PROYECTO FIN DE MÁSTER EN

INGENIERÍA INFORMÁTICA

CURSO 2015-2016

Técnicas Anti-Forenses para Vídeos de Dispositivos Móviles

Carlos Quinto Huamán

Directores:

Luis Javier García Villalba

Ana Lucila Sandoval Orozco

Departamento de Ingeniería del Software e Inteligencia Artificial

Convocatoria de Septiembre

Calificación: 9 - Sobresaliente

MÁSTER EN INGENIERÍA INFORMÁTICA

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

El presente Trabajo Fin de Máster se enmarca dentro de un proyecto de investigación titulado RAMSES aprobado por la Comisión Europea dentro del Programa Marco de Investigación e Innovación Horizonte 2020 y en el que participa el Grupo GASS del Departamento de Ingeniería del Software e Inteligencia Artificial de la Facultad de Informática de la Universidad Complutense de Madrid (Grupo de Análisis, Seguridad y Sistemas, http://gass.ucm.es, grupo 910623 del catálogo de grupos de investigación reconocidos por la UCM).

Por razones de confidencialidad del proyecto se ha omitido información del trabajo desarrollado para no infringir la normativa correspondiente.

_____

Carlos Quinto Huamán

_____ _____

Luis Javier García Villalba Ana Lucila Sandoval Orozco

Abstract

In recent years, the number of mobile phones has grown significantly. As a consequence, users spend much more time using a mobile phone rather than watching TV. This change of habits is caused by enhanced the capabilities, features, characteristics and components that are present even in low-end mobile devices. A smartphone camera is commonly used to capture vast amounts of people's daily life images and videos. This could lead to exposing, unwillingly, criminal acts. On the other hand, the large number of images and videos published on the Internet are prone to editing and tampering in order to damage a person's reputation. So, digital camera generated images and videos, are being essential in forensic science, since through the use of forensic techniques, the authenticity of graphic evidence can be asserted, in order to be used in Court, empowering Defense or Prosecutor teams. Nonetheless, if criminals or attackers had greater knowledge of the weaknesses of forensic techniques, they would be able to make use of anti-forensic techniques in order to tamper images and videos without leaving any traces. That means intentionally hiding digital tampering fingerprints, in order to misguide the forensic analyst. Forensic Science should react to anti-forensic techniques, and the straightest way to do it is to research, in depth, all kind of anti-forensic methods. To do this, forensic analysts must study these techniques, so when the time comes, they are able to implement new techniques oriented towards detection of anti-forensic countermeasures. In this paper a couple of anti-forensic techniques are proposed: first is about blocking ability to identify the model of mobile device that generated a video with MP4 format and the latter is about erasing the identity of a specific video. These techniques consist of a series of algorithms based on video metadata extraction, sensor noise and wavelet transform.

Keywords

Anti-forensic techniques, Forensic Analysis, Video Metadata, Source Identification, Video Tampering, Video Anonymization, Digital Video, PRNU, Sensor noise, Wavelet Transform.

Resumen

En los últimos años el número de teléfonos móviles ha crecido notablemente. Producto de este impacto, los usuarios pasan gran parte de su tiempo sumergidos en este dispositivo. La cámara digital integrada en un teléfono inteligente es un componente usado frecuentemente para capturar gran cantidad de imágenes y vídeos de acontecimientos de la vida cotidiana de una persona. Esto permite dejar evidencia de situaciones que pueden comprometer la presunción de inocencia de las personas en actos que contravienen las leyes. Por otro lado, la gran cantidad de imágenes y vídeos que circulan en Internet están propensos a la manipulación intencionada para culpar a una persona o inculpar a otra. En este sentido, las imágenes y vídeos generados con cámaras digitales cobran gran importancia para la ciencia forense, ya que mediante la ejecución de técnicas forenses intenta verificar la autenticidad de dichas evidencias presentadas en procesos judiciales, facilitando el trabajo de los investigadores tomar las decisiones correctas. No obstante, si los delincuentes o atacantes tienen mayor conocimiento sobre las debilidades de las técnicas forenses pueden hacer uso de técnicas anti-forenses para manipular imágenes y vídeos sin dejar rastro alguno, ocultando la huella del procedimiento realizado. La ciencia forense debe hacer frente a las técnicas anti-forenses, y la principal forma de combatirlas es conocer a profundidad los métodos anti-forenses. Para ello, un analista forense debe estudiar dichas técnicas para aplicar las contra-medidas necesarias a la hora de desarrollar técnicas para detectar operaciones anti-forenses. En este trabajo se propone un par de técnicas anti-forenses: Una para eliminar la posibilidad de identificar el modelo y marca del dispositivo móvil que generó un vídeo con formato MP4 y otra para falsificar la identidad de un vídeo dado. Estas técnicas se componen de una serie de algoritmos basados en la extracción de metadatos del vídeo, el ruido del sensor y la transformada wavelet.

Palabras clave

Análisis Anti-Forense, Análisis Forense, Anonimización de Vídeos, Falsificación de vídeos, Identificación de Fuente, Metadatos de Vídeos, PRNU, Ruido del Sensor, Transformada Wavelet, Vídeos Digitales.

Agradecimientos

En primer lugar quiero expresar mi agradecimiento a mis directores, Luis Javier García Villalba y Ana Lucila Sandoval Orozco, quienes me han dado la oportunidad de realizar esta tesis. Gracias a su ayuda, dedicación, capacidad y conocimientos me han permitido cumplir los objetivos trazados. También agradezco infinitamente las facilidades como los medios y recursos que pusieron a mi entera disposición para desarrollar mis actividades sin ningún percance.

Agradezco a todos los miembros del Grupo de Análisis, Seguridad y Sistemas (GASS) por su sincera amistad, apoyo incondicional en todo momento para hacer frente a las dificultades que se han presentado durante la realización del presente trabajo y por los grandes momentos que hemos compartido. En especial a Jocelin por su constante apoyo para realizar las experimentaciones del presente trabajo.

No puedo olvidar en agradecer a mi padre, por haberme proporcionado educación, lecciones de vida y sobre todo enseñado que con esfuerzo, trabajo y constancia todo se consigue. A mi madre aunque no esté físicamente conmigo, sé que desde el cielo siempre me guía y me cuida para que todo salga bien.

De manera especial, a mi querida familia, Valeria, Amparo, Edith, Soledad, Pamela que son el principal motivo de superación, por su constante apoyo y por enseñarme que la familia no es algo importante, sino lo es todo.

Agradezco también al Ministerio de Defensa del Perú y en concreto a mi Institución el Ejército del Perú por haberme permitido cursar los estudios de Máster.

Finalmente, agradezco al Programa Nacional de Becas y Crédito Educativo del Ministerio de Educación de Perú, por haber financiado mis estudios de Máster mediante la Beca de Excelencia para Estudios de Postgrado de la Convocatoria 2013-IV.

Lista de acrónimos

AAC	Advanced Audio Coding
ADC	Analog Digital Conversion
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASO	Arbitrary Slice Ordering
AVC	Advanced Video Coding
AVI	Audio Interleave de Microsoft
BAS	Block Artifact Strength
CABAC	Context Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
CBP	Coded Block Pattern
CCD	Charge Coupled Device
CFA	Color Filter Array
CLUT	Colour Lookup Table
CMOS	Complementary Metal Oxide Semiconductor
CMY	Cyan Magenta Yellow
CYGM	Cyan-Yellow-Green-Magenta
CYYM	Cyan-Yellow-Yellow-Magenta
DBC	Decision Boundary Carving
DCT	Discrete Cosine Transform
DIP	Digital Image Processor
DSC	Digital Still Camera

DSP	Digital Signal Processor
DVD	Digital Versatile Disc
EM	Expectation-Maximization
EXIF	Exchangeable Image File Format
FAR	False Acceptance Rate
FMO	Flexible Macroblock Ordering
FPN	Fixed Pattern Noise
GOP	Group Of Pictures
GPS	Global Positioning System
GRGB	Green-Red-Green-Blue
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
IFD	Image File Directory
IID	Independent and Identically Distributed random variable
IPTC	International Press Telecommunications Council
IQM	Image Quality Metrics
ISO	International Organization for Standardization
ITU	International Telecommunication Union
JPEG	Joint Photographic Experts Group
KNN	k-Nearest Neighbors
MAP	Maximum A-Posteriori Probability
MDCT	Modified Discrete Cosine Transform
MOS	Metal Oxide Semiconductor
MOV	Metal Oxide Semiconductor

MPEG	Moving Picture Experts Group
MSE	Mean Square Error
NAL	Network Abstraction Layer
PCE	Peak to Correlation Energy
PNU	Pixel Non-Uniformity
PPS	Picture Parameter Set
PRNU	Photo Response Non Uniformity
PSD	Photoshop Data file
QMF	Separable Quadrate Mirror Filters
QTFF	QuickTime File Format
RAT	Radon Transform
RBF	Radial Basis Function
RBPS	Raw Byte Sequence Payload
RGB	Red-Green-Blue
RGBE	Red-Green-Blue-Emerland
RGBW	Red-Green-Blue-White
ROI	Region of Interest
SFFS	Sequential Forward Featured Selection
SFS	Sequential Floating Search
SPN	Sensor Pattern Noise
SPS	Sequence Parameter Set
SVM	Support Vector Machine
TIFF	Tagged Image File Format
VCL	Video Coding Layer

