Digital Investigation 11 (2014) S68–S76

Contents lists available at ScienceDirect

Digital Investigation

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

Forensic analysis of video file formats

Thomas Gloe [a,*], André Fischer [a], Matthias Kirchner [b]

[a] dence GmbH, Dresden, Germany
[b] University of Münster, Department of Information Systems, Münster, Germany

Keywords:
Digital forensics
File format
Metadata
Video
Authentication

A B S T R A C T

Video file format standards define only a limited number of mandatory features and leave room for interpretation. Design decisions of device manufacturers and software vendors are thus a fruitful resource for forensic video authentication. This paper explores AVI and MP4-like video streams of mobile phones and digital cameras in detail. We use customized parsers to extract all file format structures of videos from overall 19 digital camera models, 14 mobile phone models, and 6 video editing toolboxes. We report considerable differences in the choice of container formats, audio and video compression algorithms, acquisition parameters, and internal file structure. In combination, such characteristics can help to authenticate digital video files in forensic settings by distinguishing between original and post-processed videos, verifying the purported source of a file, or identifying the true acquisition device model or the processing software used for video processing.

Introduction

Methods to verify the authenticity of media data are of growing relevance in our digital world. While most consumer devices lack practical authentication support at all, attacks against professional camera authentication systems have demonstrated weaknesses of existing in-device solutions.[1] Forensic techniques to infer the provenance and the processing history of media files *ex post* have thus gained more and more interest among researchers and practitioners. Forensic image analysis has been the main driver of the field (Sencar and Memon, 2013), but also video files have recently been brought to the forefront (Milani et al., 2012b). Resembling the evolution of digital image forensics, an already ample body of literature approaches the problem of video forensics through the analysis of inherent device characteristics or processing artifacts in the video data (Chen et al., 2007; Wang and Farid, 2007; Hsu et al., 2008; Conotter et al., 2011; Stamm et al., 2012; Vázquez-Padín et al., 2012, amongst others).

File format information and *metadata* are another source of forensic evidence, but have generally received less attention. Existing works mainly focus on digital images. Here, basic JPEG (ISO/IEC 10918-1, ITU-T Recommendation T.81, 1992) and EXIF (Japan Electronics and Information Technology Industries Association, 2002) metadata properties have gained major interest. Digital cameras and image processing software (or groups thereof) use customized quantization tables. Differences therein can narrow down the source device of a questioned image (Farid, 2008; Kornblum, 2008). Characteristics of thumbnail images (often saved as JPEG images themselves) have been reported to be another pool of forensically relevant features (Murdoch and Dornseif, 2004; Kee and Farid, 2010). In one of the most elaborate approaches, Kee et al. (2011) combine image and thumbnail compression parameters, image and thumbnail dimensions and the number of EXIF entries into signatures of camera model or processing software configurations. By testing against a reference database, images of unknown or uncertain provenance can be attributed to a

* Corresponding author.
E-mail addresses: thomas.gloe@dence.de (T. Gloe), matthias.kirchner@uni-muenster.de (M. Kirchner).
[1] http://www.elcomsoft.de/nikon.html http://www.elcomsoft.de/canon.html.

http://dx.doi.org/10.1016/j.diin.2014.03.009

class of source configurations. Images are flagged as suspicious if no match is found. In a different approach, Fan et al. (2013) exploit noise characteristics to determine whether image content and EXIF data are consistent. However, as tampering with compression parameters or EXIF entries is only a question of using proper software tools (which are often publicly available), concerns have frequently been expressed that high-level file format information and metadata are easily replaceable and/or forgeable (Sencar and Memon, 2008). On the contrary, existing image processing software and metadata editors do not allow users to access or to modify *core file structures*. Along these lines, Gloe (2012) reports that peculiarities in the specific internal order of JPEG and EXIF structures are particularly valuable and distinctive information for digital image authentication. Such low-level characteristics thus offer a much increased reliability and relax—to some degree—the common assumption that file format and metadata information shall not be trusted *per se*.

Following this trail, this paper extends the idea of file format forensics to popular digital video data container formats, for which—to the best of our knowledge—no systematic exploration of file format and/or metadata specifics has been reported in the forensic literature so far. Similar to differences in the JPEG file structure, we identify manufacturer- and model-specific video file format characteristics and point to traces left by processing software. Such traces can be used to authenticate digital video streams and to attribute recordings of unknown or questionable provenance to (groups of) video camera models. Note that this differs from video file carving (Pal and Memon, 2009; Lewis, 2012), where it is usually sufficient to find *valid* video streams in (fragmented) mass storage dumps. Based on a description of our (Test setup) and (General observations) on video file format forensics, the following sections demonstrate that peculiarities of the (AVI Container format), of (Quicktime and related container formats (MP4, 3GP)), and of (MJPEG Compression parameters) can yield important insights about provenance and processing history of digital videos. The paper closes with a (Summary and concluding remarks).

Test setup

We report findings from the examination of each one device of overall 19 digital camera models and 14 mobile phone models, all of them equipped with video capturing functionality. We acquired 3 to 14 videos per device by iterating over all available video quality settings (e.g., frame size and frame rate). Mobile phones were also switched between regular and MMS (Multimedia Message Service) mode where available. All devices were subject to slight motion during the video capturing process. Table 1 summarizes our test setup.

For a selected number of camera models, video editing software was used to cut short sequences (length: 10 s) from the recorded video streams. All software in our tests supports non-intrusive ('lossless') video editing, i.e., we saved files without re-compressing the original stream. Hence, the edited videos are presumably not detectable by means of double compression artifacts (Wang and Farid, 2006; Milani et al., 2012a). The 'Adobe Premiere' toolbox was a sole exception in our test set in this regard. The commercial software is one of the major professional video editing tools, but does not support lossless processing.

