96.7	57.2	172.9	169.6	34.2	83.1
14	43.1	99.7	64.2	185.9	182.96	36.8	93.3
15	43.8	147.8	101.4	193.2	215.2	29.5	49.8
16	45.3	149.2	97.2	229.5	290.9	41.9	98.2
17	59.3	162.2	154.3	275.7	294.7	33.5	90.8
18	68.7	280.6	145.7	280.8	316.1	30.8	89.9
19	117.0	407.7	132.5	497.1	497.1	35.3	94.8
20	137.0	526.0	214.8	653.3	627.3	31.3	56.4
21	174.7	531.4	348.6	862.2	902.4	27.1	44.3
22	226.1	598.8	584.7	1048.7	1095.8	32.5	90.3
23	887.9	1694.8	891.6	1596.0	1673.5	10.8	17.3
24	5264.2	5810.7	5154.3	5708.7	5569.4	28.9	52.5
25	5969.3	6085.5	5814.7	5829.7	5807.6	22.5	24.2
26	7993.2	8806.0	7981.4	9498.8	8792.6	22.8	38.4
27	30930.7	34308.7	31154.1	33850.7	34271.3	16.0	30.7
28	97827.4	99403.1	97824.6	98487.6	99108.9	21.3	18.9

TABLE III
AVERAGE PCE VALUES FOR H.264 ENCODED VIDEOS AT DIFFERENT BITRATES

Bitrate (kbps)	Basic method	Loop filter compensation	Binary masking	PRNU weighting
500	9.1	14.8	9.3	16.0
600	21.3	29.2	28.8	38.8
700	44.6	55.3	54.2	70.8
800	81.5	91.4	92.1	125.2
900	196.7	218.5	222.7	317.0
1200	371.6	414.6	424.0	615.5
1500	615.7	685.4	697.6	1000.4
2000	1334.34	1449.22	1475.15	2034.5
2500	2303.2	2454.8	2491.2	3286.8
3000	3344.9	3525.8	3580.7	4553.3
3500	4422.2	4580.9	4648.9	5736.1
4000	5492.5	5606.4	5690.0	6806.6

TABLE IV
AVERAGE PCE VALUES FOR H.265 ENCODED VIDEOS AT DIFFERENT BITRATES

Bitrate (kbps)	Basic method	Loop filter compensation	Binary masking	PRNU weighting
500	13.0	16.0	19.9	27.2
600	31.1	40.0	46.5	64.6
700	51.2	65.0	72.4	101.1
800	92.1	110.0	117.2	169.6
900	207.2	265.0	275.7	404.6
1200	390.5	474.0	492.7	692.1
1500	644.4	774.0	802.7	1101.9
2000	1269.8	1458.0	1489.9	1967.9
2500	1957.5	2178.0	2219.4	2811.3
3000	2772.0	3001.0	3043.6	3702.9
3500	3512.1	3737.0	3787.3	4461.9
4000	4281.7	4460.0	4515.7	5193.6

support stabilization with 10-37 videos available per camera. This provided us with 296 videos captured by the native application, 286 YouTube videos, and 293 WhatsApp videos for the tests.

In terms of their compression characteristics, the initially captured videos have a resolution of either 1080p or 720p with a compression bitrate of 9.7-19.7 Mbps. YouTube videos are compressed by a factor of 5-7 (down to 1.7-3.3 Mbps) while preserving the frame resolution. WhatsApp videos are further compressed (1.06-1.29 Mbps) and all videos are reduced in resolution to 848×480 pixels. One potential complication with the use of YouTube and WhatsApp videos, or multiple compressed videos in general, is that the loop filter will have been already applied before the re-encoding. Since the loop filter performs a low-pass function, this results with an additional weakening of PRNU patterns. Similarly, the QP values need not necessarily reflect the actual level of quantization each macroblock underwent as earlier compression setting cannot be known. Hence, our method will be used to compensate for the effects of the most recently applied loop filter and utilize observed QP values will be treated as the actual QP values.