VQEG	Video Quality Experts Group
XMP	Extensible Metadata Platform
YUV	Intensity-Hue-value

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

La tecnología avanza de forma acelerada dando lugar a nuevos proyectos y realidades que sorprenden a sus usuarios. Este es el caso de los dispositivos móviles que se han convertido en una parte fundamental de la vida de una persona ya que se considera que resuelven necesidades de forma rápida y sencilla. El crecimiento y consecuente éxito de las empresas fabricantes de dispositivos móviles se debe a que no solo fabrican dispositivos de gama alta si no también dispositivos a con costes asequibles sin desmejorar sustancialmente sus funcionalidades y prestaciones. Actualmente muchos usuarios llevan consigo de forma constante uno o más dispositivos móviles (Smartphone, phablets, tabletas etc.). Esto se refleja en un estudio realizado en [Liu15] que indica que a finales del 2016 aproximadamente un 44.7% de usuarios de dispositivos móviles usaran un *smartphone*, alcanzando 2.060 millones de usuarios (incrementando el mercado en un 11.7%) y se alcanzara aproximadamente 1.150 millones de usuarios de tabletas.

De acuerdo al informe presentado por Cisco Systems (Cisco) febrero 2016 [CIS16], uno de los referentes y principales fabricantes de equipamiento de redes, se proyecta para el año 2020 habrá 5.500 millones de usuarios de dispositivos móviles (70% de la población mundial), sabiendo que la población estimada para el 2020 será 7.800 millones de personas según información de las Naciones Unidas. Asimismo, Cisco estima que en 2020 los teléfonos inteligentes, computadoras portátiles y tabletas generarán el 98% del tráfico mundial de datos móviles que representará 366,8 exabytes anuales frente a 89% que representó 44,2 exabytes en 2015. La generación de vídeos móviles será el elemento con mayor demanda frente a otras aplicaciones, aumentando el uso de dispositivos con conexiones 4G o 5G. Por tanto, se estima que en 2020 el 75% del tráfico de las redes móviles será por transferencia de vídeos de alta resolución. Esto significa que se generarán 81 billones de imágenes (28

imágenes diarias) y 7 billones de vídeos (más de 2,5 vídeos diarios) por habitante en el mundo.

En la Figura 1.1 se muestra el pronóstico del tráfico de dispositivos móviles mensual para el período 2015-2020.

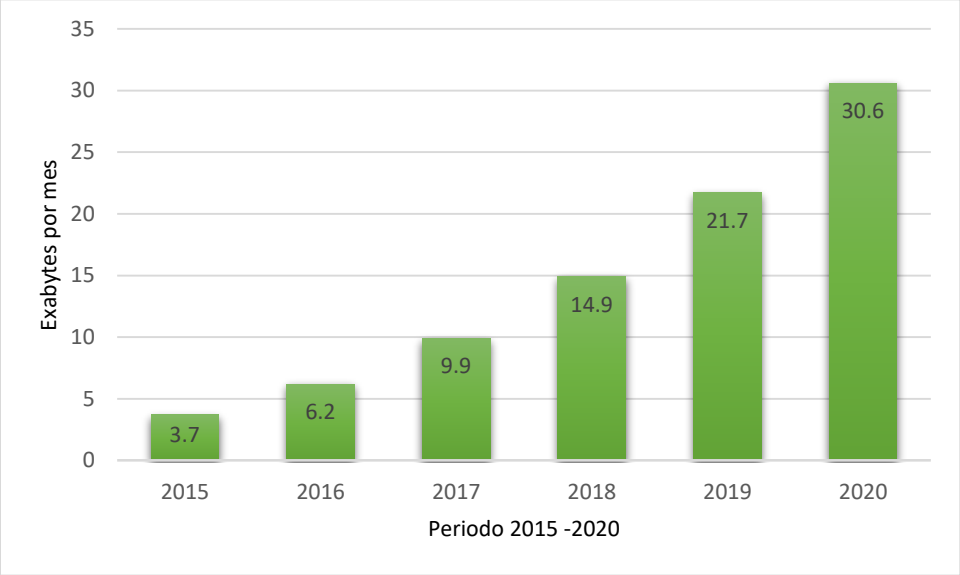


Figura. 1.1 Tráfico de red mensual en dispositivos móviles para el período 2015-2020

Las cámaras digitales de los dispositivos móviles son un elemento muy importante para el usuario, dada la capacidad que tienen de capturar escenas y grabar vídeos digitales de excelente calidad. Sin embargo, esta necesidad no se limita a dichas bondades sino además a características cada vez más exigentes. Esto obliga a los fabricantes a agregar más ventajas como: menor peso, uso más intuitivo de las funcionalidades y sobre todo la inserción de dos cámaras principales en los dispositivos móviles. Esta última característica conlleva a duplicar el sensor y las lentes para proporcionar un ángulo de visibilidad mucho más amplio, conseguir una fotografía más nítida (usando un sensor monocromático para captar el nivel de luz y el otro a color para procesar los componentes de color) y en un futuro próximo captar imágenes en tres dimensiones.

En consecuencia, los dispositivos móviles son usados para tomar fotografías y grabar vídeos, que posteriormente pueden ser almacenados en otros dispositivos, publicados en redes sociales, enviados por correo electrónico, etc. Esto hace que, desde un punto de vista forense, la gran disponibilidad de los dispositivos móviles con cámaras integradas y la cantidad de imágenes y vídeos generados hacen que los procedimientos delictivos como robo de información, pornografía infantil, secuestros, espionaje, etc., prolifere diariamente. En este sentido la detección de falsificación y anonimización de imágenes y vídeos, y la identificación de la fuente son procedimientos necesarios para determinar responsabilidades y mostrar evidencias que posteriormente serán utilizados como pruebas legales. Sin embargo, estas técnicas forenses son atacadas por otras denominadas técnicas anti-forenses responsables de alteraciones a las imágenes y vídeos, con el objeto de obstaculizar el proceso del análisis forense. Más específicamente, intentan ocultar, eliminar o falsificar alguna evidencia. Esto hace necesario desarrollar técnicas forenses capaces identificar estos métodos y hacer frente a los obstáculos y engaños digitales derivados de su uso [RCWA15].

Como se observa, el análisis forense y anti-forense para imágenes y vídeos en dispositivos móviles, son temas de gran relevancia que necesitan una investigación profunda para garantizar la creación de herramientas forenses apropiadas que permitan tomar decisiones acertadas. En el caso de imágenes y vídeos digitales, es importante estudiar las técnicas anti-forenses existentes en la literatura para anonimizar y falsificar la fuente de adquisición de imágenes y vídeos digitales.

1.1. Objeto de la investigación

En el supuesto de querer diseñar una cerradura lo suficientemente segura, que no permita su apertura sin la llave original, se debe conocer las técnicas para abrir la cerradura sin tener la llave correcta. Así, se podrá diseñar una

cerradura que evite intensiones de apertura sin tener la llave de la misma. Análogamente, el conocimiento adquirido de las técnicas anti-forenses que anonimizan o falsifican la identidad del dispositivo que generó una imagen o vídeo digital, conllevan al desarrollo de técnicas forenses de identificación de la fuente más robustas que eviten este tipo de ataques.

Las investigaciones realizadas en los últimos años han sido dirigidas en su gran mayoría a técnicas forenses y anti-forenses de imágenes digitales, dejando de lado a los vídeos digitales. Tanto las técnicas forenses de identificación de la fuente, como las anti-forenses de anonimización y falsificación de la identidad del dispositivo que la generó, se han enfocado únicamente a las cámaras tradicionales *Digital Still Camera* (DSC). Sin embargo, las cámaras de los dispositivos móviles hoy en día prácticamente han sustituido a las DSC, por lo que es necesario realizar un estudio de las técnicas anti-forenses orientado a dispositivos móviles.

Los vídeos digitales están compuestos por una secuencia de imágenes y audio. Estos vídeos puedes ser generados por diversos dispositivos diferenciándose en particularidades de cada fabricante, pero todos los vídeos se asemejan al contener intrínsecamente un patrón del ruido del sensor creado durante el proceso de generación del mismo, y utilizarse como medio de identificación de la fuente [LFG06]. Específicamente, las cámaras digitales de dispositivos móviles cuentan, en su mayoría, con un tipo de sensor que deja rasgos característicos en cada fotograma de un vídeo.

La presente investigación se centra en el estudio de las técnicas anti-forenses orientadas a la anonimización y falsificación de la fuente de un vídeo. Específicamente, se centra en el análisis exhaustivo de los átomos contenidos en un vídeo con formato MP4, esto permitirá extraer del vídeo tanto los fotogramas y el audio para posteriormente realizar la anonimización y falsificación de identidad del dispositivo móvil que generó el vídeo en cuestión.