We have written our own customized file parser(s) to read and extract *all* available file format information and metadata from the videos in our database.[2] As it is impossible to detail all model- or vendor-specific singularities within the scope of this paper, the following sections focus on selected results and observations that we believe are particularly relevant for practical forensic analyses of common video container formats.

General observations

The majority of *digital cameras* in our database stores videos in the AVI container format. Only a few of the test devices use Apple Quicktime MOV containers. We found that most digital cameras compress video data using Motion JPEG (MJPEG), where every video frame is handled as independently JPEG-compressed image. Only three camera models in our sample use more sophisticated and efficient compression algorithms (DivX, Xvid or H.264). Before compression, frames are generally converted to the YUV color space. We encountered 4:2:2 and 4:2:0 subsampling to reduce the resolution of chroma channels. MJPEG compressed video streams utilize the full intensity range of 256 intensity levels for 8-bit encoded frames (yuvj422p or yuvj420p), whereas camera models with DivX or Xvid support (yuv420p) use only a reduced number of 220/225 intensity levels for the luminance/chrominance channel(s) (ITU-R Recommendation BT.601-7, 2011). Audio data in the video container is usually stored as raw data (PCM), using linear (8-/16-bit) or logarithmic (μ-law) quantization.

All *mobile phones* in our database store video data in MOV-based container formats (MOV, 3GP, MP4). The LG KU990 camera phone is an exception and also supports AVI containers. Interestingly, none of the mobile phones uses MJPEG compression. Instead, more sophisticated compression algorithms find application, e.g., H.263, H.264, MPEG-4 video (simple profile) or DivX. Subsampling always follows a 4:2:0 scheme. In contrast to videos from digital cameras, the audio track of mobile phone videos is typically also subject to lossy compression. We found MPEG-based audio compression or the Adaptive Multi-Rate audio codec (AMR-NB) to be most common. The latter is a standard optimized for speech coding (3rd Generation Partnership Project, 1999), which is very common in mobile phones designed for GSM- and UMTS-networks.

AVI Container format

Microsoft introduced AVI (Audio Video Interleave) in 1992 as a multimedia container format, which can contain

[2] Also the free software `exiftool` (available at: http://www.sno.phy.queensu.ca/~phil/exiftool) can be used to extract metadata and high-level file format information, but it does not provide access to all information that is of interest here.

Table 1
Digital cameras, mobile phones and video editing software used in this study.

Make	Model	Container	Video stream	Audio stream
Digital camera models				
Agfa	DC-504, DC-733s, DC-830i, Sensor530s	AVI	MJPEG (yuvj422p)	PCM (8-bit)
Agfa	Sensor505-X	AVI	MJPEG (yuvj422p)	PCM (μ-law)
Canon	S45, S70, A640, Ixus IIs	AVI	MJPEG (yuvj422p)	PCM (8-bit)
Canon	EOS-7D	MOV	H.264 (yuvj420p)	PCM (16-bit)
Casio	EX-M2	AVI	MJPEG (yuvj420p)	PCM (8-bit)
Kodak	M1063	MOV	MJPEG (yuvj420p)	PCM (μ-law)
Minolta	DiMAGE Z1	MOV	MJPEG (yuvj422p)	PCM (8-bit)
Nikon	CoolPix S3300	AVI	MJPEG (yuvj422p)	PCM (16-bit)
Pentax	Optio A40	AVI	DivX (yuv420p)	MP2
Pentax	Optio W60	AVI	MJPEG (yuvj422p)	PCM (8-bit)
Praktica	DC2070	MOV	MJPEG (yuvj420p)	–
Ricoh	GX100	AVI	MJPEG (yuvj422p)	PCM (μ-law)
Samsung	NV15	AVI	Xvid (yuv420p)	PCM (μ-law)
Mobile phone models				
Apple	IPhone 4	MOV	H.264 (yuv420p)	MP4A
Benq Siemens	S88	3GP	H.263 (yuv420p)	AMR-NB
BlackBerry	8310	3GP	MP4V, H.263 (yuv420p)	AMR-NB
Google	Nexus 7	3GP	H.264 (yuv420p)	MP4A
LG	KU990	3GP, AVI	DivX, MP4V, H.263 (yuv420p)	MP3, AMR-NB
Motorola	MileStone	3GP	MP4V (yuv420p)	MP4A
Nokia	6710	3GP, MP4	MP4V, H.263 (yuv420p)	MP4A, AMR-NB
Nokia	E61i	3GP, MP4	MP4V, H.263 (yuv420p)	MP4A, AMR-NB
Nokia	E65	3GP, MP4	MP4V, H.263 (yuv420p)	MP4A, AMR-NB
Nokia	X3-00	3GP	MP4V, H.263 (yuv420p)	AMR-NB
Palm	Pre	MP4	MP4V (yuv420p)	MP4A
Samsung	GT-5500i	MP4	MP4V, H.263 (yuv420p)	AMR-NB
Samsung	SGH-D600	MP4	MP4V (yuv420p)	MP4A
Sony Ericsson	K800i	3GP	H.263 (yuv420p)	AMR-NB
Software	Version			
FFmpeg	1.1.4	all		
Avidemux	2.6.2	all		
FreeVideoDub	2.0.17.320	all		
VirtualDub	1.9.11	AVI		
Yamb	2.1.0.0 beta2	no AVI		
Adobe Premiere	CS 5			

both video and audio streams (Microsoft Developer Network).

General file structure

AVI files start with a mandatory AVI RIFF (Resource Interchange File Format) header. All following data is organized and stored in so-called *lists* and *chunks*. A four character code (FourCC) is used to identify these data segments. The FourCC for a list is `LIST`. Different FourCC's exist for chunks, e.g., `JUNK` or `idx1`. There is no strict specification that defines the sequence and occurrence of lists and chunks.