Since compression settings are mainly determined by the encoder implementation, we divided available videos into three groups which will be referred to as Native-coded,

TABLE V

AVERAGE PCE VALUES FOR H.264 ENCODED VIDEOS AT DIFFERENT COMPRESSION RATIOS IN BITS/PIXELS

Bits per pixels	Basic method	Loop filter compensation	Binary masking	PRNU weighting
0.7	38.7	49.5	56.7	55.3
1.2	135.6	134.1	157.8	156.7
1.6	165.9	170.7	219.9	229.3
2.0	208.2	225.9	292.2	308.4
2.4	166.3	225.1	279.4	309.1
2.8	581.7	649.9	867.6	986.6
3.5	1418.4	1554.7	1977.1	2229.9
4.5	1822.7	1985.7	2470.4	2742.8
5.5	3804.8	4014.5	4856.1	5308.2
7.3	1967.5	2076.3	2501.5	2729.0
9.6	3834.2	3961.5	4573.4	4853.2
12.4	4946.7	4945.6	5480.5	5726.3

TABLE VI

AVERAGE PCE VALUES FOR H.265 ENCODED VIDEOS AT DIFFERENT COMPRESSION RATIOS IN BITS/PIXELS

Bits per pixels	Basic method	Loop filter compensation	Binary masking	PRNU weighting
0.8	51.1	65.0	85.4	80.6
1.2	124.6	118.1	154.2	212.7
1.6	179.1	141.9	182.1	311.5
2.0	263.2	223.7	277.2	445.4
2.4	235.8	205.2	250.8	397.6
2.8	654.7	433.1	529.7	1169.2
3.5	1362.9	781.3	921.7	2210.4
4.5	1692.6	1111.6	1286.9	2583.8
5.5	3246.1	2454.1	2736.3	4527.2
7.3	1533.1	1309.8	1513.5	2071.7
9.6	2974.4	2631.0	2899.0	3567.5
12.4	3665.6	3052.2	3280.9	4062.8

TABLE VII

AVERAGE DURATION OF AN H.264 CODED VIDEO NEEDED FOR RELIABLE MATCH (SECONDS)

Bitrate (kbps)	Basic method	Loop filter compensation	PRNU weighting
500	14.18	12.45	4.59
600	12.82	11.27	3.95
700	9.95	8.32	3.32
800	8.91	6.95	3.14
900	7.00	6.09	2.55
1200	4.14	4.23	1.91
1500	3.36	3.14	1.32
2000	2.41	2.14	1.09
2500	1.77	1.77	0.91
3000	1.59	1.55	0.95
3500	1.41	1.14	0.95
4000	1.27	1.27	0.91

YouTube-coded, or WhatsApp-coded. We performed tests separately on each group following the below procedure.

- A pair of reference PRNU patterns are extracted by using the basic method and the proposed method from a randomly chosen video among the three sub-categories of videos labeled as *flat-still*, *indoor-move*, and *outdoor-panrot*.
- From each of the remaining videos three PRNU patterns are extracted using the basic method, by loop filter compensation alone, and with PRNU weighting.

TABLE VIII

AVERAGE DURATION OF AN H.265 CODED VIDEO NEEDED FOR RELIABLE MATCH (SECONDS)

Bitrate (kbps)	Basic method	Loop filter compensation	PRNU weighting
500	14.77	12.00	5.59
600	12.64	9.68	4.36
700	8.73	7.18	3.36
800	8.00	5.68	2.82
900	6.05	4.41	2.09
1200	3.64	3.05	1.50
1500	2.64	2.00	1.27
2000	1.77	1.64	0.91
2500	1.64	1.64	0.91
3000	1.36	1.23	0.77
3500	1.23	1.18	0.68
4000	1.05	0.86	0.64