En el trabajo se propone, en primer lugar, desarrollar una técnica anti-forense para realizar la anonimización de un vídeo con formato MP4; en este proceso se implementan 2 algoritmos de descomposición de vídeo: uno para la extracción del flujo elemental *Advanced Audio Coding* (AAC), que contiene el audio, y otro para la extracción de fotogramas desde los átomos del vídeo con formato MP4. Asimismo un algoritmo para eliminar la huella del sensor basado en el ruido *Photo Response Non Uniformity* (PRNU) y la recomposición del vídeo con formato MP4 anonimizado. En segundo lugar, se presenta una técnica anti-forense para realizar la falsificación de la fuente de un vídeo con formato MP4. Para tal efecto se presenta un algoritmo principal para realizar la falsificación de la fuente de un vídeo con formato MP4, que reutiliza algoritmos planteados para anonimizar un vídeo y un algoritmo para falsificar la identidad de los fotogramas de un vídeo con formato MP4.

1.2. Trabajos Relacionados

Actualmente existe un número reducido de investigaciones sobre técnicas anti-forenses enfocadas a vídeos de dispositivos móviles, y en concreto técnicas para anonimizar y falsificar la identidad del dispositivo móvil que lo generó. La mayor parte de las investigaciones realizadas en el campo de la identificación de la fuente se han realizado para imágenes fotográficas. Sin embargo, se requiere herramientas forenses específicas para las señales de vídeo debido a sus peculiaridades y la amplia gama de posibles alteraciones que se pueden aplicar en ellos.

La mayoría de las técnicas que se pueden aplicar a una imagen, se pueden utilizar con los fotogramas de un vídeo. En [SOAGRC+13] se realiza una comparación detallada de los principales grupos de técnicas de identificación de fuente de adquisición. Estas se dividen en cinco grupos y están basadas en: metadatos, características de la imagen, defectos de la matriz PRNU e interpolación cromática, imperfecciones del sensor y la transformada wavelet.

En [M.11] se propone una técnica para realizar la modificación del número fotogramas y orden del grupo de imágenes o *Group Of Pictures* (GOP) de un archivo de vídeo con formato MP4 de forma oculta, sin asumir la decodificación del flujo de vídeo. Este método se origina porque la norma *Moving Picture Experts Group* (MPEG) no especifica el número ni el orden de los fotogramas P y B en la estructura (GOP), así como tampoco especifica el número de fotogramas en los trozos individuales del vídeo y audio. Pero si el codificador del vídeo procesa una estructura fija (GOP) esta técnica no puede utilizarse.

En [YHW12] se propone un método de identificación de la fuente utilizando los fotogramas extraídos de los vídeos. Las características de probabilidad condicional se extraen directamente de los fotogramas del vídeo. En las pruebas realizadas se utilizaron 4 modelos diferentes de cámaras y un clasificador *Support Vector Machine* (SVM). En un primer experimento aplicado en el dominio del espacio con los valores de luminancia, se obtuvo un 82.6% de precisión. En un segundo experimento, usando el mismo conjunto de vídeos y tomando el valor de luminancia el porcentaje de acierto en la clasificación fue del 100%. En un tercer experimento en donde se utilizaron un conjunto de vídeos con mayores cambios en las escenas se obtuvo un 97.2% de acierto.

En [PPML15] se estudian técnicas forenses y anti-forenses aplicables a los procesos de edición de vídeos con codificación H.264/AVC. El trabajo se basa en el hecho de que la excesiva predicción residual en la codificación puede aparecer en los fotogramas que se han codificado con predicción intra e inter. En el primer método implementado los autores evalúan el resultado residual después del filtrado de desbloqueo para revelar las operaciones de edición realizadas. En el segundo, se utiliza un mecanismo de control de frecuencia para comprobar los parámetros de cuantificación.

Contexto de la Investigación

El presente Trabajo Fin de Máster se enmarca dentro de un proyecto de investigación titulado RAMSES aprobado por la Comisión Europea dentro del Programa Marco de Investigación e Innovación Horizonte 2020 y en el que participa el Grupo GASS del Departamento de Ingeniería del Software e Inteligencia Artificial de la Facultad de Informática de la Universidad Complutense de Madrid (Grupo de Análisis, Seguridad y Sistemas, http://gass.ucm.es, grupo 910623 del catálogo de grupos de investigación reconocidos por la UCM). Seguidamente se detallan algunos datos:

Convocatoria: H2020-FCT-2015

Tipo de Propuesta: Acción de Innovación

Número de Propuesta: 700326

Acrónimo de la Propuesta: RAMSES

Entidad financiadora: Comisión Europea, Horizonte 2020 – Programa Marco de Investigación e Innovación

Entidades participantes: Policia Judiciária (Portugal), Belgian Federal Police (Bélgica), Research Centre on Security and Crime (Italia), Politecnico di Milano (Italia), College of the Bavarian Police (Alemania), Saarland University (Alemania), Trilateral Research and Consulting (Reino Unido), University of Kent (Reino Unido), Dirección General de la Policía (España), Treelogic Telemática y Lógica Racional para la Empresa Europea S.L. (España), Universidad Complutense de Madrid (España).

Duración, desde: 01-09-2016 hasta: 31-08-2019

Investigador responsable (UCM): Luis Javier García Villalba

1.3. Estructura del trabajo

El presente trabajo está dividido en 7 capítulos. El Capítulo 2 introduce algunas definiciones fundamentales para comprender el análisis forense y las técnicas anti-forenses en vídeos de dispositivos móviles. Se presenta el concepto de un vídeo digital, se explica el proceso de generación de un vídeo digital y los elementos de la cámara que sirven de base para las técnicas forenses en vídeos. También se realiza un estudio de la compresión y los tipos de codificación de un vídeo. Por último se realiza un análisis sobre la compresión de vídeo H264/MPEG AVC y la compresión de audio AAC.

El Capítulo 3 realiza un estudio de los diferentes contenedores multimedia que almacenan la información de un vídeo, los metadatos almacenados en imágenes y vídeos, la estructura de los átomos que componen un vídeo y por último un análisis exhaustivo sobre la especificación del contenedor multimedia MP4.

El Capítulo 4 presenta un estado del arte sobre técnicas forenses y anti-forenses relacionadas con la identificación de la fuente de un vídeo. Primero, se presentan las principales técnicas de análisis forense en vídeos e imágenes agrupadas según su objetivo, se detallan principalmente las de técnicas de identificación de la fuente de adquisición y las de compresión y manipulación de vídeos digitales de dispositivos móviles. Seguidamente, se estudian las técnicas anti-forense de imágenes y vídeos.

El Capítulo 5 presenta las contribuciones de este trabajo. En primer lugar se presentan las técnicas anti-forenses para realizar la anonimización de un vídeo con formato MP4, detallando los algoritmos implementados durante el proceso: dos algoritmos de descomposición del vídeo (uno de extracción del flujo elemental AAC (audio), y el otro extracción de fotogramas desde los átomos del vídeo con formato MP4), un algoritmo para realizar la eliminación de la huella del sensor basado a PRNU de los fotogramas extraídos y, por último, la

reconstrucción del vídeo con formato MP4 anonimizado. En segundo lugar se presenta otra técnica anti-forense para realizar la falsificación de identidad de un vídeo con formato MP4.

El Capítulo 6 describe los experimentos realizados para evaluar la efectividad de los algoritmos implementados de anonimización y falsificación de un vídeo con formato MP4 propuestos en el capítulo 5 y presenta los resultados obtenidos.

El Capítulo 7 muestra las principales conclusiones de este trabajo, las líneas futuras de investigación y las publicaciones derivadas del presente trabajo.

En el Anexo A se realiza un análisis de los átomos contenidos en el átomo *moov* de la especificación.

2. VÍDEOS DIGITALES

El objetivo de este capítulo es presentar información detallada sobre los vídeos digitales. Primero se presenta el proceso de generación de un vídeo digital de diferentes tipos de dispositivos mostrando los componentes que intervienen en este proceso como son tipos de codificación y compresión del vídeo, resaltando el estándar de compresión de vídeo MPEG y compresión de audio AAC.

Estas definiciones son de vital importancia en las técnicas de análisis forense descritas en los siguientes capítulos.

2.1. Componentes

Un vídeo está compuesto, principalmente, por una secuencia de imágenes (fotogramas) capturadas junto a una secuencia de audio, que a una velocidad determinada genera una escena en movimiento [MT15]. Técnicamente, un vídeo no es más que un contenedor que se encarga de encapsular todos los componentes (códecs de vídeo, de audio y datos adicionales como los subtítulos). En la Figura 2.1 se ilustra un esquema de componentes básicos de un vídeo estándar.

La calidad de un vídeo digital es independiente del medio y depende solo de la calidad de los procesos de conversión. La mayor ventaja de producir un vídeo digital es el bajo coste añadido por la facilidad de realizar copias las cuales no ocasionan pérdida de calidad.