Fig. 1 illustrates the basic file structure that we found to appear similarly in all examined AVI files captured by digital cameras or mobile phones. All devices store the mandatory list `hdrl` directly after the AVI RIFF header. This list segment comprises all the information that is necessary to decompress the video and audio data stored in the file. The third mandatory AVI segment after header and `hdrl` list is the `movi` list. It contains the actual video and audio data. Depending on the specific camera or phone model, additional lists and/or `JUNK` chunks may exist between the `hdrl` and `movi` lists. These optional data segments are either used for padding or to store metadata. The `idx1` chunk indexes the data chunks and their location in the file. It is mentioned as an optional element in the AVI file reference (Microsoft Developer Network, 0000), yet we found it in all video files in our database.

Camera model specifics

An inspection of our video database revealed considerable differences between device models and software vendors, a selection of which we discuss in the following. These and related characteristics can be tested for to verify or to infer the source of questioned video content.

Fig. 2 exemplarily illustrates four typical AVI file structures of video recordings from Canon A640, Canon S45, Nikon CoolPix S3300, and Ricoh GX100 digital cameras. The tree-like representations reflect the nested character of the

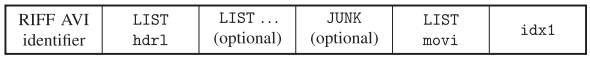

Fig. 1. Internal structure of AVI files acquired with our cameras.

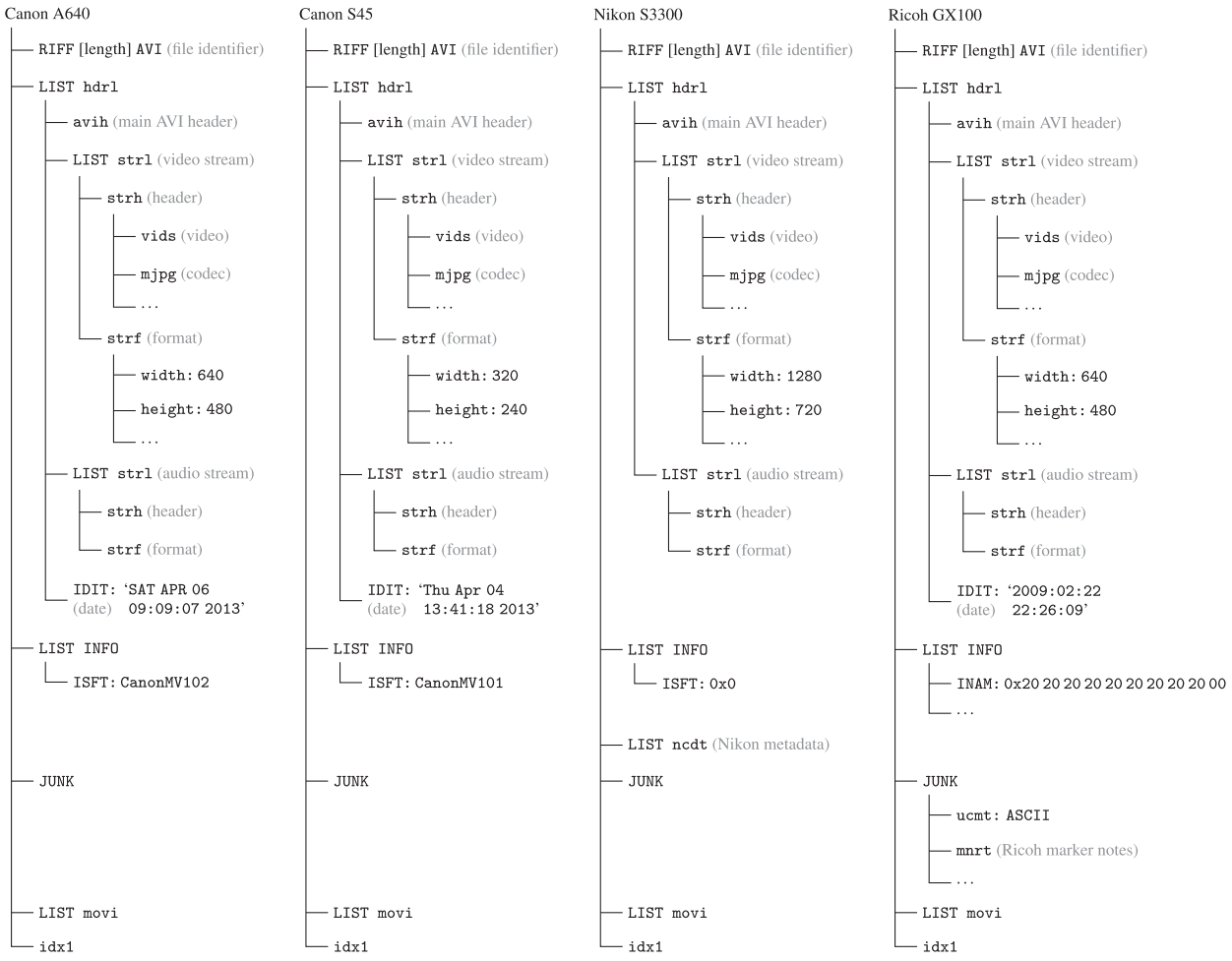

Fig. 2. AVI file structure of videos acquired with Canon A640, Canon S45, Nikon CoolPixS3300, and Ricoh GX100 digital cameras. Equivalent elements are arranged on the same horizontal level. Not all nested sub-elements are listed.