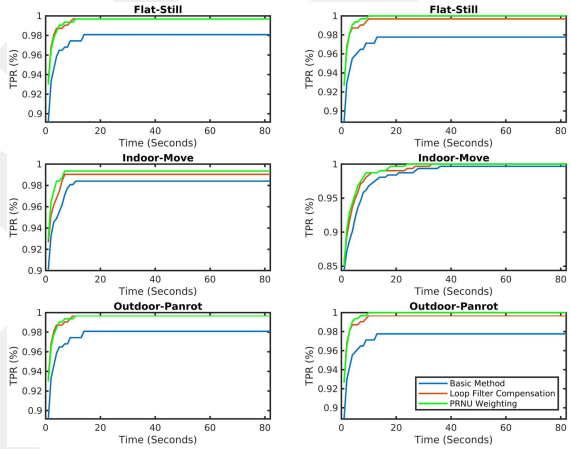

Fig. 8. Measured attribution rates on Native-coded videos as a function of video length when the reference PNRU pattern is obtained by the proposed method (first column) or the basic method (second column). Each row corresponds to a case where the reference PRNU pattern is obtained from a different sub-category of videos.

- After each second of a video (30 frames) the three PRNU patterns are updated and compared to the two reference PRNU patterns obtained in the first step.

Using this procedure and assuming a decision threshold of PCE value 60 we determined the successful attribution rate of each PRNU estimation method as a function of the length of videos used for estimating PRNU patterns. Through these tests, we also determined how the type of video used in obtaining a reference PRNU pattern affects performance as well as the effect when reference and tested PRNU patterns are estimated using (mis)matching methods.

Figures 8, 9, and 10 display the change in attribution rates depending on the length of videos when the reference PRNU pattern is obtained from different sub-category of videos. Corresponding achievable attribution rates are summarized in Tables IX, X, and XI. These results show that for the native-coded videos, the proposed method increases the best achievable performance by the basic method from 98% to 100%. Similarly, the improvement in attribution of YouTube-coded videos is from 90% to 95% and for WhatsApp-coded it is from 81% to 85%. These results also

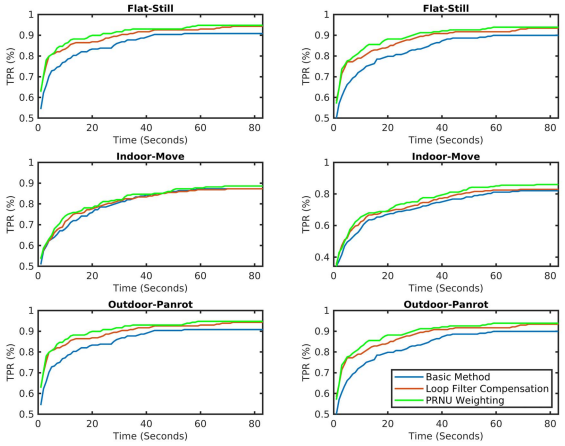

Fig. 9. Measured attribution rates on YouTube-coded videos as a function of video length when the reference PNRU pattern is obtained by the proposed method (first column) or the basic method (second column). Each row corresponds to a case where the reference PRNU pattern is obtained from a different sub-category of videos.

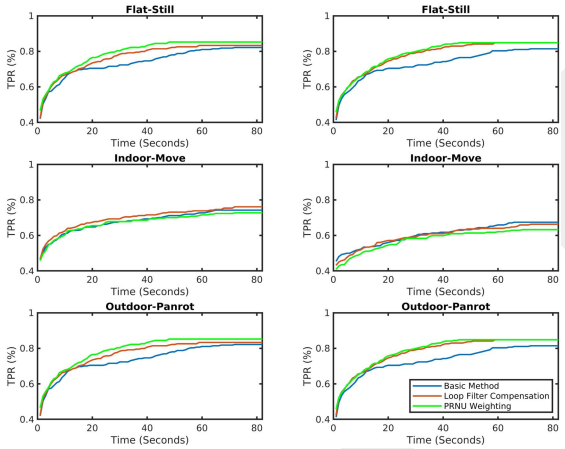

Fig. 10. Measured attribution rates on WhatsApp-coded videos as a function of video length when the reference PNRU pattern is obtained by the proposed method (first column) or the basic method (second column). Each row corresponds to a case where the reference PRNU pattern is obtained from a different sub-category of videos.