Figura 2.1 Esquema de componentes básicos de un vídeo estándar

Existen diferentes métodos de procesamiento de imágenes y vídeo agrupados en las siguientes categorías [Moe12]:

- **Compresión**: Es probablemente el método más conocido y usado porque posee procedimientos que permiten la compresión de fichero multimedia (imagen o vídeo). Su principal objetivo es reducir el número de datos usados para representar el fichero. Por ejemplo, la compresión de vídeo combinando la compresión espacial de imágenes y la compensación de movimiento temporal.

- **Manipulación**: Abarcan los métodos que permiten editar una imagen o vídeo. Por ejemplo, girar o escalar una imagen y mejorar la calidad de la imagen cambiando el contraste.

- **Post-Procesamiento**: Aplica técnicas para mejorar la calidad de la imagen y hacer más evidente en ellas ciertos detalles que se desean hacer notar. Está más enfocado a imágenes, pero incluye técnicas para el tratamiento de datos de vídeo, como compresión y encapsulamiento.

- **Análisis:** Analiza una imagen o vídeo con el propósito de encontrar los objetos de interés y extraer algunos parámetros de este objeto. Por ejemplo la búsqueda de la posición y tamaño del objeto.

- **Visión artificial**: Son procesos utilizados por industrias productoras para aumentar la fiabilidad del procesamiento de vídeo, procesamiento de imagen y análisis de imagen.

- **Visión por ordenador**: Se realiza a través de algoritmos avanzados, como aquel que realiza reconocimiento de caras y como procesamientos en que se aplican más de una cámara.

2.2. Proceso de Generación de un Vídeo

En el proceso de generación de vídeos digitales, también conocido como *pipeline* [THN+] [Inc12] se combinan imagen, audio y datos. La estructura del *pipeline* es similar entre dispositivos del mismo tipo, diferenciándose sólo en la calidad de la cámara o las prestaciones adicionales que ofrece.

Muchos de los detalles del proceso de generación de un vídeo en una cámara digital son propios de cada fabricante y tipo de dispositivo, almacenando información confidencial. Sin embargo, dado que un vídeo se crea a partir de un conjunto de imágenes, hay una estructura general aplicable a todas las cámaras. Una cámara digital se compone de un sistema de lentes, un grupo de filtros, una matriz de filtro de colores o *Color Filter Array* (CFA), un sensor de imagen y un procesador de imagen o *Digital Image Processor* (DIP) [BSM08a].

La generación del vídeo sigue un proceso similar al de generación de una imagen hasta el proceso realizado por el (DIP) que forma parte del procesador digital de señales o *Digital Signal Processor* (DSP), en el que se adiciona un micrófono y un conversor de señal analógica a digital. La Figura 2.2 muestra esquemáticamente este proceso.

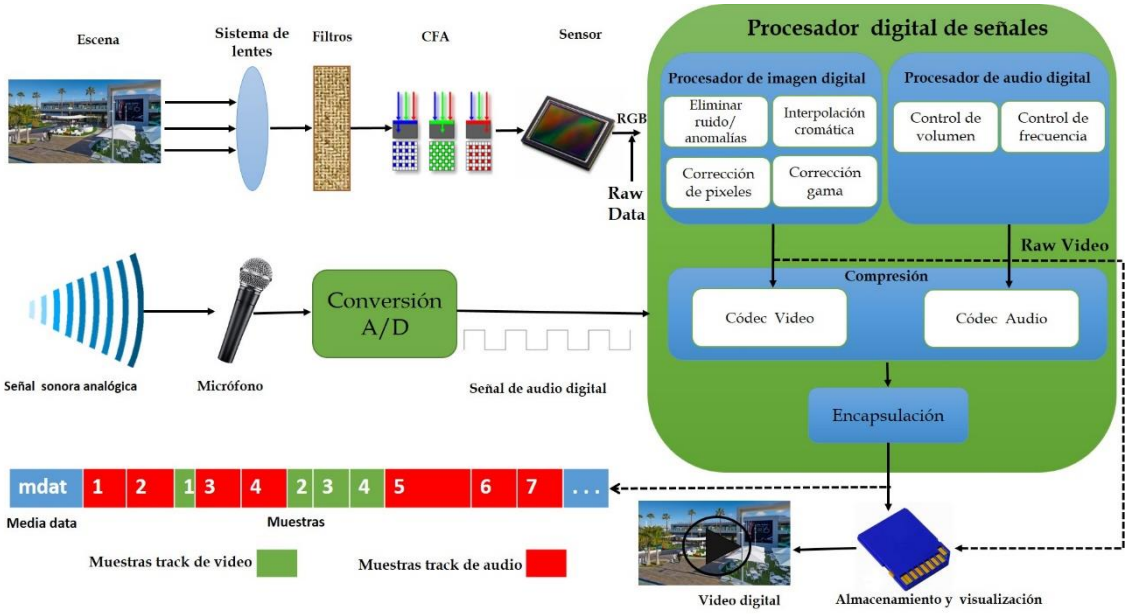


Figura 2.2: Proceso de generación de un vídeo digital

Para generar un vídeo estándar normalmente se ejecutan dos acciones en paralelo: El procesamiento de secuencia de imágenes o vídeo y el procesamiento de audio. El procesamiento de secuencia de imágenes o vídeo se inicia cuando el sistema de lentes captura la luz de la escena controlando la exposición, el foco y la estabilización de la imagen. Luego, la luz que entra en la cámara a través del sistema de lentes pasa varios filtros que mejora la calidad visual de la imagen. Por lo general se incluyen dos filtros: un filtro infrarrojo que absorbe o refleja la luz permitiendo que únicamente pase a la siguiente fase la parte visible del espectro, evitando que la radiación infrarroja cause pérdida de nitidez en la imagen y un filtro *anti-aliasing* que se encarga de limpiar la señal generando imágenes con contornos más suaves.

A continuación, la luz pasa al sensor de la imagen a través de la matriz de filtro de color, que son elementos sensibles a la luz llamados píxeles. Para reducir el coste de los dispositivos móviles generalmente se usa un solo sensor de imagen monocromático y se le antepone una matriz de filtros de color para generar una imagen a color. Debido a la demanda de calidad de imágenes y vídeo, actualmente los fabricantes de dispositivos móviles están optando por

insertar dos sensores de imagen que cubran esta necesidad: un primer sensor 100% monocromático encargado de mejorar la captura de luz en las tomas con luminosidad más escasa y un sensor *Red-Green-Blue (*RGB) que en realidad es monocromático y trabaja con una matriz de filtros de color en concreto.

Esta señal se convierte en una señal digital y se transmite al procesador de imagen que forma parte del DSP. La señal digital generada por uno o más sensores es captada por el procesador de imagen que la somete a diferentes procesos (procesos de cámaras) con el fin de estabilizar la señal y corregir alteraciones *artifacts* como eliminar el ruido y otras anomalías introducidas [Cor12] [Nak05].

A continuación, la señal estabilizada pasa al proceso de compresión mediante un códec en concreto que luego se encapsula en un contenedor que soporte estos datos. En las cámaras de dispositivos móviles normalmente se utiliza H.264 o MPEG-4 parte 10 que es la norma que define el códec de vídeo de alta compresión, con un contenedor multimedia MP4 incluido en la norma MPEG-4 Parte 14. Este contenedor tiene la capacidad de almacenar o encapsular vídeo y audio. Finalmente el fichero con extensión MP4 será guardado en un dispositivo de almacenamiento.

El procesamiento del audio se inicia cuando la señal sonora, transmitida por el aire, es capturada a través del micrófono que funciona como un sensor electro acústico. El micrófono transforma las ondas sonoras en una señal eléctrica para aumentar su intensidad y transmitirla a un conversor analógico digital o *Analog Digital Conversion* (ADC), que a su vez, la convierte a una señal digital. La señal de audio digital es capturada por el procesador de audio digital, que la procesa para mejorar la calidad del audio antes de la compresión. Entre los procesos que realiza están el control de volumen ajustándolo al nivel más óptimo posible y el control de frecuencia.

Después la señal es comprimida con algún algoritmo de codificación para

posteriormente ser encapsulado en un contenedor y almacenada en un dispositivo. En su gran mayoría las cámaras de los dispositivos móviles utiliza el algoritmo de compresión con perdida AAC que es un formato de señal digital establecido en MPEG-4 parte 3, norma que define la codificación de audio.

Otros procesos de cámaras comunes en el DSP son los siguientes:

- **Interpolación cromática o *demosaicing***: Es el proceso más complejo desde el punto de vista computacional porque el algoritmo utiliza los valores de los pixeles vecinos para obtener todos los canales que faltan que no han sido medidos, debido a que el sensor únicamente proporciona los colores filtrados por el CFA.

- **Corrección de píxeles**: Corrige píxeles defectuosos originados por imperfecciones en el sensor. Esta corrección de píxeles se realiza mediante la interpolación.

- **Balanceo de negros y blancos**: Asegura que no hayan dominantes indeseables en grises, blancos y negros. Para ello compara el canal verde (G) con los canales (R y B), igualando éstos hasta que los tres tengan la misma amplitud.

