Video source identification using machine learning: A case study of 16 instant messaging applications

By:
Hyomin Yang, Junho Kim, Jungheum Park

From the proceedings of
The Digital Forensic Research Conference
DFRWS APAC 2024
Oct 22-24, 2024

Forensic Science International: Digital Investigation 50 (2024) 301812

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

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

DFRWS APAC 2024 - Selected Papers from the 4th Annual Digital Forensics Research Conference APAC

Video source identification using machine learning: A case study of 16 instant messaging applications

Hyomin Yang [1], Junho Kim [1], Jungheum Park [*]

School of Cybersecurity, Korea University, Seoul, South Korea

A R T I C L E I N F O

Keywords:
Digital forensics
Multimedia forensics
AVC
HEVC
Source identification
Machine learning

A B S T R A C T

In recent years, there has been a notable increase in the prevalence of cybercrimes related to video content, including the distribution of illegal videos and the sharing of copyrighted material. This has led to the growing importance of identifying the source of video files to trace the owner of the files involved in the incident or identify the distributor. Previous research has concentrated on revealing the device (brand and/or model) that "originally" created a video file. This has been achieved by analysing the pattern noise generated by the image sensor in the camera, the storage structural features of the file, and the metadata patterns. However, due to the widespread use of mobile environments, instant messaging applications (IMAs) such as Telegram and Wire have been utilized to share illegal videos, which can result in the loss of information from the original file due to re-encoding at the application level, depending on the transmission settings. Consequently, it is necessary to extend the scope of existing research to identify the various applications that are capable of re-encoding video files in transit. Furthermore, it is essential to determine whether there are features that can be leveraged to distinguish them from the source identification perspective. In this paper, we propose a machine learning-based methodology for classifying the source application by extracting various features stored in the storage format and internal metadata of video files. To conduct this study, we analyzed 16 IMAs that are widely used in mobile environments and generated a total of 1974 sample videos, taking into account both the transmission options and encoding settings offered by each IMA. The training and testing results on this dataset indicate that the Extra-Trees model achieved an identification accuracy of approximately 99.96 %. Furthermore, we developed a proof-of-concept tool based on the proposed method, which extracts the suggested features from videos and queries a pre-trained model. This tool is released as open-source software for the community.

1. Introduction

With the widespread adoption of smartphones, users can easily capture photos, record videos, and audio anytime and anywhere using their mobile devices. The sharing of these multimedia files via various social networking services (SNS) and instant messaging applications (IMA) has led to a significant increase in the volume of multimedia data (Hamdi et al., 2016). However, some malicious users have exploited these platforms for illegal activities, such as distributing unlawful videos and sharing copyright-infringing materials. Consequently, the importance of verifying the authenticity and identifying the sources of vast amounts of multimedia content has been increasingly emphasized.

Researchers have been engaged in investigating multimedia analysis techniques with a view to supporting mobile forensics. Recently, deepfake multimedia generated by artificial intelligence has proliferated online, prompting the development of technologies to determine the authenticity of such data (Reis and Ribeiro, 2024; Li et al., 2023). In addition, significant research has been conducted on the subject of Photo Response Non-Uniformity (PRNU) to identify the source device of images and employing image metadata to determine the editing software used (Lawgaly and Khelifi, 2016; Ahmed et al., 2019). Research has also demonstrated the use of PRNU to link videos to the smartphones that created them (Chen et al., 2008; Dirik et al., 2008). Thus, techniques for identifying the source device of multimedia files, based on sensor PRNU, have been well-established. However, identifying sources using PRNU is a resource-intensive task (de Roos and Geradts, 2021).

When sharing videos with others, various applications perform re-encoding of the videos. During this process, the transmitted videos

* Corresponding author.
E-mail addresses: yhm3621@korea.ac.kr (H. Yang), rlawnsgh92@korea.ac.kr (J. Kim), jungheumpark@korea.ac.kr (J. Park).

[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.fsidi.2024.301812

Available online 18 October 2024

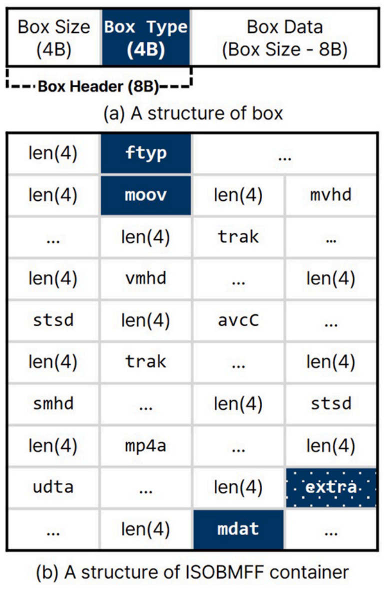


Fig. 1. Structure of box and ISOBMFF container.

contain traces of the origin of each application. Data capable of identifying the source of the video is stored within the container structure or codec of the multimedia. Amerini et al. (2017) observed that when users upload videos to Facebook and X (formerly Twitter), each service compresses or reduces the size of the video. The authors proceeded to identify the source service of uploaded videos by extracting PRNU from such videos. Altinisik et al. (2022) conducted research to classify the sources of 109 camera models, 3 video editing tools, and 4 SNS services using publicly available datasets ACID (Hosler et al., 2019), VISION (Shullani et al., 2017), SOCRatES (Galdi et al., 2019), and EVA-7K (Yang et al., 2020). Despite research into identifying the sources of videos generated by SNS services or video editing tools, there remains a lack of datasets and research on identifying the sources of videos transmitted via IMA.

In this study, we propose a method to identify the source application of mobile videos by exploiting the differences in metadata left by various encoding techniques specific to each IMA through the use of machine learning. To discern the origin of video files, we generated a substantial video dataset, capturing the unique characteristics of formats and codecs produced by different IMAs. These features are then used to train various machine learning models, and their performance is assessed. Furthermore, we developed an automated tool that can be utilized by investigators in their inquiries and made it publicly available.

The main contributions of this paper are as follows:

- Identified characteristics of multimedia container structures and codec information that can be used to determine the video source
- Proposed a systematic methodology for identifying the source application of videos transmitted by specific IMAs
- Created and publicly released a dataset comprising 1974 video files generated from 16 widely used IMAs
- Assessed the importance to the features that significantly influence the identification of video source applications
- Developed and publicly released a proof-of-concept tool based on pretrained machine learning models

This paper is organized as follows: Section 2 provides an introduction to background knowledge on multimedia container formats and video codecs, as well as a review of research related to video source identification. Section 3 explains the features for identifying the source of videos. Section 4 describes the proposed methodology. In Section 5, the methodology is applied to identify the source and evaluate the accuracy for 16 IMA applications. Section 6 introduces a concept proof tool based on machine learning models. Finally, Section 7 concludes this study and suggests directions for future research.