TABLE IX
RESULTS ON NATIVE-CODED VIDEOS

Reference PRNU Pattern		Tested PRNU Pattern		
Estimation Method	Video Sub-Category	Basic Method	Loop Filter Compensation	PRNU Weighting
Basic Method	Flat-Still	97.8	99.7	100
	Indoor-Move	99.7	100	100
	Outdoor-Panrot	97.8	99.7	100
PRNU Weighting	Flat-Still	98.0	99.7	99.7
	Indoor-Move	98.4	99.4	99.4
	Outdoor-Panrot	98.0	99.7	99.7

show that obtaining a reference PRNU pattern from a video captured under high camera motion poses the greatest challenge to accurate attribution due to motion blur introduced to successive video frames. Therefore, this type of videos must be deprioritized when obtaining a reference PRNU pattern.

At a given level of accuracy, as expected, the PRNU weighting method requires shorter videos than just performing loop filter compensation. When the best achievable attribution rate is considered, PRNU weighting is observed to improve marginally over loop filter compensation. This is because with the increasing length of videos, it becomes easier for the PCE value to surpass the designated decision threshold of 60. As compared to the basic method, however, the effect of video length becomes more apparent on WhatsApp-coded videos where the proposed method requires 20 seconds shorter videos to achieve the best accuracy. It must be noted that results with the proposed method should be expected to further increase when videos are acquired using the camera applications built into YouTube and WhatsApp applications, rather than performing double compression.

TABLE X
RESULTS ON YOUTUBE-CODED VIDEOS

Reference PRNU Pattern		Tested PRNU Pattern		
Estimation Method	Video Sub-Category	Basic Method	Loop Filter Compensation	PRNU Weighting
Basic Method	Flat-Still	89.9	93.4	93.9
	Indoor-Move	82.0	82.9	85.9
	Outdoor-Panrot	89.9	93.9	93.4
PRNU Weighting	Flat-Still	90.8	94.3	94.7
	Indoor-Move	87.2	87.3	88.6
	Outdoor-Panrot	90.8	94.3	94.7

TABLE XI
RESULTS ON WHATSAPP-CODED VIDEOS

Reference PRNU Pattern		Tested PRNU Pattern		
Estimation Method	Video Sub-Category	Basic Method	Loop Filter Compensation	PRNU Weighting
Basic Method	Flat-Still	81.4	84.8	84.8
	Indoor-Move	67.4	66.2	63.2
	Outdoor-Panrot	81.4	84.8	84.8
PRNU Weighting	Flat-Still	82.2	83.3	85.2
	Indoor-Move	74.2	76.1	72.7
	Outdoor-Panrot	82.2	83.3	85.2

C. Impact on False Attribution

To evaluate whether the proposed method causes an increase in false attribution rate, we performed tests on videos corresponding to 40 different cameras in both datasets. For this purpose, we utilized 28 videos obtained by cameras in the custom dataset as well as 72 Native-coded and YouTube-coded videos captured by 12 cameras in the VISION dataset, *i.e.,* 6 videos from each of the 12 cameras. From each video, we extracted two PRNU patterns, one using the basic method and the other with the proposed method. This overall yielded us two sets of 100 PRNU patterns. We then evaluated pair-wise matches between all PRNU patterns in both sets. With the basic method, the average PCE value across all matches is measured to be 0.12 with a standard deviation of 5.3, where the highest observed PCE value was 49.9. Similarly using the proposed method, the PCE values are measured to have a mean of -0.12 and a standard deviation of 6.4, with the highest observed PCE value being 52.6. Hence, both settings yielded PCE values with similar statistics, and no false matches were observed (*i.e.,* PCE$<$ 60). This finding essentially verifies our earlier intuition that blockiness effect introduced by video compression does not exhibit a deterministic pattern to cause spurious matches. In addition, it shows that PRNU weighting