- **Proceso de corrección *gamma***: Ajusta los valores de intensidad de la imagen. Los algoritmos para ejecutar estos procesos varían de acuerdo a cada fabricante y modelo.

- **Corrección de sombras**: Se corrigen errores típicos de sombreados introducidos por la lente o por falta de uniformidad en el sensor.

- **Ajuste de nivel de negros**: Establece el punto donde comienza a construirse la señal de vídeo. Crea un negro perfecto de referencia.

Un proceso muy importante para la generación del vídeo es la sincronización de las muestras de audio y vídeo realizado por el DSP a través del software que contiene. Este proceso se realiza para obtener un vídeo sin desfase y coherente a la realidad.

2.3. Compresión del Vídeo

Para reducir la cantidad de datos de un vídeo existen tres estrategias: reducir las dimensiones de los fotogramas, reducir la velocidad de los fotogramas capturados y la compresión de datos, siendo esta última la más usada. Los procesos de compresión o codificación convierten el vídeo digital en un formato adecuado para su transmisión o almacenamiento, mientras que reduce típicamente el número de bits. En la Tabla 2.1 se muestra una cronología de los formatos y estándares de codificación de vídeos.

En bruto o sin comprimir, un vídeo digital requiere una gran velocidad de bits; aproximadamente 216 Mbits por segundo en vídeos de definición estándar no comprimido [Iai10]. La compresión es necesaria para un almacenamiento y transmisión óptima del vídeo. La compresión se compone de un sistema de codificación que convierte los datos recibidos en un formato comprimido usando un número reducido de bits antes de la transmisión o almacenamiento, y un sistema de decodificación que convierte los datos comprimidos a una representación de los datos de vídeo original. La compresión de datos se consigue mediante la eliminación de la redundancia, es decir, los componentes que no son necesarios para una reproducción fiel de los datos.

Los tipos de algoritmos que se utiliza para realizar la compresión de un vídeo son dos: (1) algoritmo de compresión con pérdida y (2) algoritmo de compresión sin pérdida. Para la compresión de la imagen e información del vídeo generalmente se utiliza el algoritmo de compresión con pérdida. Según el tipo de algoritmo de compresión utilizado, la calidad de la imagen tendrá

perdidas en mayor o menor proporción. Los sistemas de compresión de vídeo con pérdida se basan en el principio de la eliminación redundancia subjetiva, es decir, los elementos de la secuencia de fotogramas que se pueden quitar sin que afecte de manera significativa la percepción del espectador de la calidad visual.

La compresión puede estar implementada a nivel de *software* o de *hardware*, siendo esta última más rápida y eficaz. Asimismo, existen códecs optimizados orientados a la calidad de los fotogramas y otros a la velocidad de codificación.

Para realizar la compresión del vídeo como se ha detallado anteriormente, se obtiene la señal digital estabilizada, luego se codifica la imagen a sus componentes originales RGB, *Intensity-Hue-value* (YUV) o cualquier método de almacenamiento de vídeo digital, y por último se aplican los algoritmos que realizan la compresión. Algunas de las técnicas de compresión existentes son las siguientes:

- **Tabla de consulta de color**: También denominada *Colour Lookup Table* (CLUT), es una tabla que almacena la información de color de los píxeles o regiones.

- **Truncamiento**: Reduce el número de bits por cada componente, píxel o resolución, siendo esta técnica la más simple de todas por su mínima complejidad al momento de ser procesado.

- **Interpolación de regiones**: Permite que las regiones tengan cambios logrando almacenar la región por primera vez y, gracias a las técnicas de interpolación, reconstruir la siguiente región.

- **Transformación de regiones**: Se cambia la información de una región por otra, devolviendo un resultado visual similar. Uno de los algoritmos usados para la transformación es la Transformada Coseno Discreta o *Discrete Cosine Transform* (DCT).

- **Compensación de movimiento**: Usa varias de las técnicas descritas anteriormente, centrándose en encontrar las partes que sufren cambios menores, dividir la imagen en bloques y realizar los cambios necesarios. Cuando los cambios de las regiones son mínimos, procura no hacer cambios o predecir valores. El estándar MPEG aplica esta técnica.

Tabla 2.1: Cronología de los formatos y estándares de codificación de vídeo

Año	Formato/Estándar	Desarrollado/Estandarizado	Rango de aplicación
1984	H.120	CCITT (ITU-T now)	Video transmission (NTSC or PAL) over P2P data communication lines
1988	H.261	CCITT (ITU-T now)	Video transmission over ISDN lines
1993	MPEG-1	ISO/IEC	ISO/IEC&Digital satellite/cable TV;Vídeo CD; ...
1994	H.262/MPEG-2	(joint)ITU-T & ISO/IEC	DVD players;camcorders; video recorders;distribution networks
1995	H.263	ITU-T	Video conferencing; cellphone codec;World Wide Web-distributed video
1998	MPEG-4	Visual ISO/IEC	Visual ISO/IEC&DVD; mobile multimedia;internet multimedia; broadcast TV;videophone
2000	VP3	On2 Tech.	Released into the open source community in 2001 by On2 Tech.
2001	VP4	On2 Tech.	
2002	VP5	On2 Tech.	
2003	H.264/MPEG-4 AVC	(joint)ITU-T & ISO/IEC	HD video like Blu-ray Disc;HDTV broadcast over terrestrial, cable and satellite; video streaming (Vimeo, YouTube, iTunes Store, ...);
2003	VP6	On2 Tech.	Adobes Flash video, YouTube video,video conference over IP,satellite broadcasts
2005	VP7	On2 Tech.	Adobes Flash video, video streaming,...
2005	AVS	The government of the People's Republic of China	within mainland China
2006	SMPTE 421M/VC-1	Microsoft -SMPTE	Blu-ray Disc,Window media,HD DVD; ...
2008	VP8	On2 Tech. Google	Adobes Flash video, YouTube video,... (open source by Google)
2010	Dirac/VC-2	BBC Research -SMPTE	Ultra HDTV,used by BBC to transmit HDTV
2013	VP9	Google	HTML5 video, some smart TVs, You-Tube for 4K resolution, ...
2013	H.265/HEVC	(joint) ITU-T & ISO/IEC	Ultra HDTV, used by ATSC, DVB, Blu-ray disc

2.3.1. Tipos de Codificación

Existen los siguientes tipos de codificación:

- **Codificación Intra o Espacial**: Se encarga de comprimir cada imagen de forma independiente dejando de lado los datos de tiempo en el proceso de compresión. Se denomina codificación intra, interna, o espacial. Dentro de esta categoría se encuentra el estándar de compresión *Joint Photographic Experts Group* (JPEG). El vídeo se puede codificar mediante una sucesión de fotogramas codificados inicialmente con JPEG. Esta codificación recibe el nombre de JPEG en movimiento. Para comprimir un vídeo se utilizan las similitudes entre píxeles adyacentes en zonas de imágenes lisas, para zonas de color variado se utilizan frecuencias espaciales dominantes. Este tipo de codificación se divide en codificación por predicción y codificación de la transformada del coseno.

- **Codificación Inter o Temporal**: Este tipo de codificación aprovecha la similitud de imágenes sucesivas. El codificador inter envía la diferencia entre la imagen previa y la actual en forma de codificación diferencial, para eliminar la redundancia temporal teniendo en cuenta la información de las imágenes previamente enviadas. Luego, envía únicamente las zonas de la imagen que han sufrido cambios de un fotograma a otro. La primera imagen almacenada con anterioridad es denominada fotograma de referencia o *key frame*, y es un elemento necesario para el codificador. Esto se debe a que el fotograma será comparado con los fotogramas siguientes, también es necesario una imagen previamente almacenada para que el codificador genere las imágenes siguientes. La información sobre un grupo de fotogramas sirve de referencia para identificar y eliminar información redundante en secuencias posteriores hasta identificar un nuevo fotograma de referencia. Las imágenes obtenidas por la diferencia entre un fotograma de referencia y una imagen no comprimida se le

conocen como *delta frames*. Para garantizar una mejor transmisión de datos el compresor utiliza un número limitado de fotogramas de referencia, para evitar transmitir errores por el exceso de secuencias de fotogramas de referencia. MPEG es un estándar que realiza este tipo de compresión para mantener una alta calidad de la imagen a pesar de usar tasas de compresión superiores a la codificación intra o espacial.

2.3.2. Estándar de Compresión MPEG

Los dispositivos móviles en su mayoría utilizan el estándar de compresión MPEG. MPEG contiene varios estándares y éstos a su vez se clasifican en partes, que se actualizan o amplían periódicamente para dar soporte a nuevos requerimientos. Este algoritmo de compresión de vídeo utiliza dos técnicas fundamentales: Compensación del movimiento basada en bloques para la reducción de la redundancia temporal, y Codificación DCT para la reducción de la redundancia espacial.