2. Background and related work

2.1. Container file format and video codecs

A multimedia container file format encapsulates multimedia data, such as video and audio streams, along with metadata and additional information. Mobile devices primarily use formats such as MP4, MOV, and 3 GP, while IMAs predominantly implement MP4 and MOV formats for video transmission. Both MP4 and MOV are video formats based on the ISOBMFF (ISO/IEC Base Media File Format). MP4 is a video file format encoded using the MPEG-4 (Motion Pictures Expert Group-4) Part 14 standard, created by ISO/IEC (2020). MOV, developed by Apple, is a container format primarily used by Apple's QuickTime program (Apple Inc., 2001) for video. Since the only significant difference between the two formats is compatibility depending on the manufacturer, this paper will explain the ISOBMFF container structure.

The ISOBMFF container is primarily composed of basic units known as boxes or atoms; in this paper, they are referred to as boxes. Fig. 1 illustrates the structure of a box and an ISOBMFF container. Fig. 1 (a) is a structure of box within the ISOBMFF container, and Fig. 1 (b) is the overall structure of the ISOBMFF container. Boxes are stored in a hierarchical structure within the container, with a minimum size of 8 bytes. Typically, a video based on ISOBMFF contains a file type box (`ftyp`), a movie box (`moov`), and a media data box (`mdat`). The `ftyp` box contains information for verifying file compatibility, the `moov` box holds all the multimedia metadata in its sub-boxes, and the `mdat` box stores the video, audio, and subtitles. These are the top-level boxes of a video file, and the `extra` area in Fig. 1 may include other top-level boxes, such as `free` and `wide`, aside from `ftyp`, `mdat`, and `moov`. Furthermore, the structure's characteristics can vary slightly depending on the device or software generating the multimedia. For instance, the positions of `moov` and `mdat` can be flexible, and the variety and internal composition of the top-level boxes can differ.

Multimedia containers can encapsulate videos encoded with various codecs. Initially, the process of creating a video involves compressing raw original video and audio data using specific codecs. This process, known as encoding, combines the encoded video and audio data into a container format through a muxing process. For a video player to playback the video, it undergoes a demuxing process to extract the video and audio data from the container format, which is then decoded using respective decoders. Codecs, which encode or decode the original data, come in various types. Among video codecs, the most commonly used are the H.264/AVC and H.265/HEVC codecs (Bitmovin, 2023). The H.264/ AVC codec (International Telecommunication Union, 2003), standardized jointly by ITU-T and ISO/IEC, is a motion-compensation-based compression codec, while the H.265/HEVC codec (International Telecommunication Union, 2013) employs compression techniques based on Discrete Cosine Transform (DCT) to support videos of 4K resolution and above. As for audio codecs, there are AAC and MPEG-H codecs, among others. The AAC codec, developed collaboratively by Fraunhofer Institute, Bell Labs, Dolby Labs, Sony, Nokia, and others to improve upon the MP3 codec, is currently the standard for streaming, while MPEG-H is a next-generation standard technology under development and standardization by various companies.

H.264 and H.265 codecs are controlled by a set of dynamically determined parameters by the encoder to achieve the target compression bitrate and quality. These parameters are stored along with the encoded video data because the decoder requires the same parameters to playback the video properly. For H.264, the decoding information is grouped into Sequence Parameter Set (SPS), Picture Parameter Set (PPS), and Video Usability Information (VUI) parameter sets. For H.265,

Table 1
Video transmission settings and editing options of IMAs.

Application	Resolution	Editing option
Band	High	Mute
Discord	–	–
Facebook Messenger	–	Trim, Sticker, Text, Draw, Mute
KakaoTalk	Standard, High	Trim, Rotation, Filter, Mute
Line	–	Trim, Sticker, Text, Draw, Filter, Mute
QQ	Original	Trim, Text, Mute, Crop, Rotation
Session	–	–
Signal	–	Trim
Slack	–	–
Snapchat	–	Rotation, Sticker, Text, Draw, Filter, Speed, Zoom, Horizontal Flip, Insert audio, Automatic subtitles, Mute
Teams	–	–
Telegram	240p, 360p, 480p, 720p, 1080p	Trim, Rotation, Sticker, Text, Draw, Crop, Mute, Adjust(Brightness, Contrast, Saturation, Warmth, etc.)
Viber	–	Trim, Sticker, Text, Draw, Reverse, Speed, Mute
WeChat	–	–
WhatsApp	–	Rotation, Sticker, Text, Draw, Filter, Crop, Mute
Wire	–	–

the decoding information includes SPS, PPS, VUI, and Video Parameter Set (VPS) parameter sets. The encoding parameters are stored in NAL units under the avcc box of the container file (Park and Lee, 2014).

2.2. Video re-encoding of IMA

IMAs enable real-time communication between two or more users, offering not only text messaging but also convenient video sharing capabilities. Some IMAs additionally provide features for video editing and options to select the video quality during transmission. Table 1 summarizes the quality settings and editing functionalities available in various messaging apps for video transmission. When videos are sent using these features, IMAs perform re-encoding of the videos. During this process, the structure of the video, its metadata, and encoding parameters are altered according to the specific settings of each application. Consequently, to identify the source of a video file from an IMA, it is essential to consider all transmission and editing options when creating a dataset.

2.3. Related work

Numerous studies have been conducted on identifying the source of videos. Previous research primarily focused on determining the recording camera or verifying the editing and originating application.

Altinisik and Sencar (2020) introduced a method for verifying the source camera of a video by considering the spatial transformation characteristics of video stabilization within the camera's image pipeline. Flor et al. (2021) proposed a new algorithm aimed at distinguishing between cameras of the same manufacturer and model by emphasizing pixels contributing to sensor noise in the PRNU pattern. They utilized the Jaccard index to provide a measure of the ratio of matching pixel positions shared between the noise patterns of two devices. Bennabhaktula et al. (2022) fine-tuned a pre-trained, unrestricted single MobileNet using inputs from up to 50 I-frames to identify the source camera. The authors aimed to develop cyber tools to help identify the origin of illegal content involving minors. Akbari et al. (2022) enhanced a CNN (Convolutional Neural Network) named PRNU-NET with a new PRNU-based layer to leverage the strengths of PRNU and machine learning approaches commonly used in source camera identification. Anmol and Sitara (2024) combined two texture features, LBP and GLCM, with the statistical characteristics of PRNU. They selected relevant features from the extracted 170 features using ANOVA-based univariate feature selection, and input the best feature set into an SVM classifier to

Table 2
Metadata used for video source identification.