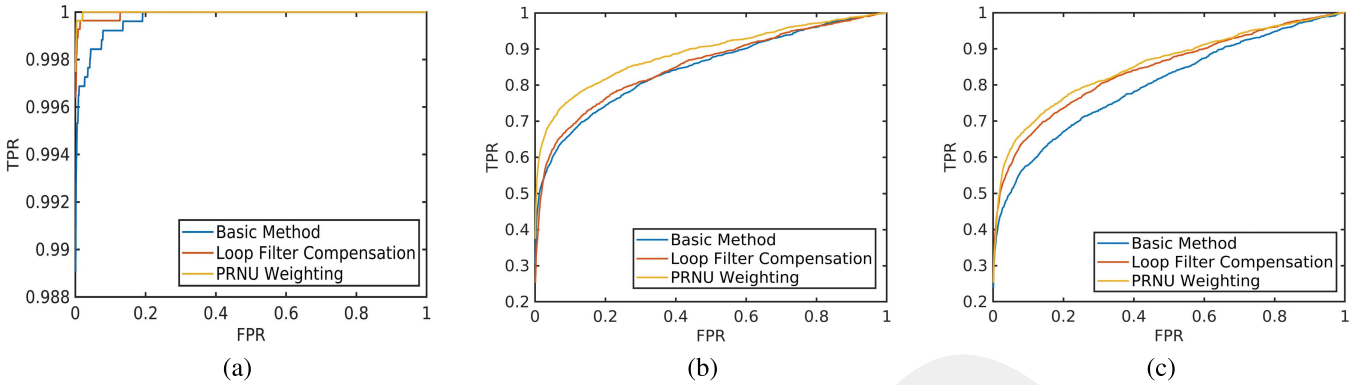

Fig. 11. ROC curves comparing the attribution accuracy of the three methods on (a) native-coded, (b) YouTube-coded, and (c) WhatsApp-coded videos.

does not introduce a bias in the PCE values when there is a mismatch with the reference PRNU pattern.

To further evaluate the impact on the false positive rate, we performed new experiments on the VISION dataset. For this purpose, PRNU patterns are obtained from each of the unstabilized videos using the three methods. All patterns are estimated utilizing the first 20 seconds of videos, which is an adequate duration to obtain a reliable estimate as indicated by our earlier results. These patterns are then matched against each other considering intra-camera and inter-camera settings by computing PCE values. For native-coded videos, this yielded 2776 values for the case when the two PRNUs matched and 82176 values when they mismatched. Similarly, for YouTube-coded videos, 2280 matching and 54074 non-matching cases and, for WhatsApp-coded videos, 2217 matching and 92029 non-matching cases are evaluated. These measurements are finally used to create receiver operating characteristic (ROC) curves for each group of videos as presented in Fig. 11. It can be seen from these curves that at a fixed true positive detection rate, PRNU weighting consistently yields the lowest false positive rate, followed by loop-filter compensation and the basic method. Therefore, we can assert that the proposed method does not cause an increase in false positive matches.

VI. Conclusion

Estimation of PRNU pattern of a sensor from a video involves many challenges. The change in the sensor's acquisition behavior, the necessity for stabilization, and the need for very effective compression are at the core of these challenges. Therefore, PRNU estimation and verification procedures need to be adapted to mitigate against disruptive effects of such processing steps in the camera pipeline. In this work, we introduced methods to tackle artefacts related to widely deployed H.264 and H.265 video coding standards.