La primera, se aplica en ambas direcciones: hacia adelante o causal y hacia atrás o no causal. La señal restante es codificada utilizando las técnicas basadas en transformaciones. Los predictores de movimiento, denominados vectores de movimiento, son transmitidos junto con la información espacial. Una secuencia de vídeo MPEG es básicamente la salida del material en bruto (flujo de bits) de un codificador y solo contiene lo necesario para que un decodificador restablezca la imagen original. Según [Woo05] la secuencia de vídeo MPEG tiene una estructura en capas bien definidas como se muestra en la Tabla 2.2. Cada capa, contiene la muestra individual, un encabezado que posee fragmentos de metadatos y una alineación del patrón de bits para que se diferencien dentro del flujo de bits.

Tabla 2.2: Estructura de una secuencia de vídeo MPEG

Capas	Descripción
1	Secuencia de Vídeo
2	Grupo de Imágenes o GOP
3	Imágenes
4	Rebanada o *Slice*
5	Macrobloques
6	Bloques
7	Muestras o *Samples*

Un GOP es la unidad fundamental de codificación temporal y una de sus características es especificar el orden de las imágenes. Está representado por secuencias de 10 a 30 fotogramas. Puede contener distintos tipos de imágenes [Bov05]:

- **Imágenes de codificación intra (*I-Frames*)**: Son imágenes de referencia que representan una imagen fija independientes de los otros tipos de imágenes.

- **Imágenes de codificación mediante predicción (*P-Frames*)**: Contienen información de la compensación de movimiento de la imagen precedente, ya sea del tipo *P-Frame* o *I-Frame*.

- **Imágenes de codificación mediante predicción bidireccional (*B-Frames*)**: Contienen información diferente de las imágenes precedente y siguiente.

Un GOP siempre empieza con una imagen tipo *I-Frame*, seguida de cualquier número de *I-Frames* y *P-Frames*, considerados como marcos de anclaje. Entre cada par de fotogramas consecutivos de anclaje pueden aparecer varios *B-Frames* como se observa en la Figura 2.3.

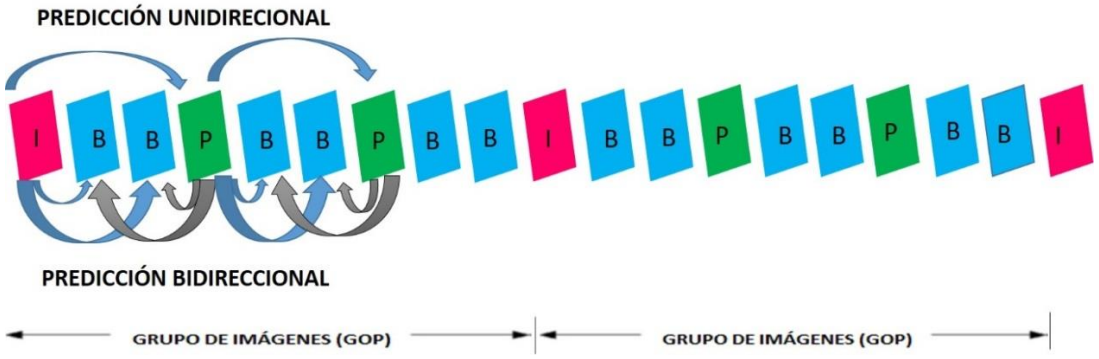


Figura 2.3: Estructura de un grupo de imágenes

Un *I-Frame* no se refiere a ningún otro fotograma de vídeo y, por lo tanto, puede ser decodificado de forma independiente, proporcionando un punto de entrada para un rápido acceso aleatorio al vídeo comprimido. Por otro lado, la codificación de un fotograma *P-Frame* se basa en un marco de anclaje anterior, mientras que la codificación de un fotograma *B-Frame* se puede basar en dos marcos de anclaje: uno anterior y uno posterior. Si la información de la imagen I-*frame* se encuentra en RGB debe convertirse a YUV, antes de comenzar el proceso de compresión. Esto es heredado de JPEG y es vital para la compresión ya que el ojo humano es más sensible al componente Y que a las componentes CbCr. Esto permite al algoritmo disminuir la cantidad de datos para la representación del color.

Estas imágenes se dividen en una secuencia de macrobloques denominados *slice* o rebanada agrupados según el color promedio para ayudar al decodificador a recuperarse en caso de perder la sincronización. Así si se pierde sincronización de un macrobloque, el decodificador recurre a la rebanada correspondiente del *I-frame* o *P-frame* que proporciona la rebanada asociado a dicho macrobloque. Los macrobloques se usan para reducir la complejidad en la generación de los *B-frames* y encontrar fácilmente las áreas de colores idénticos. Un macrobloque se define como un arreglo cuadrado de 16x16 píxeles operados individualmente. Con los macrobloques se puede identificar fácilmente cuáles son las redundancias con respecto a un *I-frame* y un *B-frame*. La generación de

macrobloques, en consecuencia, depende de qué tipo de fotograma que se quiere componer. La codificación de un macrobloque se realiza mediante la DCT. Estos macrobloques a su vez se dividen en bloques de coeficientes DCT que tienen un tamaño de hasta de 4x4 pixeles, esto varía de acuerdo al estándar MPEG que se utilice. Posteriormente, se cuantifican los coeficientes del DCT con pérdidas y la entropía, y se codifica la tabla de Huffman sin pérdida para lograr la compresión.

2.3.3. Compresión H.264/MPEG-4 AVC

El estándar H.264/MPEG-4 *Advanced Video Coding* (AVC) es una evolución de la codificación establecida en los estándares MPEG e *International Telecommunication Union* (ITU). Algunas mejoras con respecto a los estándares anteriores son: (1) el uso de estimación de movimiento o *Motion Estimation* para mejorar la predicción entre fotogramas *(inter-picture prediction)* y eliminar redundancias temporales, (2) el uso de la correlación espacial de los datos para realizar la predicción intra fotogramas *(intra-picture prediction)* y (3) a construcción de residuos como la diferencia entre las imágenes predecidas y las imágenes originales. El uso de una transformada espacial discreta y la aplicación de un filtro para eliminar redundancias espaciales y por último la codificación de la entropía de los coeficientes residuales de la transformada y de los datos de apoyo, como los vectores de movimiento que se calculan al usar la estimación de movimiento.

El estándar H.264/AVC se divide en dos partes: Una capa de codificación de vídeo o *Video Coding Layer* (VCL), que representa la codificación y el contenido de vídeo, y la capa de abstracción de red o *Network Abstraction Layer* (NAL), que adapta la información que proviene de VCL, que posteriormente le proporciona un formato. En este trabajo se hace énfasis en la capa de abstracción de red que es necesaria para realizar la extracción de fotogramas presentado en capítulos posteriores.

2.3.3.1. Formato de la Trama H.264

Una trama H.264 está constituida por unidades NAL, que a su vez están compuestos por una cabecera y una zona de datos *Raw Byte Sequence Payload* (RBPS) como se ilustra en la Figura 2.4. Esta zona de datos puede contener dos tipos de información: unidades VCL que transportan la secuencia de vídeo codificada o unidades no VCL que transportan información adicional necesaria para la decodificación [PO11].

Figura 2.4: Trama H.264 con tres unidades NAL.

Por consiguiente, una secuencia de vídeo codificada está formada por una colección de unidades NAL que pueden ser transferidas a través de una red orientada a la transmisión de paquetes o almacenadas en un fichero. La separación que existe entre el VCL y NAL permite diferenciar la información codificada del vídeo, de la información de transporte de la trama. Las unidades no VCL más importantes que se usan en el proceso de decodificación son:

- **Conjunto de parámetros de secuencia o *Sequence Parameter Set* (SPS):** Transporta información válida para toda la secuencia de imágenes como el perfil, el nivel, las dimensiones y el formato del muestreo.

- **Conjunto de parámetros de imágenes o *Picture Parameter Set* (PPS)**: Incluye datos relativos a una o más imágenes de forma individual como la codificación entrópica o el uso de la forma de transformación 8x4.

Las unidades VCL transportan la secuencia de vídeo codificada y organizada en rebanadas o *slice*. En la zona útil de datos de cada unidad VCL se encuentra la información de los macrobloques como tipo de macrobloque, el parámetro de cuantificación (Q), el modo de predicción *Coded Block Pattern* (CBP), vectores de movimiento asociados (MV) y la información residual del resultado de su

codificación. En la Figura 2.5 se ilustra la estructura jerárquica de una NAL que contiene una VCL.