Category	Feautre	Description	Example
CFF	Box sequence	Type and order of top box	ftyp moov free mdat
VM-G	Format profile	Base media file format version	QuickTime
	Brands	Major and compatible brands	mp42 isom mp42
	Writing application	Encoder information	Lavf58.12.100
	Movie name	Information that Line only has	Line Video
	Copyright	Copyright information	wxmmcabr00dr0000
	Overall bitrate	Average bitrate of the entire file (all stream + container overhead)	17.1Mbps
VM-A	Title	Name of the handler that handles a audio track	Core Media Audio
	ID	Order of audio stream	2
	Bitrate	Bits per second	128 Kbps
	Alternate group	Number of a group in order to provide versions of the same track	1
VM-V	Title	Name of the handler that handles a video track	Core Media Video
	ID	Order of video stream	1
	Bitrate	Bits per second	16.8Mbps
	Width	Width of frame in pixels	1280
	Height	Height of frame in pixels	720
EP-V	Format profile	Standard of video coding technology	Baseline L2.1
	Format settings	Settings used and required by decoder	CABAC 4 Ref Frames
	Color range	Color range for YUV color space	Full
	Color primaries	Chromaticity coordinates of the source primaries	BT.709
	Transfer characteristics	Opto-electronic transfer characteristic of the source picture Matrix coefficients used in deriving	BT.601
	Matrix coefficients	luma and chroma signals from the green, blue, and red primaries	BT.470 System B/G

CFF: Container File Format, VM: Video Metadata, EP: Encoding Parameter, G: General, A: Audio, V: Video.

perform video source camera identification.

Amerini et al. (2021) proposed a multi-stream neural network architecture, MultiFrame-Net, capable of capturing double compression traces left by social networks and messaging apps on videos. MultiFrame-Net performs classification on videos uploaded to YouTube and those shared via WhatsApp from the VISION dataset (Shullani et al., 2017). Kouokam and Dirik (2019) demonstrated the accuracy of identifying the source of YouTube videos using I-frame fingerprints of flat content native videos. López et al. (2020) developed a technique for identifying the source of digital videos generated by multimedia devices and source applications (e.g., YouTube and WhatsApp) using an unsupervised algorithm based on the structural analysis of multimedia devices. Huamán et al. (2020) created a dataset consisting of videos shared through 10 social networks, transmitted via 3 instant messaging applications, and manipulated by 5 editing programs, and conducted a structural comparison and analysis of the videos. Orozco et al. (2020) described a box extraction algorithm and classified the dataset provided by Huamán et al. (2020), along with additional videos downloaded from TikTok and Snapchat, using a Random Forest model.

As such, most research has focused on identifying the source camera, with relatively few studies addressing the identification of source applications. Previous studies have classified a very limited number of applications using publicly available datasets. Additionally, studies that identified a variety of applications did not make their datasets public,

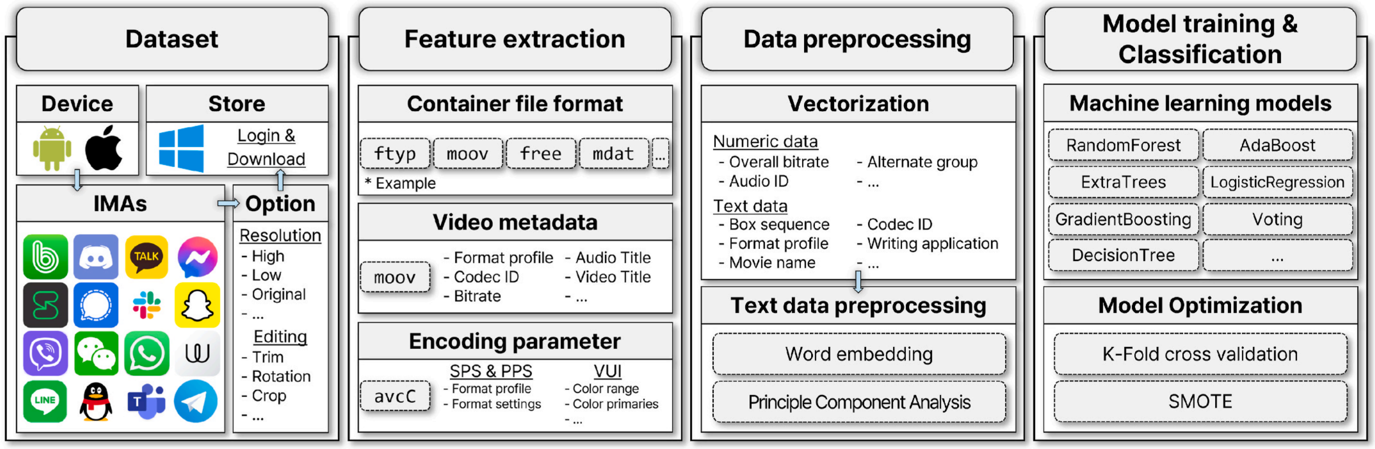


Fig. 2. An overview of the proposed methodology.

and there is a lack of research classifying a large number of instant messaging applications.

3. Metadata for video source identification

We categorized the metadata used for source classification into three types: Container File Format (CFF), Video Metadata (VM), and Encoding Parameters (EP). There are numerous detailed items within VM and EP, but we extracted and compared a substantial amount of data to select the features that appear to be applicable for source identification. Table 2 summarizes the metadata useful for source identification as described in this section.

3.1. Container file format

In the previous section, we explained that the boxes in the ISOBMFF container file format are organized in a hierarchical structure. The boxes can be stored in any order and can contain various sub-boxes, and the values of these boxes may vary depending on the source of the video. Therefore, the structure of the boxes can be used as a means to identify the source that encapsulated the video file (López et al., 2020). Several researchers have utilized the structure of CFF for source identification. López et al. (2020) considered not only the structure of the CFF but also the values of the lowest-level boxes as features, while Xiang et al. (2021) used specific boxes within the video for their study. However, these approaches are time-consuming as they require checking the hierarchy of all boxes, and it is challenging to identify the source for new devices and applications.

To enable rapid source identification, we aimed to find features that do not require extensive time for feature extraction and can be applied to the latest versions and applications. We discovered that videos transmitted through various IMAs have unique characteristics in their top-level box information (names, order) depending on the IMA, the operating system of the device, and the settings. For instance, a video sent via WhatsApp contains the boxes ftyp, beam, moov, and mdat in order, whereas a video sent to Discord using an iOS device includes the boxes ftyp, moov, wide, and mdat in sequence. However, since some applications share the same top-level box information, these data alone are insufficient for complete source classification. Therefore, additional features must be utilized to address this issue.