Our analysis demonstrates that the use of the deblocking filter and the quantization related information loss impair PRNU pattern significantly at medium to high video compression levels. To cope with effects of these encoding steps, we took a proactive approach by modifying the decoding process and utilizing block-level encoding parameters to weigh the contribution of each macroblock in accordance with the level of compression it underwent. Results show that incorporation of the introduced loop filter compensation and PRNU weighting methods into the conventional estimation method yields on average $3-5$ times increase in measured PCE values. This essentially means at a given video length a higher attribution accuracy can be achieved or, alternatively, shorter videos can be used to achieve a target level of attribution accuracy. Further, the proposed method is found to be effective even on double compressed videos. Tests also indicated that the method does not cause an increase in false attributions.

Our results further show that at the same compression level P and B type frames yield more reliable PRNU estimates than I frames primarily due to the ability to accurately predict P and B type macroblocks from other frames. This in turn introduces lesser quantization noise and thus lesser distortion on the PRNU pattern. We determined that this effect gets further emphasized as the number of frames used for prediction increases. Our findings also implicitly verify the superiority of H.265 encoding over H.264 in preserving image quality at lower compression levels. It is observed that at low to medium bitrates, H.265 coded videos yield higher PCE values than H.264 coded videos. In contrast, at higher bitrates, this trend switches which indicates that H.264 is less intrusive to content in that compression regime.

Finally, we note that the ability to mitigate video coding artefacts marks a vital step in devising effective source attribution methods for stabilized video.

Appendix

The list of camera makes and models used for capturing videos under controlled settings by a custom camera application [12]. Some camera models had more than one instances as denoted in parentheses.

Sony Experia Z3	GM 5 Plusd (2)	Galaxy S4	HTC M9
Lenova P1m	Lenova S90	LG G3	LG G4
LG G Flex 2	Galaxy A3	Galaxy A5	Galaxy A7
Samsung G361H	Galaxy S5	Galaxy S6	Galaxy S7 (2)
Galaxy Note 8	Galaxy J5	Galaxy J7 (2)	Galaxy S9
Galaxy Note 3	Galaxy Tab A	Casper V10	Venus (2)

References

[1] J. Xu, H.-W. Chang, S. Yang, and M. Wang, "Fast feature-based video stabilization without accumulative global motion estimation," *IEEE Trans. Consum. Electron.*, vol. 58, no. 3, pp. 993–999, Aug. 2012.

[2] (Sep. 2018). *Android Developers*. [Online]. Available: https://developer.android.com/reference/android/hardware/camera2/Captur%eRequest#CONTROL_VIDEO_STABILIZATION_MODE

[3] Sep. 2018. *Developer.Apple.Com*. [Online]. Available: https://developer.apple.com/library/archive/documentation/DeviceInforma%tion/Reference/iOSDeviceCompatibility/Cameras/Cameras.html#

[4] M. Chen, J. Fridrich, M. Goljan, and J. Lukáš, "Source digital camcorder identification using sensor photo response non-uniformity," *Proc. SPIE, Secur., Steganogr., Watermarking Multimedia Contents IX*, vol. 6505, Mar. 2007, Art. no. 65051G.

[5] D.-K. Hyun, C.-H. Choi, and H.-K. Lee, "Camcorder identification for heavily compressed low resolution videos," in *Computer Science and Convergence* (Lecture Notes in Electrical Engineering), vol. 114, J. J. Park, H. C. Chao, M. S. Obaidat, and J. Kim, Eds. Springer, 2012, pp. 695–701.

[6] W.-H. Chuang, H. Su, and M. Wu, "Exploring compression effects for improved source camera identification using strongly compressed video," in *Proc. 18th IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2011, pp. 1953–1956.

[7] E. Altinisik, K. Tasdemir, and H. T. Sencar, "Extracting PRNU noise from H.264 coded videos," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, Sep. 2018, pp. 1367–1371.