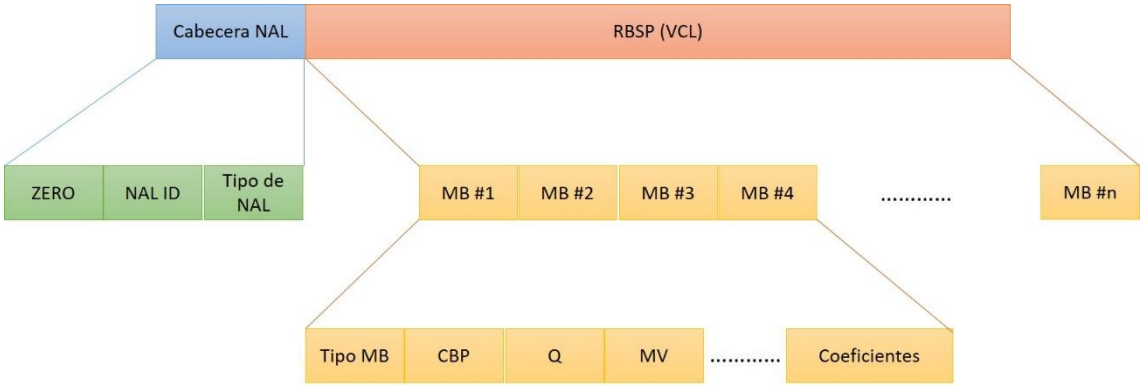


Figura 2.5: Estructura jerárquica de una trama H264

2.3.4. Formato de Tren de Bytes

El tren de bytes se refiere al encapsulado de un tren de unidades NAL que contiene prefijos de código de inicio y unidades NAL. En el anexo b de la recomendación ITU H264 (02/2016) se especifica la sintaxis y la semántica de un formato de tren de bytes. Se puede utilizar en aplicaciones que distribuyen algunos o todos los trenes de unidades NAL en la forma de un tren ordenado de bytes o de bits en el que es necesario identificar los límites de las unidades NAL con respecto a patrones de los datos.

El formato del tren de bytes consiste en una secuencia de estructuras sintácticas unidad NAL de tren de bytes. Cada estructura de este tipo puede contener: un prefijo o código de inicio, una estructura sintáctica *nal_unit* (*NumBytesInNALunit*), un elemento sintáctico *zero_byte* adicional, uno o más elementos sintácticos *trailing_zero_8bits* adicionales. Si se trata de la primera unidad NAL del tren de bytes en el tren de bits, puede también incluir uno o más elementos sintácticos adicionales *leading_zero_8bits*. En la tabla 2.3 se presenta el diagrama sintáctico de unidades NAL.

Tabla 2.3 Diagrama sintáctico de unidades NAL.

byte stream nal unit(NumBytesInNALunit) {	Descriptor
while(next bits(24) != 0x000001 && next bits(32) != 0x00000001)	
leading zero 8bits /* igual a 0x00 */	f(8)
if(next bits(24) != 0x000001)	
zero byte /* igual a 0x00 */	f(8)
if(more data in byte stream( )) {	
start code pre_x one 3bytes /* igual a 0x000001 */	f(24)
nal unit(NumBytesInNALunit)	
}	
while(more data in byte stream( ) && next bits(24) != 0x000001 && next bits(32) != 0x00000001)	
trailing zero 8bits /* igual a 0x00 */	f(8)

El orden de las unidades NAL del tren de bytes tiene que ser idéntico al orden de decodificación de las unidades NAL incluidas en las unidades NAL de tren de bytes. El contenido de cada unidad NAL de tren de bytes se asocia con la misma unidad de acceso de la unidad NAL incluida en la unidad NAL de tren de bytes.

2.3.5. Codificación de Audio Avanzado

La codificación de audio avanzado o AAC es un estándar de codificación de audio para la compresión de audio digital con pérdida establecido en el estándar MPEG-4 parte 3 y la norma *International Organization for Standardization* (ISO)/*International Electrotechnical Commission* (IEC):2009 [fS03b]. Está diseñado para ser el sucesor del formato MP3, ya que consigue una calidad de sonido mejor que MP3 a velocidades de bits similares [Bra99].

AAC es un algoritmo de codificación de audio de banda ancha que utiliza dos estrategias de codificación primaria para reducir drásticamente la cantidad de datos necesarios para representar un audio digital en alta calidad:

- Los componentes irrelevantes de la señal se descartan.
- Las redundancias en la señal de audio codificado son eliminados.

El proceso de codificación real consta de los siguientes pasos:

- La señal se pasa del dominio del tiempo al dominio de la frecuencia usando *Modified Discrete Cosine Transform* (MDCT). Esto se hace mediante el uso de bancos de filtros que tienen un número apropiado de muestras de tiempo.

- La señal de dominio de la frecuencia se cuantifica usando un modelo psicoacústico y codificado.

- Se añaden códigos de corrección de error interno.

- La señal se almacena o transmite.

- Con el fin de evitar las muestras corruptas, se aplica a cada marco una implementación moderna del algoritmo de Luhn mod N. [Gil07].

El estándar de audio MPEG-4 no define un solo o un pequeño conjunto de esquemas de compresión de alta eficiencia, sino más bien una caja de herramientas complejas para llevar a cabo una amplia gama de operaciones de baja tasa de bits de codificación de voz a la codificación de audio de alta calidad y la síntesis de la música. El algoritmo MPEG-4 de audio abarca el rango de baja tasa de bits de codificación de voz (hasta 2 kbit/s), para audio de alta calidad de codificación (a 64 kbit/s por canal y superior). AAC ofrece frecuencias de muestreo de entre 8 kHz y 96 kHz y cualquier número de canales entre 1 y 48. En contraste con el banco de filtros híbrido de MP3, AAC utiliza la MDCT, junto con el aumento de las longitudes de ventana de 1024 o 960 puntos.

3. METADATOS EN VÍDEOS DIGITALES

Los metadatos son conocidos también como datos sobre datos [Org04]. En el campo de la tecnología multimedia permiten registrar y clasificar información relevante sobre datos de captura de la imagen o vídeo, como título, duración, formato del fichero, hora de generación, presencia o ausencia de flash, distancia de los objetos, tiempo de exposición, modelo del dispositivo, etc. Unos metadatos precisos y completos son de gran ayuda en una organización para realizar una óptima clasificación y búsqueda de imágenes y vídeos. Las imágenes digitales son almacenadas en una gran variedad de formatos como *Tagged Image File Format* (TIFF), JPEG, *Photoshop Data file* (PSD) entre otros. Cada formato de imagen tiene distintas reglas de como los distintos formatos de metadatos son almacenados junto al propio archivo que contiene la imagen. Los contenedores de metadatos que destacan el *Image File Directory* (IFD), *Exchangeable Image File Format* (EXIF), TIFF, Adobe EXIF, y *International Press Telecommunications Council* (IPTC-IIM) ASEA14}. EXIF es el contenedor más utilizado para metadatos de imágenes de cámaras digitales y la más común para realizar la identificación de la fuente. Entre los cientos de etiquetas que incluye la especificación se encuentran la marca y modelo de la cámara. Sin embargo, cabe destacar que la propia especificación no hace obligatoria su existencia en los archivos.

A su vez, los vídeos digitales son almacenados en diversos contenedores multimedia, que almacenan información de vídeo, audio, subtítulos y metadatos. Estos contenedores siguen un formato preestablecido en su especificación. En [KS16] [HPN07] indican que un contenedor multimedia está constituido generalmente por una pista de vídeo y audio.

Las pistas de vídeo y audio en su gran mayoría son comprimidas con un códec propietario de cada fabricante. Los códecs permiten a los dispositivos

descomprimir un fichero que contenga imágenes y sonido para posteriormente reproducirlo. El uso de un determinado códec conlleva una mayor o menor calidad y tamaño. En la Tabla 3.1 se muestran los contenedores de vídeo más conocidos con los códecs de vídeo y audio que soportan. Para poder separar el audio del vídeo y viceversa es necesario conocer el formato del contenedor y realizar el proceso de decodificación. Los contenedores más usados en la actualidad son el MP4 que forma parte del estándar MPEG-4 parte 14, MOV de propiedad de *Apple*, *Audio Interleave de Microsoft* (AVI) y *Matroska* MKV.

Tabla 3.1: Contenedores multimedia de vídeos

Contenedor	Propietario	Soporte de codificación de vídeo	Soporte de codificación de audio
3GP	3GPP	H.263,MPEG-4 Part 2, H.264/MPEG-4 AVC	AMR-NB, AMR-WB,AMR-WB+, AAC, HE-AAC and HE-AAC v2
3G2	3GPP2	H.263,MPEG-4 Part 2, H.264/MPEG-4 AVC	AMR-NB, AMR-WB,AAC, HE-AAC, EVRC,EVRC-B, EVRC-WB, 13K (QCELP), SMV or VMR-WB
ASF	Microsoft	VFWor DMO	ACM o DMO
AVI	Microsoft	VFW	ACM
DIVX	DivX, Inc.	MPEG-4 Part 2	MP3, PCM, AC-3
EVO	MPEG	MPEG-2 Part 2,H.264/MPEG-4 AVC, VC-1	AC-3, E-AC-3, Dolby TrueHD, Linear PCM, DTS, DTS-HD, MPEG-2 Part 3
F4V	Adobe Sys-tems	H.264/MPEG-4 AVC	MP3, AAC, HE-AAC
FLV	Adobe Sys-tems	Sorenson,VP6, H.264/MPEG-4 AVC	MP3, Nellymoser, ADPCM, Linear PCM,AAC, Speex
Matroska	CoreCodec	Virtually anything	Todos
MP4	MPEG	MPEG-2 Part 2, MPEG-4 ASP, H.264/MPEG-4 AVC, H.263, VC-1,Dirac	MPEG-2/4 (HE)-AAC, MPEG-1/2 Layers I, II, III (MP3), AC-3, Apple Lossless, ALS, SLS
MPG/MPEG	MPEG	MPEG-1,MPEG-2	MPEG-1 Layers I, II, III (mp3)
MXF	SMPTE	Virtually anything	Todos
MOV	Apple	MPEG-2, MPEG-4 Part 2, H.264, H.263, H.261, Apple ProRes, Apple Pixlet, Cinepak, , DV, DVC Pro 50, Graphics, Motion JPEG, Photo JPEG	AAC, HE-AAC, Apple Lossless, MP3, AMR Narrowband, MS ADPCM, QDesign Music 2, QCELP, IMA 4:1, MACE 3:1 S_1
RMVB	RealNetworks	RealVideo 8, 9, 10	(HE)-AAC, Cook Codec, Vorbis
VOB+IFO	DVD Forum	MPEG-2 Part 2, MPEG-1 Part 2	AC-3, Linear PCM,DTS, MPEG-2 Part 3,MPEG-1 Layer II