3.2. Video metadata

Video metadata includes various technical information related to video content, such as codec, bitrate, and subtitles (Lee et al., 2021). Metadata can be categorized into three main types: general metadata for the container file, video metadata for the video stream, and audio metadata for the audio stream. General metadata is stored in the ftyp and/moov/udta boxes, video metadata can be found in the video-related trak boxes under the moov box, and audio metadata is located in the audio-related trak boxes.

First, the *Format profile* represents the basic media file version: if the value of the/ftyp/@Major Brand field is mp42, it indicates Base Media Version 2; if it is isom, it indicates Base Media; and if it is qt, it indicates QuickTime. *Brands* includes the values of the/ftyp/@Major Brand field and the/ftyp/@Compatible Brands field. *Writing application* is the encoder information found in the/moov/udta/meta/ilst/xa9too/data/@encoder field. The *Movie name* field contains the value Line_Video only for Line among the 16 messenger apps, located in the/moov/udta/titl/@movieName field. *Copyright* is a feature related to copyright, stored in the/moov/udta/cprt/@copyright field. Finally, *Overall bitrate* is the sum of the video and audio stream bitrates and the container overhead, which can be calculated using the following formula.

$$Overall\_bitrate = file\_size \times 8 \times timescale/duration \tag{1}$$

The metadata for the video stream that can be obtained from the trak box includes the *Video Title*, which is the value of the/moov/trak/mdia/hdlr/@component name field. The *Video ID* indicates the order of the video stream and is the @track id value of the trak where the/moov/trak/mdia/hdlr/@component subtype field value is vide. *Width* and *Height* represent the video's width and height in pixels, stored in/moov/trak/mdia/minf/stbl/stsd/@width and @height. Lastly, *Video bitrate* is the calculated number of bits per second for the video stream, which can be determined using the following formula.

$$video\_stream\_size \mathrel{+}= video\_stsz\_entry \tag{2}$$

$$\begin{aligned} video\_bitrate &= video\_stream\_size \times 8 \\ &\times video\_timescale/video\_duration \end{aligned} \tag{3}$$

Metadata for the audio stream can be obtained from the audio-related trak box. The *Audio ID* indicates the order of the audio stream and is the @track id value of the trak where the/moov/trak/mdia/hdlr/@component subtype field value is soun. The *Audio Title* is stored in/moov/trak/mdia/hdlr/@component name in the audio-related trak box. The *Alternate group* is a feature that indicates the number of alternate audio tracks, stored in/moov/trak/tkhd/@alternateGroup. Lastly, the *Audio bitrate* is the calculated number of bits per second for the audio stream, which can be determined using the following formula.

$$audio\_stream\_size \mathrel{+}= audio\_stsz\_entry \tag{4}$$

Table 3
Details of target IMAs and number of datasets.

Application	Total	OS		
		OS	**Version**	**Total**
Band	114	Android	10.2.1	44
		iOS	10.2.1	70
Discord	49	Android	176.21	22
		iOS	175.0	27
Facebook Messenger	207	Android	406.0.0.13.115	110
		iOS	406.0	97
KakaoTalk	245	Android	10.2.9	110
		iOS	10.1.8	135
Line	204	Android	13.10.1	22
		iOS	13.6.1	182
QQ	218	Android	8.9.76	110
		iOS	8.9.84	108
Session	27	Android	1.16.9	–
		iOS	2.2.13	27
Signal	98	Android	6.18.4	44
		iOS	6.21.0	54
Slack	49	Android	23.04.40.0	22
		iOS	23.04.40	27
Snapchat	147	Android	12.33.1.19	79
		iOS	12.31.0	64
Teams	97	Android	1416/1.0.0.2024093502	44
		iOS	6.7.1	53
Telegram	141	Android	9.6.5	66
		iOS	9.6.3	75
Viber	147	Android	19.9.4.0	70
		iOS	19.9.1	77
WeChat	49	Android	8.0.30	22
		iOS	8.0.37	27
WhatsApp	155	Android	2.23.8.76	66
		iOS	23.20.79	89
Wire	27	Android	3.82.38	–
		iOS	3.109	27

$$audio\_bitrate = audio\_stream\_size \times 8 \times audio\_timescale / audio\_duration \quad (5)$$

3.3. *Encoding parameter*

Values useful for classifying source applications were identified not only from metadata but also from encoding parameters required for decoding. This was based on the official documents of H.264 and H.265 (International Telecommunication Union, 2003, 2013). Firstly, *Format profile* represents the video encoding technology according to the usage area and is a combination of `profile_idc` and `level_idc` in the SPS header. *Format settings* indicate the number of reference frames and whether CABAC is applied in the video stream. If the `entropy_coding_mode_flag` in the PPS header is 1, CABAC is applied, and the number of reference frames can be determined from `num_ref_frames` in the SPS header. Features related to the video's color, such as *Color range*, *Color primaries*, *Transfer characteristics*, and *Matrix coefficients*, refer to the YUV color space, the chromaticity coordinates of the source primaries, the electro-optical transfer characteristics of the source image, and the matrix coefficients used to derive luma and chroma signals from the green, blue, and red primaries, respectively. These four features are stored in the VUI.

4. Methodology for identifying video sources

In this study, we propose a methodology for classifying the source application of videos shared through IMAs based on their metadata. Fig. 2 provides an overview of the proposed methodology. First, we capture videos using the native camera application and then create a dataset by applying all editing options and transmission settings of each IMA. Features are extracted from the generated dataset and pre-processed into a format suitable for machine learning models. Finally, various machine learning algorithms and an ensemble voting model are employed to determine the optimal algorithm. To enhance classification accuracy, we utilized SMOTE and K-Fold cross-validation techniques.

4.1. *Dataset*

The video dataset was created using a Galaxy A23 for Android devices and an iPhone 13 for iOS devices. Table 3 summarizes the types and versions of applications used for dataset creation. The selection criteria for the target IMAs focused on widely used applications, including work-related messengers like Slack and Teams, as well as secure messaging apps like Telegram and Signal.

The dataset was generated by recording videos with the default camera application on the Galaxy A23 and iPhone 13. Videos were then sent using all available options for each IMA, and subsequently downloaded from web browsers or desktop applications on a Windows operating system.

For Android devices, videos were sent unaltered on Wire, and for both Android and iOS devices, the QQ application offers a quality option to select the original format, which does not alter the metadata. Consequently, these settings was excluded from the dataset, and only the editing options were used. The differences in the number of datasets per IMA, as shown in Table 3, result from the presence or absence of editing and quality options. To address this issue, additional measures were taken to mitigate data imbalance during the experiments.