[8] E. K. Kouokam and A. E. Dirik, "PRNU-based source device attribution for youtube videos," *Digit. Invest.*, vol. 29, pp. 91–100, Jun. 2019.

[9] D. Marpe, T. Wiegand, and G. J. Sullivan, "The H.264/MPEG4 advanced video coding standard and its applications," *IEEE Commun. Mag.*, vol. 44, no. 8, pp. 134–143, Aug. 2006.

[10] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[11] V. Sze, M. Budagavi, and G. J. Sullivan, "High efficiency video coding (HEVC)," in *Integrated Circuit and Systems, Algorithms and Architectures*, vol. 39. Springer, 2014, p. 40.

[12] E. Tandogan, E. Altinisik, S. Sarimurat, and H. T. Sencar, "Tackling in-camera downsizing for reliable camera ID verification," *Electron. Imag.*, vol. 13, no. 5, p. 545, Jan. 2019.

[13] D. Shullani, M. Fontani, M. Iuliani, O. A. Shaya, and A. Piva, "VISION: A video and image dataset for source identification," *EURASIP J. Inf. Secur.*, vol. 2017, no. 1, p. 15, 2017.

[14] M. Chen, J. Fridrich, M. Goljan, and J. Lukäs, "Determining image origin and integrity using sensor noise," *IEEE Trans. Inf. forensics security*, vol. 3, no. 1, pp. 74–90, Mar. 2008.

[15] M. Goljan, "Digital camera identification from images—Estimating false acceptance probability," in *Digital Watermarking* (Lecture Notes in Computer Science), vol. 5450, H. J. Kim, S. Katzenbeisser, and A. T. S. Ho, Eds. Springer, 2008, pp. 454–468.

[16] M. Chen, J. Fridrich, and M. Goljan, "Digital imaging sensor identification (further study)," *Proc. SPIE, Secur., Steganogr., Watermarking Multimedia Contents IX*, vol. 6505, Mar. 2007, Art. no. 65050P.

[17] D. Cozzolino and L. Verdoliva, "Noiseprint: A CNN-based camera model fingerprint," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 144–159, May 2019.

[18] W. Van Houten and Z. Geradts, "Source video camera identification for multiply compressed videos originating from YouTube," *Digit. Invest.*, vol. 6, nos. 1–2, pp. 48–60, 2009.

[19] S. Taspinar, M. Mohanty, and N. Memon, "Source camera attribution using stabilized video," in *Proc. IEEE Int. Workshop Inf. Forensics Secur. (WIFS)*, Dec. 2016, pp. 1–6.

[20] S. Mandelli, P. Bestagini, L. Verdoliva, and S. Tubaro, "Facing device attribution problem for stabilized video sequences," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 14–27, May 2019.

[21] M. Iuliani, M. Fontani, D. Shullani, and A. Piva, "Hybrid reference-based video source identification," *Sensors*, vol. 19, no. 3, p. 649, 2019.

[22] S. Taspinar, M. Mohanty, and N. Memon, "Source camera attribution of multi-format devices," Apr. 2019, *arXiv:1904.01533*. [Online]. Available: https://arxiv.org/abs/1904.01533

[23] T. Tan, A. Fujibayashi, Y. Suzuki, and J. Takiue, "Objective and subjective evaluation of HM5.0," in *Proc. 8th Meeting*, San Jose, CA, USA, vol. 1, no. 10, Feb. 2012.

[24] T. K. Tan *et al.*, "Video quality evaluation methodology and verification testing of HEVC compression performance," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 76–90, Jan. 2016.

[25] E. Uzun and H. T. Sencar, "Carving orphaned JPEG file fragments," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 8, pp. 1549–1563, Aug. 2015.

[26] S. Bayram, H. T. Sencar, and N. Memon, "Efficient Sensor Fingerprint Matching Through Fingerprint Binarization," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 4, pp. 1404–1413, Aug. 2012.