AVI container format and depict all major lists and chunks in accordance to their actual order in the respective files. Corresponding elements are arranged on the same horizontal level. The figure indicates that not all files share the same components, and that both the content and the format of specific chunks may vary. The Nikon camera, for instance, does not use the `IDIT` chunk to store the recording date, but adds a dedicated `ncdt` list to organize its metadata. All four cameras insert an `INFO` list with distinct content after the stream header. The two Canon cameras further differ in the format of the `IDIT` date specification, but none of them uses the same format as the Ricoh GX100.

We also found that digital cameras with MJPEG compression may specify the video codec in lowercase or uppercase letters. Canon Ixus IIs, Canon PowerShot (A640, S45, S70), Nikon CoolPix, Optio W60, Ricoh GX100, and CASIO EX-M2 cameras use '`mjpg`', whereas all other tested cameras prefer the string '`MJPG`'. Further variations exist in the `hdrl` list. The Agfa Sensor505-X camera adds an additional stream description that refers to the `VendorName`. The Pentax Optio A40 camera explicitly specifies the `StreamName` ('`Video`' or '`Audio`'). Video and audio stream descriptions in `hdrl` lists from LG KU990 mobile phones are followed by `JUNK` chunks. Similar to the examples in Fig. 2, most camera models add their own `INFO` list after the `hdrl` list to provide additional metadata. Also here, padding with `JUNK` data chunks is common.

Video editing

All software tools in our test setting leave distinct traces in the file structure of edited AVI videos, which do not match any of the camera characteristics in our database. Even losslessly edited videos are thus detectable by comparing the file structure of a questioned file with a reference of the purported source.

Fig. 3 gives two specific examples. The familiar tree-like graphs depict AVI file structures of a Canon A640 video after editing with the tools AVIDemux and VirtualDub, respectively. Differences to the structure of the original file (shown in the leftmost part of Fig. 2) are printed on gray background. Observe that both tools place `JUNK` chunks in the `hdrl` list at the end of audio and video stream

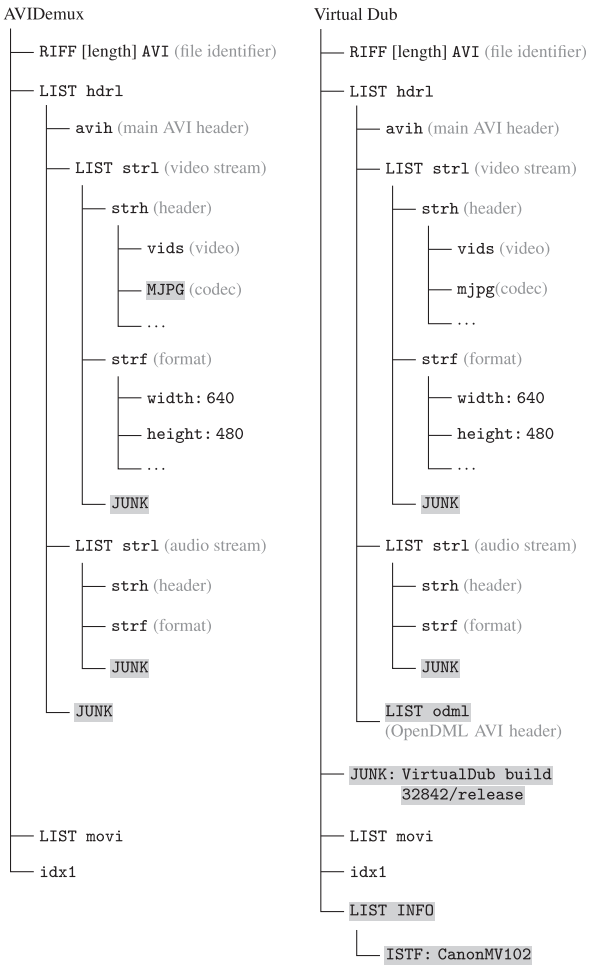

Fig. 3. AVI file structure of a Canon A640 video after editing with AVIDemux and VirtualDub. Equivalent elements are arranged on the same horizontal level. Not all nested sub-elements are listed. Differences to the original file are printed on gray background (see also Fig. 2).

descriptions. VirtualDub adds another `LIST` with OpenDML[3] AVI header data. The same tool extends the top-level file hierarchy by its own identifier `JUNK` chunk (‘`VirtualDub build 32842/release`’). AVIDemux changes the video codec information to capitalized letters. Other editing softwares leave their own traces. FFmpeg, for instance, also inserts additional `JUNK` chunks. The content of these entries is fixed and specific to FFmpeg. In addition, a list `info` with details about the employed encoder software and version (here Lavf54.59.106) is present in videos edited with FFmpeg.

Metadata lists and `JUNK` chunks that would hint to the recording device are typically lost in edited files. VirtualDub is an exception and appends the device-specific `INFO` list of the original file to the edited video (cf. Fig. 3). In general, however, losslessly edited video streams might still contain distinct compression characteristics of the source camera, as we will discuss in the section on (MJPEG Compression parameters).

[3] OpenDML is an AVI-like file format specification that supports larger file sizes, amongst other extensions.

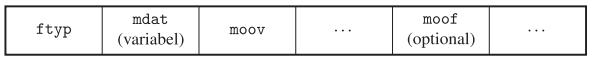

Fig. 4. Internal structure of most MP4-like files acquired with our cameras.

QuickTime and related container formats (MP4, 3GP)

The MOV container format was introduced by Apple for its QuickTime multimedia framework in 1991. It was later used as a basis for the MP4 and 3GP container specifications (Apple Computer, Inc., 2001; ISO/IEC 14496, 2003; 3rd Generation Partnership Project, 2004). For the sake of simplicity, we refer to all three container formats as *MP4-like formats*.