4.2. *Feature extraction*

IMAs leave traces in the internal metadata that can be used to identify the source when videos are transmitted, making it necessary to extract this data. To extract the CFF described in Section 3, we analyzed the ISOBMFF container format structure and developed a custom script to extract top-level box data from the videos. While VM can be extracted using FFmpeg (FFmpeg, 2000) and MediaInfo (MediaArea, 2002), these tools do not allow extraction in the desired format. Thus, we developed source code to extract metadata similar to the CFF. Lastly, EP for the H.264 codec can be extracted using the h264bitstream tool (Aizvorski, 2014), but it does not support the analysis of EP for the H.265 codec. However, through our experiments, we confirmed that videos shared using the high-quality option on KakaoTalk are re-encoded with the H.265 codec. Therefore, it is necessary to analyze the parameters of the H.265 bitstream. Consequently, we developed source code to extract EP by referencing the standard documents for H.264 and H.265 (International Telecommunication Union, 2003, 2013).

4.3. *Data preprocessing*

To train a machine learning model using the extracted features, the features must be vectorized. These features include numerical data such as Bitrate, ID, Width, and Height, as well as textual data like Format, Format profile, and Codec ID. Numerical data can be directly used as input to machine learning models without additional preprocessing. However, textual data requires a preprocessing step to convert them into a format suitable for machine learning algorithms. The textual data were tokenized, converted into integer indices for each token, and then encoded using Word Embedding. After encoding, Principal Component Analysis (PCA) was applied to reduce the dimensionality of the features. PCA is a widely used dimensionality reduction technique that extracts principal components from the correlations among multiple variables to reduce the dimensionality.

4.4. *Model training & classification*

In this study, we compare 11 commonly used models in classification problems to select the most suitable machine learning model for video source classification. These models are classification algorithms designed to distinguish between classes in the given data. The models include Support Vector Machine (SVM), DecisionTree (DT), AdaBoost (AB), RandomForest (RF), ExtraTrees (ET), GradientBoosting (GB), Multi-Layer Perceptron (MLP), K-Nearest Neighbors (KNN), Logistic Regression (LR), Linear Discriminant Analysis (LDA), and Bernoulli Naive Bayes (BNB). Additionally, to achieve better performance, we selected the top three models and conducted further experiments using the ensemble technique of voting. Voting involves training several machine learning algorithms on the same dataset and combining their predictions through a voting process to determine the final prediction (Dietterich, 2000).

When evaluating the models, the validation data is selected randomly, which can sometimes introduce bias and reduce the reliability of the model evaluation. To address this limitation, we performed K-Fold cross-validation (Wong and Yeh, 2019). K-Fold cross-validation involves dividing the entire dataset into k smaller subsets and using each subset as validation data at least once. In this study, we used 5-Fold cross-validation with k set to 5. Additionally, to address class imbalance caused by the uneven number of samples, we used synthetic minority over-sampling technique (SMOTE), an algorithm that selects the nearest neighbors of sampled data points, connects these points with line segments, and generates new samples along these segments (Chawla et al., 2002).

5. Result and evaluation

5.1. Performance metrics

We generated a confusion matrix to evaluate the trained classification model. The confusion matrix is used to compare predicted values with actual values to measure prediction performance. It applies to both binary and multi-class classification problems. TP (True Positive) refers to cases where the actual value is True and the model correctly predicts True. TN (True Negative) indicates cases where the actual value is False and the model correctly predicts False. FP (False Positive) refers to cases where the actual value is False but the model incorrectly predicts True. FN (False Negative) indicates cases where the actual value is True but the model incorrectly predicts False. Based on these values, we used four commonly used performance metrics: Accuracy, Recall, Precision, and F1-Score.

Accuracy is the most commonly used metric for evaluating models, representing the proportion of correct predictions over the entire test dataset.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{6}$$

Recall is the ratio of correctly predicted positive values to all actual positive values.

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

Precision is the ratio of correctly predicted positive observations to the total predicted positives.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

F1-Score is the harmonic mean of Precision and Recall, and it is primarily used when there is a significant imbalance between the classes.

$$F1 - Score = 2\cdot\frac{Precision \times Recall}{Precision + Recall} \tag{9}$$

5.2. Evaluation

In this section, we evaluate the classification performance of the proposed methodology. The results of experiments conducted with the 11 machine learning model algorithms mentioned in Section 4.4 and the three best-performing models used for voting are presented in Table 4. The model performance evaluation shows that the ExtraTrees model outperformed the voting model, with RandomForest and GradientBoosting models achieving the next highest classification accuracy. We generated a confusion matrix for the ExtraTrees model, which provided the best classification of video sources. Fig. 3 illustrates the confusion matrix for class classification using the ExtraTrees model.

Table 4
Classification performance by machine learning models.

Model	Accuracy	Precision	Recall	F1-Score
ExtraTrees	99.9247	99.9275	99.9254	99.9253
Voting (RF, ET, GB)	99.7742	99.7826	99.7750	99.7754
RandomForest	99.7366	99.7468	99.7382	99.7370
GradientBoosting	99.5107	99.5283	99.5109	99.5130
DecisionTree	98.2678	98.3523	98.2665	98.2607
K-Neareest Neighbors	91.7178	91.9144	91.7291	91.6135
AdaBoost	68.2210	67.7052	68.1952	63.9668
Multi-Layer Perceptron	64.8373	66.3409	64.8362	62.2051
Linear Discriminant Analysis	62.8775	68.3619	62.8844	59.8663
Support Vector Machine	47.0256	45.9592	47.0711	40.5715
Logistic Regression	45.5564	41.0556	45.5515	40.4119
Bernoulli Naive Bayes	40.4368	32.4767	40.4846	31.5011

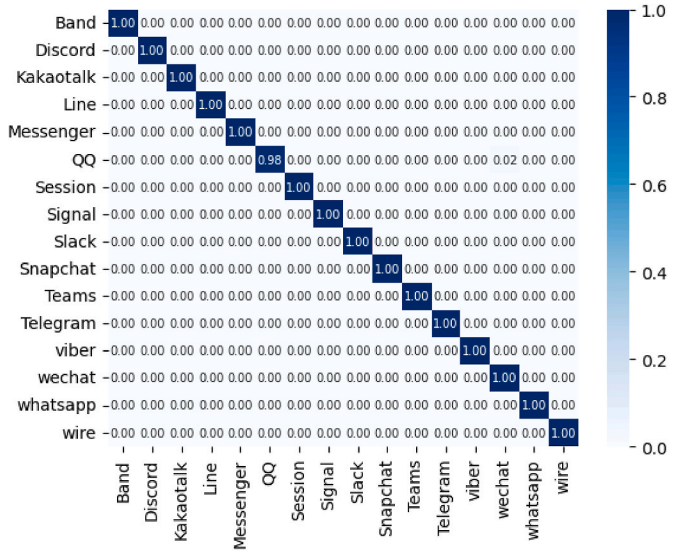


Fig. 3. Confusion matrix for IMA classification.