General file structure

Similar to the AVI format, MP4-like containers consist of individual data structures (‘atoms’ or ‘boxes’), which are identified via unique 4-byte sequences. Information about the atom size precedes the corresponding identifier. Atoms can be nested. Although the format explicitly does not define any particular order or number of data segments, we found the general structure in Fig. 4 to be present in most of the examined files. MP4-like video files usually start with the `ftyp` atom, which refers to the file type specifications the file is compatible with.[4] The actual data stream is stored in the `mdat` atom, which is accompanied by corresponding metadata in the `moov` atom. We also encountered files with `moof` atoms, which contain shorter data chunks of elementary streams.

Camera model specifics

Reflecting the by far more complex file format, MP4-like videos exhibit an even larger degree of source-dependent internal variations than AVI files.[5] This is only to the advantage of forensic investigators, who will find these characteristics useful for authentication purposes. Yet it is beyond the scope of this paper to detail all the differences we observed, and we rather focus on select indicative examples.

Most MP4-like videos in our database start with the `ftyp` atom, yet we also found a number of exceptions. Kodak videos, for instance, start with a `skip` atom, whereas Minolta Z1 videos place a `pnot` atom with a reference to preview data at the beginning of a file. The Praktica DC2070 camera does not use a header at all and starts with the `mdat` atom directly. As MP4-like container formats are compatible with a plethora of different data formats, codecs and parameterizations, the `ftyp` atom—if

[4] As per the 2008 ISO media file standard ISO/IEC 14496 (2008), the `ftyp` atom is mandatory and must be placed as early as possible in the file.

[5] We refrain from detailed graphical illustrations because of this increased complexity.

Table 2
Major and compatible brands stored in the MP4 `ftyp` atom.

Model/**container**: model	Major brand	Compatible brands
Apple IPhone 4	qt	qt
Benq S88	isom	isom, 3g2a
BlackBerry 8310, Palm Pre	3gp4	3gp5, 3gp4, isom
Canon 7D	qt	qt, CAEP
Google Nexus 7	3gp4	isom, 3gp4
Kodak M1063	–	–
LG KU990	3gp5	3gp5, 3gp4
LG KU990, Samsung SGH-D600	3gp5	3gp5, isom
Minolta DiMAGE Z1	–	–
Motorola MileStone Samsung GT-5500i (MP4V)	3gp4	3gp4, mp41, 3gp6
3GP: Nokia 6710, E61i, E65	3gp4	3gp4, 3g2a, isom
MP4: Nokia 6710, E61i, E65	mp42	mp42, 3gp4, isom
Nokia X3-00	3gp5	3gp5, 3gp4, 3g2a, isom
Praktica DC2070	–	–
Samsung GT-5500i (H.263)	3gp4	3gp4, 3gp6
SonyEricsson K800i	3gp5	vfj1, 3gp4, 3gp5, mp42
FFmpeg	isom	isom, iso2, mp41
YAMB	mp42	isom, mp42, 3gp5
Adobe Premiere CS 5	3gp5	isom, 3gp4, mp41, mp42

present—states which specification is the 'best use' of the file in the `major_brand` field. Additional fields then list the minor version and compatible brands. Table 2 reports combinations of `ftyp` major and compatible brand specifications in our database. It indicates that these specifications can differ to a large degree between different recording devices. We do not report the `minor_version` field, as we usually found it set to `0`.

Table 3
Values stored in the MP4 `moov` atom `time_scale` field, as observed by iterating over all available quality settings per device. Symbol '.' indicates that `mvhd` and `mdhd` entries do not differ.

Model	`time_scale`		
	`mvhd`	`mdhd` (video)	`mdhd` (audio)
Benq S88	90000	.	8000
BlackBerry 8310	1000	.	1000
Canon 7D	25000, 24000, 50000	.	48000
Google Nexus 7	1000	90000	44100
IPhone 4	600	.	44100
Kodak M1063	150191–279991	.	11025
LG KU990	1000	.	8000,1000
Minolta Z1	600	.	7875
Motorola MileStone	1000	.	44100
Nokia 6710	10000	30000	**3GP:** 8000, **MP4**: 48000, 16000
Nokia E61i, E65	10000	30000	**3GP**: 8000, **MP4**: 16000
Nokia X3-00	15750	.	15750
Palm Pre	1000	.	44100
Praktica DC2070	60	.	–
Samsung GT-5500i	1000	.	8000
Samsung SGH-D600	1000	.	11025
SonyEricsson K800i	1000	.	8000
FFmpeg	1000	variable	48000
YAMB	600	30000	48000
Adobe Premiere CS5	90000	29970	32000

Table 4
Additional atoms in MP4-like files.

Model	Atoms
Benq S88	`mvex`, `mdat` file end, `moof`
BlackBerry 8310	
Canon 7D	`udta`
Google Nexus 7	`udta`
IPhone 4	`wide`, `free`, `meta`
Kodak M1063	`skip`, `edts`
LG KU990	
Minolta Z1	`pnot`, `PICT`
Motorola MileStone	`udta`
Nokia 6710	
Nokia E61i, E65	
Nokia X3-00	`free`, `mdat` file end
Palm Pre	`udta`
Praktica DC2070	
Samsung GT-5500i	`udta`
Samsung SGH-D600	`iods`
SonyEricsson K800i	`udta`
FFmpeg	`free`, `edts`, `udta`
YAMB	`iods`, `tref`, `nmhd`, `free`, `mdat` file end
Adobe Premiere CS5	`iods`, `udta`, `uuid`, `mdat` file end

The `moov` atom is one of the most complex data structures in MP4-like files. It specifies the decoding parameters for parsing the `mdat` data stream correctly. The atom itself is organized in a number of sub-structures, such as the general `mvhd` movie header atom or the more specific `mdhd` media header atom. We found that parameter settings largely depend on the specific camera model. Table 3 exemplarily summarizes settings for the `time_scale` field, which controls both frame rate and audio sampling rate. Observe that `mvhd` and `mdhd` atom may define different values for this field, and that corresponding combinations thereof vary between different recording devices.

Besides the (quasi-)standard `ftyp`, `mdat` and `moov` atoms, we also encountered a variety of additional elements in MP4-like video files. The overview in Tab. 4 indicates, for instance, that some sources store metadata in `udta` user data atoms, or use `free` atoms for padding. Because the specific order of atoms is generally not explicitly defined, it is finally not surprising that there exist differences here, too. Notable deviations are Benq S88 and Nokia X3-00 videos. The former follow a `ftyp moov mdat` sequence (and optionally multiple subsequent pairs of `moof mdat`). The latter are organized as `ftyp moov free mdat free`.[6] Motorola MileStone exhibit another interesting peculiarity. Here, the audio stream precedes the video stream.

Video editing

Like in the case of AVI editing, none of the MP4-like files produced by editing software in our test set is compatible

[6] On a side note, we mention that the apparent prevalence of the `mdat moov` order may pose a problem to forensic file carving scenarios. If the last part of an MP4 file (including the `moov` atom) is lost, its reconstruction in the absence of metadata becomes considerably more complicated. This is generally not the case for AVI file containers or JPEG files.

Table 5
Common JPEG file markers. Symbol '×' indicates mandatory markers. (JIF's entropy encoding table is not restricted to DHT.)

Marker id	Short value	JIF	JFIF	EXIF	Description
SOI	0xFF D8	×	×	×	Start of image
APPn	0xFF En				Application data
APP0	0xFF E0		×		(e.g., JFIF)
APP1	0xFF E1			×	(e.g., EXIF)
DQT	0xFF DB	×	×	×	Quant. tables
DHT	0xFF C4	(×)	×	×	Huffman tables
SOF	0xFF Cn	×			Start of frame
SOF	0xFF C0		×	×	(baseline DCT)
SOS	0xFF DA	×	×	×	Start of scan
DRI	0xFF DD				Restart interval
RSTn	0xFF Dn				*n*th restart
COM	0xFF FE				Comment
EOI	0xFF D9	×	×	×	End of image

with any recording device in our database. The three bottommost rows in Tables 2–4 exemplify some of the differences for FFmpeg, YAMB, and Adobe Premiere CS5. Observe that each software has its own unique signature, which emphasizes the usefulness of such characteristics in authentication scenarios.

Table 2 suggests that the combination and order of major and compatible brands is particularly indicative. The inspection of the time_scale field revealed that FFmpeg, AVIDemux and FreeVideoDub always set the mvhd value to 1000, with a mdhd (video) value depending on the original frame rate. Other tools employ different specifications (cf. Table 3). Table 4 indicates that Samsung SGH-D600 videos share the existence of additional iods object descriptor atoms with edited YAMB and Adobe Premiere videos, whereas the latter two also use extra tref and uuid atoms, respectively. FFmpeg-based software is the only one in our test set that adds an edts edit list atom to the file. We further observed that AVIDemux sets the acquisition time and modification time of edited files incorrectly to 01-01-1904.

MJPEG Compression parameters

Independent of the supported container format, the majority of digital cameras in our database uses Motion JPEG (MJPEG) to encode recorded videos (cf. Table 1). Individual MJPEG frames are stored as JPEG compressed still images, which itself consist of marker segments. Such segments are internally identified by 2-byte numbers, usually abbreviated with character sequences known from the JPEG standard (cf. Tab. 5). Based on our prior work on JPEG file forensics (Gloe, 2012), we expect that the existence and the order of specific segments here depends on the recording device, too.

Table 7
Number of unique quantization tables in MJPEG videos.

Camera model	Y/CbCr
Agfa DC-504	1/1
Agfa DC-733s	589/390
Agfa DC-830i	489/314
Agfa Sensor505-X	893/286
Agfa Sensor530s	1/1
Canon Ixus IIs	5/5
Canon A640	6/6
Canon S45	6/6
Canon S70	8/8
Casio EX-M2	121/121
Kodak M1063	10/10
Minolta DiMAGE Z1	13/13
Nikon CoolPix S3300	465/111
Pentax Optio W60	73/73
Praktica DC2070	1/1
Ricoh GX100	924/338 (2×)
∑ unique quantization tables[a]	2914/1279

[a] The total number of unique tables is not the sum of unique tables per camera model.

For all MJPEG videos in our test set, we observed the same sequence of JPEG marker segments for all frames of one video. Differences do exist between groups of camera models. Within these groups, the sequence of JPEG marker segments is independent of respective quantization settings. Table 6 summarizes our findings for cameras with MJPEG support and the corresponding thumbnails (if available). Interestingly, MJPEG frames do not strictly follow the JFIF (Hamilton, 1992) or JPEG/Exif (Japan Electronics and Information Technology Industries Association, 2002) standards. Most cameras identify JPEG frames by means of an AVI application data segment (APP0(AVI1)) instead. Also the selection of JPEG marker segments—and in some cases even their sequence—is different from the JPEG still images of the same digital camera.

Kodak M1063 video frames, for instance, contain a combination of AVI and JFIF application data segments,

Table 6
JPEG marker segment sequences in MJPEG compressed videos.