Table 5
Feature importance in ExtraTrees

Feature	Importance (%)	Feature	Importance (%)
CFF-Box Sequence	9.2	EP-V-Matrix coefficient	4.3
EP-V-Format settings	8.5	VM-V-Title	3.8
VM-G-Writing application	6.3	VM-G-Format profile	3.7
VM-V-Width	6.2	VM-G-Brands	3.7
EP-V-Format profile	6.1	VM-A-Title	3.6
VM-G-Movie name	5.9	VM-A-ID	3.5
VM-A-Bitrate	5.7	EP-V-Color primaries	2.3
VM-V-Bitrate	5.2	EP-V-Transfer characteristics	2.3
VM-V-Height	5.2	VM-A-Alternate group	2.2
VM-V-ID	4.7	VM-G-Copyright	1.8
VM-G-Overall bitrate	4.6	EP-V-Color range	1.5

Table 6
Classification performance by number of metadata based on feature importance in ExtraTrees

ET-# of features	Accuracy	Precision	Recall	F1-Score
ET-8	99.7364	99.7447	99.7382	99.7352
ET-9	99.8117	99.8183	99.8151	99.8128
ET-10	99.8117	99.8183	99.8128	99.8117
ET-11	99.8870	99.8897	99.8875	99.8869
ET-12	99.8493	99.8540	99.8485	99.8490
ET-13	99.8494	99.8540	99.8507	99.8501
ET-14	99.9623	99.9643	99.9621	99.9627
ET-15	99.9623	99.9643	99.9621	99.9627
ET-16	99.9247	99.9306	99.9242	99.9252

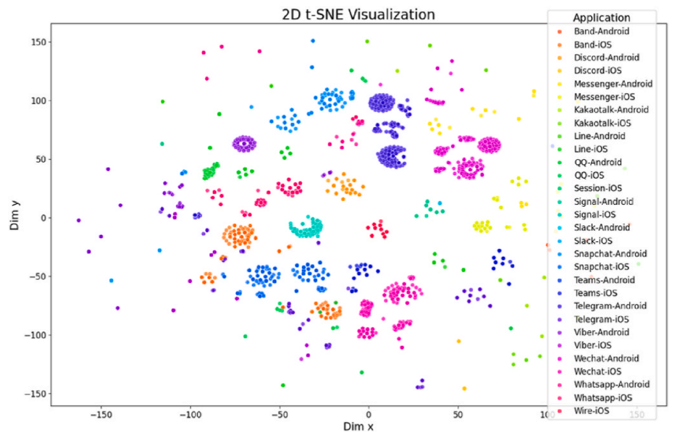


Fig. 4. 2D t-SNE visualization illustrating the effectiveness of the selected features in differentiating between application sources.

Among the 16 classes, data from the QQ application was misclassified as WeChat by 2 %, and the overall classification accuracy was 99.92 %.

5.3. *Assessment of feature importance*

In this section, we evaluate feature importance and identify the optimal feature set for the best-performing machine learning model, ExtraTrees. Table 5 summarizes the importance of each feature when classifying video sources using the ExtraTrees model. Since the top-level box sequence is unique for each IMA, *CFF-Box Sequence* had the highest importance. The EP features, such as *EP-V-Format settings* and *EP-V-Format profile*, also had high importance because videos transmitted through the same application often have identical values. The *VM-G-Movie name* feature showed high importance because it is metadata that is stored only in videos sent via Line. The *VM-A-Bitrate* and *VM-G-Bitrate* represent the number of bits processed per unit of time. A higher *VM-A-Bitrate* generally improves audio quality but increases file size. Similarly, a higher *VM-V-Bitrate* improves video quality but also increases file size, while a lower bitrate reduces quality. Since each messaging app uses appropriate *VM-A-Bitrate* and *VM-V-Bitrate* settings, these features are highly important for classifying the source messaging app.

Through experimentation, we manually selected a total of 22 features from a vast array of metadata. Applying these features to the ExtraTrees model, we achieved the accuracy of 99.92 %. To ensure that none of the selected features negatively impacted the classification, we incrementally removed the features with the lowest importance and measured the classification performance. As a result, we found that using the top 14 or 15 most important features yielded the highest accuracy of 99.96 % for identifying the video source. Table 6 summarizes the classification performance of the ExtraTrees model based on the number of features used.

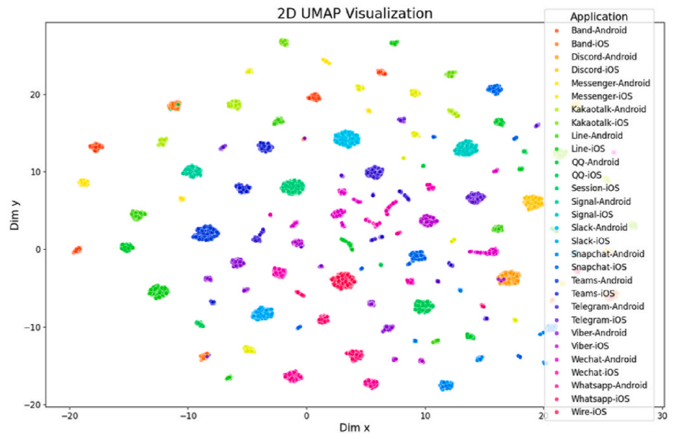


Fig. 5. 2D UMAP visualization illustrating the effectiveness of the selected features in differentiating between application sources.

5.4. *Effectiveness of selected features*

To validate the effectiveness of our selected features, we employed t-Distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP) for visualizing the feature relationships. These dimensionality reduction techniques allow us to observe how well the features differentiate between various application sources. When visualizing the selected features, we further subdivided the total 16 application classes by operating system. The reason for this is that even the same application exhibits differences in feature values across different operating systems. Consequently, as shown in Table 3, we classified the data into a total of 30 distinct classes.

The t-SNE visualization, shown in Fig. 4, demonstrates the clustering of video samples based on their extracted features. Each point represents a video sample, colored according to its source application. The clear separation between clusters indicates that our features effectively capture unique characteristics of each application. This distinct clustering confirms the discriminative power of our feature set, which is crucial for accurate classification.