Model/**thumbnail**: model	Sequence of JPEG marker segments
Agfa DC-504, Sensor530s	SOI, DQT, SOF0, DHT, COM, SOS, EOI
Agfa DC-733s, DC-830i	SOI, APP0(AVI1), DQT, DHT, SOF0, SOS, EOI
Agfa Sensor505-X, Nikon CoolPix S3300	SOI, APP0(AVI1), DRI, DQT, DHT, SOF0, SOS, EOI
Canon PowerShot A640	SOI, APP0(AVI1), DRI, DQT, SOF0, SOS, EOI
Canon S45, S70, Ixus IIs	SOI, APP0(AVI1), DRI, DQT, SOF0, APP2, SOS, EOI
Casio EX-M2, Ricoh GX100	SOI, APP0(AVI1), DQT, SOF0, SOS, EOI
Kodak M1063	SOI, APP0(AVI1), DRI, APP0(JFIF), DQT, DQT, SOF0, DHT, DHT, DHT, DHT, SOS, EOI
Minolta DiMAGE Z1	SOI, DHT, DHT, DHT, DHT, DQT, DQT, SOF0, SOS, EOI
Pentax Optio W60	SOI, APP0(AVI1), DRI, DQT, SOF0, DHT, SOS, EOI
Praktica DC2070	SOI, APP1(0x0000 mjpg), DQT, DHT, SOF0, SOS, EOI
thumbnail: Nikon CoolPix S3300	SOI, DQT, DHT, SOF0, SOS, EOI
thumbnail: Pentax Optio W60, Ricoh GX100	SOI, DQT, SOF0, DHT, SOS, EOI

whereas the Praktica DC2070 digital camera uses a specific `APP1(0x0000 mjpg)` marker segment. Differences also exist in the organization of Huffman and quantization tables. Older Canon camera models (S45, S70, Ixus IIs) use the `APP2` marker instead of the dedicated `DHT` marker segment to store the Huffman tables for entropy coding. Canon A640, Casio EX-M2, and Ricoh GX-100 video frames rely on standardized Huffman tables and do not store tables along with the frames. On the contrary, Kodak M1063 and Minolta Z1 cameras employ four `DHT` marker segments, i. e., one segment per Huffman table instead of a single `DHT` segment with all four tables. The same camera models use two `DQT` segments instead of one to store the two required quantization tables. Another characteristic of Minolta videos are padding bits between subsequent frames.

It is also interesting to note that we identified the impressive number of *2914 unique JPEG quantization tables* (luminance only) in our relatively small set of test videos. We refer to Table 7 for an overview. This large amount can be mainly attributed to cameras with scene-dependent adaptive quantization settings. We found that these cameras may also switch quantization tables between frames of the same video. Backed with similar experiences with adaptive quantization tables of thumbnail images in the Dresden Image Database (Gloe, 2012), our observations for MJPEG video frames strongly emphasize the challenges that forensic investigators have to deal with when attempting to create a comprehensive database of quantization parameters.

We close this section with the remark that marker segment characteristics (and quantization tables) are most useful to verify or to determine the source device of an MJPEG video (or a group of devices). Lossless video editing does not modify the internal structure of MJPEG frames. This is different from the analysis of JPEG files, where we reported software-specific traces in the file structure of processed images (Gloe, 2012). At the same time, however, any video editing software is likely to change the structure of the container format in its very own specific way. It is thus still possible to detect lossless video editing based on file structure information, cf. our discussion of (Camera model specifics) of the AVI container format.

Summary and concluding remarks

This paper has presented a first systematic exploration of popular video container formats from a forensic viewpoint. Specifically, we have focused on the internal file structure of AVI and MP4-like (MOV, 3GP, MP4) multimedia containers. Our examination of videos from 19 digital cameras, 14 mobile phones, and various video editing tools indicate a profusion of model- and software-specific peculiarities. The identified characteristics complement the toolbox of forensic investigators and provide valuable clues to verify the authenticity of digital video streams.

Our main findings can be summarized as follows:

- Videos from digital cameras and mobile phones often employ different container formats and compression codecs. Mobile phones opt for sophisticated compression algorithms (MP4V, H.26x). Most digital cameras in our test set prefer a combination of AVI containers and basic MJPEG compression.
- The structure of AVI and MP4-like containers is not strictly defined. We observed considerable differences both in the order and in the presence of individual data segments. AVI files often contain specific `INFO` lists or `JUNK` chunks. MP4-like files may employ various non-standard atoms and different parametrizations of specific atom entries.
- Different camera models implement different JPEG marker segment sequences for MJPEG-compressed video frames. Content-adaptive quantization tables seem to be more frequent than for JPEG images. MJPEG frame and JPEG still image marker sequences and compression settings of the same camera model can be completely disjoint.
- Lossless video editing leaves compression settings of the original video stream untouched, but introduces its very own distinctive artifacts in the structure of container files. While file format peculiarities of the genuine source device are typically lost after video editing, all tested software tools have unique file format signatures throughout our test set.

We note that our test set did not comprise multiple devices per camera model, so we can only surmise that our observations generalize to arbitrary devices of the same model as well. Yet this is to be expected, as our observations resemble similar reports about the structure of JPEG still images (Kee et al., 2011; Gloe, 2012), RAW image formats (Kalms et al., 2012), and MP3 audio files (Böhme and Westfeld, 2005). Because our analysis relies on file structure internals, we are currently not aware of any publicly available software that would allow users to consistently forge such information without advanced programming skills. This makes the creation of plausible forgeries undoubtedly a highly non-trivial undertaking and thereby re-emphasizes that file characteristics and metadata must not be dismissed as unreliable source of evidence for the purpose of file authentication *per se*.