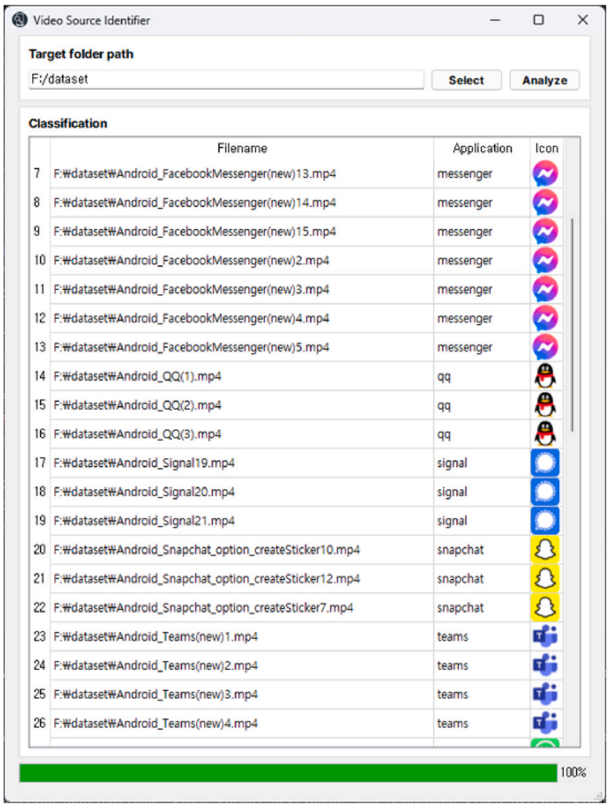
In addition to t-SNE, we applied UMAP to further validate the robustness of our feature set. Fig. 5 presents the UMAP visualization, where each point corresponds to a video sample, with colors indicating the source application. Similar to t-SNE, UMAP reveals distinct clusters, reinforcing the ability of our features to separate different applications effectively. UMAP provides an alternative perspective, complementing the t-SNE results and confirming the robustness of our feature selection process.

The insights from both t-SNE and UMAP visualizations provide strong evidence of our feature extraction method's effectiveness. These visualizations highlight the features' discriminative power and their capability to facilitate accurate classification. By incorporating these dimensionality reduction techniques, we demonstrate the utility of our feature set in distinguishing between different video source applications, thus validating our methodology and reinforcing the reliability of our results.

6. Implementation

In this paper, we developed an automated tool based on the proposed methodology. This tool can aid in the detailed analysis of multimedia during the Analysis phase of the digital forensic process (Kent and Grance, 2006). It is particularly useful for detecting forgery and tampering of video files, as well as identifying their sources, which can be crucial evidence.

Fig. 6 illustrates an example of executing this tool. The tool accepts a directory path containing video files and recursively searches all



Target folder path

F:/dataset

Classification

	Filename	Application	Icon
7	F:₩dataset₩Android_FacebookMessenger(new)13.mp4	messenger	
8	F:₩dataset₩Android_FacebookMessenger(new)14.mp4	messenger	
9	F:₩dataset₩Android_FacebookMessenger(new)15.mp4	messenger	
10	F:₩dataset₩Android_FacebookMessenger(new)2.mp4	messenger	
11	F:₩dataset₩Android_FacebookMessenger(new)3.mp4	messenger	
12	F:₩dataset₩Android_FacebookMessenger(new)4.mp4	messenger	
13	F:₩dataset₩Android_FacebookMessenger(new)5.mp4	messenger	
14	F:₩dataset₩Android_QQ(1).mp4	qq	
15	F:₩dataset₩Android_QQ(2).mp4	qq	
16	F:₩dataset₩Android_QQ(3).mp4	qq	
17	F:₩dataset₩Android_Signal19.mp4	signal	
18	F:₩dataset₩Android_Signal20.mp4	signal	
19	F:₩dataset₩Android_Signal21.mp4	signal	
20	F:₩dataset₩Android_Snapchat_option_createSticker10.mp4	snapchat	
21	F:₩dataset₩Android_Snapchat_option_createSticker12.mp4	snapchat	
22	F:₩dataset₩Android_Snapchat_option_createSticker7.mp4	snapchat	
23	F:₩dataset₩Android_Teams(new)1.mp4	teams	
24	F:₩dataset₩Android_Teams(new)2.mp4	teams	
25	F:₩dataset₩Android_Teams(new)3.mp4	teams	
26	F:₩dataset₩Android_Teams(new)4.mp4	teams	

Fig. 6. Output of the proof-of-concept tool developed based on the proposed methodology for video source identification.

subdirectories for files with.mp4 and.mov extensions. Subsequently, as described in Section 5.3, it extracts the top 14 features that showed the best performance. The extracted features are preprocessed using pre-trained models saved in pickle files, and the source application is subsequently classified using a pre-trained ExtraTrees model, also loaded from pickle files.

To validate the proposed methodology, this tool has been released as an open-source Python project (Yang, H., 2024). In this study, the experimental setup comprised the following environment: Python 3.8.1, scikit-learn 1.2.2, TensorFlow 2.9.1, and Keras 2.9.0. Additionally, a release executable is provided to simplify dependency management.

7. Conclusions and perspectives

This paper described the machine learning-based method for identifying the source application of video files by extracting and utilizing various feature information stored in the video file's storage format and internal metadata. We generated a total of 1974 sample videos for 16 different IMAs, considering all transmission options and encoding settings provided by each IMA. By extracting features that can identify the source from these media files and training various machine learning models, we achieved an identification accuracy of approximately 99.96 %. We developed an automated analysis tool implementing this methodology and made it available as open-source for the forensic community.

The results of this study showed that the video files transmitted by the 16 IMAs contain clearly distinguishable features from each other. However, in real-world digital investigation, we have to deal with a more complex situation: a large number of video files are generated by various hardware, software, and services, including multimedia-related devices such as digital cameras, smartphones, and dashcams. Therefore, in order for the video source identification method proposed in this study to be of practical use in digital forensics, it needs to be continuously improved to identify a wide variety of possible video sources and to clearly report on video files that are not pre-trained and cannot be identified. We plan to further strengthen our implementation through future research efforts.

With the advancement of technology, video-related cybercrimes such as illegal video distribution and copyright infringement are on the rise. Consequently, there is an increasing need to identify the owners of potential evidence files or pinpoint the distributors. As a solution to these issues, we proposed a method to identify the source application through which the video was shared. We hope that we can contribute to the detailed analysis of video files that can serve as crucial digital evidence.

Acknowledgements

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2022-0-00281, Development of digital evidence analysis technique using artificial intelligence technology).

References

Ahmed, F., Khelifi, F., Lawgaly, A., Bouridane, A., 2019. Comparative analysis of a deep convolutional neural network for source camera identification. In: 2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3). IEEE, pp. 1–6.
Aizvorski, 2014. h264bitstream. URL: https://github.com/aizvorski/h264bitstream. (Accessed 15 July 2024).
Akbari, Y., Almaadeed, N., Al-Maadeed, S., Khelifi, F., Bouridane, A., 2022. Prnu-net: a deep learning approach for source camera model identification based on videos taken with smartphone. In: 2022 26th International Conference on Pattern Recognition (ICPR), IEEE, pp. 599–605.
Altinisik, E., Sencar, H.T., 2020. Source camera verification for strongly stabilized videos. IEEE Trans. Inf. Forensics Secur. 16, 643–657.
Altinisik, E., Sencar, H.T., Tabaa, D., 2022. Video source characterization using encoding and encapsulation characteristics. IEEE Trans. Inf. Forensics Secur. 17, 3211–3224.
Amerini, I., Anagnostopoulos, A., Maiano, L., Celsi, L.R., 2021. Learning double-compression video fingerprints left from social-media platforms. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 2530–2534.
Amerini, I., Caldelli, R., Mastio, A.D., Fuccia, A.D., Molinari, C., Rizzo, A.P., 2017. Dealing with video source identification in social networks. Signal Process. Image Commun. 57, 1–7.
Anmol, T., Sitara, K., 2024. Video source camera identification using fusion of texture features and noise fingerprint. Forensic Sci. Int.: Digit. Invest. 49, 301746.
Apple Computer, I, 2001. Quicktime file format specification. URL: https://developer.apple.com/standards/qtff-2001.pdf. (Accessed 15 July 2024).
Bennabhaktula, G.S., Timmerman, D., Alegre, E., Azzopardi, G., 2022. Source camera device identification from videos. SN Computer Science 3, 316.
Bitmovin, 2023. Video developer servey 2023. URL: https://bitmovin.com/video-developer-report. (Accessed 15 July 2024).
Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. Smote: synthetic minority over-sampling technique. J. Artif. Intell. Res. 16, 321–357.
Chen, M., Fridrich, J., Goljan, M., Lukas, J., 2008. Determining image origin and integrity using sensor noise. IEEE Trans. Inf. Forensics Secur. 3, 74–90.
Dietterich, T.G., 2000. Ensemble methods in machine learning. In: International Workshop on Multiple Classifier Systems. Springer, pp. 1–15.
Dirik, A.E., Sencar, H.T., Memon, N., 2008. Digital single lens reflex camera identification from traces of sensor dust. IEEE Trans. Inf. Forensics Secur. 3, 539–552.
FFmpeg, 2000. FFmpeg. URL: https://ffmpeg.org/. (Accessed 15 July 2024).
Flor, E., Aygun, R., Mercan, S., Akkaya, K., 2021. Prnu-based source camera identification for multimedia forensics. In: 2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI). IEEE, pp. 168–175.
Galdi, C., Hartung, F., Dugelay, J.L., 2019. Socrates: a database of realistic data for source camera recognition on smartphones. In: ICPRAM, pp. 648–655.
Hamdi, D., Iqbal, F., Baker, T., Shah, B., 2016. Multimedia file signature analysis for smartphone forensics. In: 2016 9th International Conference on Developments in eSystems Engineering (DeSE). IEEE, pp. 130–137.
Hosler, B.C., Zhao, X., Mayer, O., Chen, C., Shackleford, J.A., Stamm, M.C., 2019. The video authentication and camera identification database: a new database for video forensics. IEEE Access 7, 76937–76948.
Huamán, C.Q., Orozco, A.L.S., Villalba, L.J.G., 2020. Authentication and integrity of smartphone videos through multimedia container structure analysis. Future Generat. Comput. Syst. 108, 15–33.
International Telecommunication Union, 2003. Advanced video coding for generic audiovisual services H, 264. URL: https://www.itu.int/rec/T-REC-H.264. (Accessed 15 July 2024).

International Telecommunication Union, 2013. High efficiency video coding. URL: https://www.itu.int/rec/T-REC-H.265. (Accessed 15 July 2024).

ISO/IEC, 2020. Information technology – coding of audio–visual objects – part 14: mp4 file format. URL: https://www.iso.org/standard/79110.html. (Accessed 15 July 2024).

Kent, Karen, S.C., Grance, T., 2006. Guide to integrating forensic techniques into incident. Tech. Rep. 800–886.

Kouokam, E.K., Dirik, A.E., 2019. Prnu-based source device attribution for youtube videos. Digit. Invest. 29, 91–100.

Lawgaly, A., Khelifi, F., 2016. Sensor pattern noise estimation based on improved locally adaptive dct filtering and weighted averaging for source camera identification and verification. IEEE Trans. Inf. Forensics Secur. 12, 392–404.

Lee, K., Choi, J., Park, J., Lee, S., 2021. Your car is recording: metadata-driven dashcam analysis system. Forensic Sci. Int.: Digit. Invest. 38, 301131.

Li, G., Zhao, X., Cao, Y., 2023. Forensic symmetry for deepfakes. IEEE Trans. Inf. Forensics Secur. 18, 1095–1110.

López, R.R., Luengo, E.A., Orozco, A.L.S., Villalba, L.J.G., 2020. Digital video source identification based on container's structure analysis. IEEE Access 8, 36363–36375.

MediaArea, 2002. MediaInfo. URL: https://mediaarea.net/en/MediaInfo. (Accessed 15 July 2024).

Orozco, A.L.S., Huamán, C.Q., Álvarez, D.P., Villalba, L.J.G., 2020. A machine learning forensics technique to detect post-processing in digital videos. Future Generat. Comput. Syst. 111, 199–212.

Park, J., Lee, S., 2014. Data fragment forensics for embedded dvr systems. Digit. Invest. 11, 187–200.

Reis, P.M.G.I., Ribeiro, R.O., 2024. A forensic evaluation method for deepfake detection using dcnn-based facial similarity scores. Forensic Sci. Int. 358, 111747.

de Roos, L., Geradts, Z., 2021. Factors that influence prnu-based camera-identification via videos. Journal of Imaging 7, 8.

Shullani, D., Fontani, M., Iuliani, M., Shaya, O.A., Piva, A., 2017. Vision: a video and image dataset for source identification. EURASIP J. Inf. Secur. 1–16.

Wong, T.T., Yeh, P.Y., 2019. Reliable accuracy estimates from k-fold cross validation. IEEE Trans. Knowl. Data Eng. 32, 1586–1594.

Xiang, Z., Horváth, J., Baireddy, S., Bestagini, P., Tubaro, S., Delp, E.J., 2021. Forensic analysis of video files using metadata. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, pp. 1042–1051.

Yang, P., Baracchi, D., Iuliani, M., Shullani, D., Ni, R., Zhao, Y., Piva, A., 2020. Efficient video integrity analysis through container characterization. IEEE Journal of Selected Topics in Signal Processing 14, 1–16.

Yang, H., 2024. Video source identifier. URL: https://github.com/yhyoy/video_source_identifier. (Accessed 15 July 2024).