We note that further examinations will have to show how distinctive video file format characteristics are on a larger scale. It is without question that the identified general differences between different recording devices and editing softwares could also be combined with a signature-based quantitative approach as proposed by Kee et al. (2011) for JPEG files. From the viewpoint of practical case work, however, we believe that such “condensed” signatures are never optimal, as they do not exploit all available information. This is particularly so when we consider that video container formats are by far more complex (and less strictly defined) than the JPEG format. At the same time, it is an open question how a more holistic signature would scale with all the unknown file format variations that doubtlessly still exist in the wild, but also to what degree modern streamlined smartphone designs are built around standard open source or vendor-based reference implementations. We can only surmise that a collection of source-specific file parsers, which mimic the careful manual inspection of a forensic investigator, could help in

authentication scenarios. Future work will have to investigate how the specification of such parsers can be automated and how practical this approach is also for large-scale source identification applications.

References

3rd Generation Partnership Project. Mandatory speech CODEC speech processing functions; AMR speech codec; General description, 3GPP TS 26.071; 1999.

3rd Generation Partnership Project. Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP), 3GPP TS 26.244; 2004.

Apple Computer, Inc. Quicktime file format; 2001.

Böhme R, Westfeld A. Feature-based encoder classification of compressed audio streams. Multimed Syst 2005;11:108–20.

Chen M, Fridrich J, Goljan M, Lukáš J. Source digital camcorder identification using sensor photo-response non-uniformity. In: Delp EJ, Wong PW, editors. Proceedings of SPIE: Security and Watermarking of Multimedia Content IX. SPIE Press; 2007. 65051G.

Conotter V, O'Brien J, Farid H. Exposing digital forgeries in ballistic motion. IEEE Trans Inf Forensics Security 2011;7:283–96.

Fan J, Cao H, Kot AC. Estimating EXIF parameters based on noise features for image manipulation detection. IEEE Trans Inf Forensics Security 2013;8:608–18.

Farid H. Digital image ballistics from JPEG quantization: a followup study [Technical Report TR2008–638]. Hanover, NH, USA: Department of Computer Science, Dartmouth College; 2008.

Gloe T. Forensic analysis of ordered data structures on the example of JPEG files. In: IEEE international Workshop on Information Forensics and Security (WIFS). IEEE; 2012. pp. 139–44.

Hamilton E. JPEG file interchange format, Version 1.02; 1992.

Hsu CC, Hung TY, Lin CW, Hsu CT. Video forgery detection using correlation of noise residue. In: IEEE workshop on Multimedia Signal Processing (MMSP). IEEE; 2008. pp. 170–4.

ISO/IEC 10918-1, ITU-T Recommendation T.81. Information technology. Digital compression and coding of continuous-tone still images; 1992.

ISO/IEC 14496. Information technology. coding of audio-visual objects, Part 14: MP4 file format; 2003.

ISO/IEC 14496. Information technology. Coding of audio-visual objects, Part 12: ISO base media file format. 3rd ed. 2008.

ITU-R Recommendation BT.601-7. Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratio; 2011.

Japan Electronics and Information Technology Industries Association. JEITA CP-3451, Exchangeable image file format for digital still cameras: Exif version 2.2; 2002.

Kalms M, Gloe T, Laing C. File forensics for raw camera image formats. In: 6th Conference on Software, Knowledge, Information Management and Applications (SKIMA); 2012.

Kee E, Farid H. Digital image authentication from thumbnails. In: Memon ND, Dittmann J, Alattar AM, Delp EJ, editors. Proceedings of SPIE: Media Forensics and Security II; SPIE Press; 2010. 75410E.

Kee E, Johnson MK, Farid H. Digital image authentication from JPEG headers. IEEE Trans Inf Forensics Security 2011;6:1066–75.

Kornblum JD. Using JPEG quantization tables to identify imagery processed by software. Digit Investig 2008;5:S21–5.

Lewis AB. Reconstructing compressed photo and video data [Technical Report UCAM-CL-TR-813]. Cambridge, United Kingdom: University of Cambridge, Computer Laboratory; 2012.

Microsoft Developer Network. AVI RIFF file reference. http://msdn.microsoft.com/en-us/library/ms779636(VS.85).aspx.

Milani S, Bestagini P, Tagliasacchi M, Tubaro S. Multiple compression detection for video sequences. In: IEEE International Workshop on Multimedia Signal Processing (MMSP); 2012a. pp. 112–7.

Milani S, Fontani M, Bestagini P, Barni M, Piva A, Tagliasacchi M, et al. An overview on video forensics. APSIPA Transactions Signal Information Process 2012b;1:e2.

Murdoch SJ, Dornseif M. Hidden data in internet published documents. In: 21st Chaos communication congress; 2004.

Pal A, Memon N. The evolution of file carving. IEEE Signal Process Mag 2009;26:59–71.

Sencar HT, Memon N. Overview of state-of-the-art in digital image forensics. In: Bhattacharya BB, Sur-Kolay S, Nandy SC, Bagchi A, editors. Algorithms, architectures and information systems security. Statistical Science and Interdisciplinary Research, vol. 3. World Scientific Press; 2008. pp. 325–48 [chapter 15].

Sencar HT, Memon N, editors. Digital image forensics. Springer; 2013.

Stamm MC, Lin WS, Liu KJR. Temporal forensics and anti-forensics for motion compensated video. IEEE Trans Inf Forensics Security 2012;7:1315–29.

Vázquez-Padín D, Fontani M, Bianchi T, Comesaña P, Piva A, Barni M. Detection of video double encoding with GOP size estimation. In: IEEE international Workshop on Information Forensics and Security (WIFS). IEEE; 2012. pp. 151–6.

Wang W, Farid H. Exposing digital forgeries in video by detecting double MPEG compression. In: ACM workshop on multimedia and security. ACM Press; 2006. pp. 37–47.

Wang W, Farid H. Exposing digital forgeries in interlaced and deinterlaced video. IEEE Trans Inf Forensics Security 2007;2:438–49.