3.1. Átomos

Los vídeos están estructurados por átomos o cajas que sirven de soporte fundamental para el almacenamiento de toda la información necesaria para su posterior reproducción (metadatos, vídeo y sonido) [SZC+03]. Un átomo contiene una cabecera, seguido de los datos del propio átomo como se ilustra en la Figura 3.1 y estos átomos están organizados de forma jerárquica, es decir un átomo puede contener otros átomos, y éstos pueden contener otros, pero cuando un átomo no contiene otros átomos se le llama átomo de hoja o *leaf atom*, y típicamente contiene solo campos o tablas [fS03a] [Spe16].

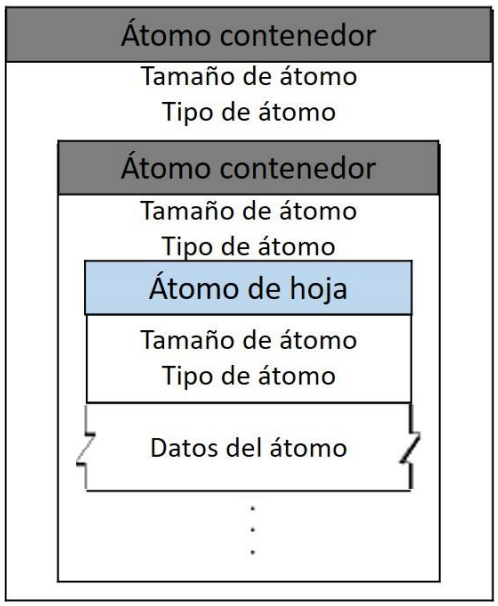


Figura 3.1: Estructura de un átomo

La cabecera contiene generalmente campos de tamaño (*size*), tipo (*type*) y excepcionalmente los campos tamaño ampliado (*extended size*), versión y banderas (*flags*). El campo tipo (*type*) del átomo está especificado por un entero sin signo de 32 bits, interpretado como un código *American Standard Code for Information Interchange* (ASCII) de cuatro caracteres generalmente en letras minúsculas. El tamaño real de un átomo no puede ser menor de 8 bytes al igual que los campos tipo y tamaño. A continuación se detalla los campos de la cabecera de un átomo.

- **Tamaño**: Es un número entero sin signo de 32 bits que indica el tamaño real del átomo representado en bytes, considerando la cabecera del átomo con su respectivo contenido e incluyendo todos los átomos contenidos. Este campo puede contener otros valores que indican un método alternativo para determinar el tamaño del átomo. Estos valores se utilizan comúnmente sólo para el átomo llamado datos de los medios o *mdat*. Uno de estos valores es el 0, que permite, sólo por un átomo de nivel superior, designar el último átomo en el archivo e indica que el átomo se extiende hasta el final del archivo. Otro valor es el 1 e indica que el tamaño real se da en el campo tamaño extendido (*extended size*).

- **Tipo**: Es un número entero de 32 bits que contiene el tipo de átomo. Esto generalmente es tratado de manera útil como un campo de cuatro caracteres con un valor nemotécnico, como el átomo *moov* (0x6D6F6F76) por un átomo de película, o átomo *trak* (0x7472616B) para un átomo de pista, pero los valores como (0x00000001) también se utilizan.

- **Tamaño ampliado**: Este indica que el tamaño de un átomo es grande y representa un entero de 64 bits. Si el campo *extended size* está presente, el campo *size* se establece en 1 y es utilizado por átomos con datos que contienen más de 232 bytes.

- **Versión y Banderas**: Estos campos no son tratados como parte de la cabecera del átomo en este documento, sino como campos de datos específicos para cada tipo de átomo que los contiene. Tales campos siempre deben ser puestos a cero, a menos que se especifique lo contrario.

Cada tipo de átomo puede contener diferentes tipos de datos que dependerán del átomo raíz que lo contiene. Conocer el tipo de un átomo permite interpretar sus datos. Estos datos sean campos, tablas o átomos no tienen la obligatoriedad de situarse en algún orden específico, pero es común que sigan un esquema parecido. La estructura de datos es específica para el tipo

de átomo. Si existiera un átomo de un tipo desconocido, no se debe intentar interpretar los datos del átomo, para esto se utiliza el campo tamaño del átomo para omitir este átomo y todo su contenido. Esto permite un grado de compatibilidad hacia adelante y poder seguir analizando el resto de átomos del vídeo. En la Tabla 3.2 se muestran los principales átomos que se pueden encontrar en un vídeo y el uso de cada uno de ellos. Estos átomos son considerados átomos raíz de un vídeo ya que no tienen ningún átomo padre.

Tabla 3.2 Principales átomos de un vídeo

Átomo	Características	Uso en contenedores
FTYP	Tipo de compatibilidad de archivo, identifica el tipo de archivo y lo diferencia de los tipos de archivos similares, tales como archivos MPEG-4 y JPEG-2000.	MP4, MOV
MOOV	Película de metadatos de recursos sobre la película (número y tipo de pistas, localización de datos de la muestra, y así sucesivamente). Describe donde se pueden encontrar y cómo se interpretan los datos de la película.	MP4, MOV
MDAT	Muestras de datos media de la muestra de películas tales como marcos y grupos de muestras de audio de vídeo. Por lo general, estos datos se pueden interpretar sólo mediante el uso del recurso de película.	MP4, MOV
FREE	Indica el espacio no utilizado disponible en el archivo. Este átomo se compone de tan sólo un encabezado de átomo (campo de tipo y tamaño), seguido por el número adecuado de bytes de espacio libre.	MP4, MOV
SKIP	Indica el espacio no utilizado disponible en el archivo. Este átomo se compone de tan sólo un encabezado de átomo (campo de tipo y tamaño), seguido por el número adecuado de bytes de espacio libre.	MOV
WIDE	Espacio reservado puede ser sobrescrito por un campo de tamaño ampliado si el siguiente átomo supera los 232 bytes, sin desplazar el contenido del siguiente átomo.	MOV
PNOT	La referencia a los datos de la película de vista previa.	MOV

3.2. Contenedor Multimedia MP4

El formato MP4 es una extensión oficial para almacenar y reproducir audio, vídeo y contenidos avanzados como subtítulos, imágenes fijas, entre otros. Este contenedor forma parte de la especificación *QuickTime File Format* (QTFF), estándares como MPEG-4 parte 14, ISO/IEC 14496-14:2003. En los dispositivos móviles modernos el contenedor MP4 generalmente utiliza el códec para vídeo

H.264/MPEG-4 AVC que forma parte de la recomendación ITU H264 (02/2016) y el estándar MPEG-4 parte 10 [SMW07], y para audio el AAC que está establecida en el estándar MPEG-4 parte 3 y la norma ISO/IEC: 2009, aunque admite archivos con compresión MP3.

3.3. Especificación del Contenedor Multimedia MP4

La gran tasa de crecimiento de vídeos producidos por dispositivos móviles en la actualidad hace que el analista forense tenga la necesidad de conocer su estructura para realizar un análisis exhausto a fin de emitir resultados precisos y solucionar temas legales. Por tanto, en el presente capítulo se ha realizado una profunda investigación y un análisis completo sobre la estructura del contenedor multimedia MP4 con códec de compresión H.264 de vídeos capturados con dispositivos móviles Android. El análisis de la estructura de un vídeo es uno de los primeros pasos y el más importante que permite obtener información relevante que sirve de evidencia de manipulación, posteriormente estos datos servirán para realizar procedimientos avanzados como extracción, edición de átomos o flujos de bits en bruto como por ejemplo la extracción del audio natural de un contenedor de vídeo con formato MP4. Los átomos para éste formato MP4 y como para otros formatos, pueden o no aparecer en un vídeo. Asimismo, estos átomos pueden seguir o no el mismo orden en vista que algunos son átomos opcionales, todo dependerá únicamente del fabricante del dispositivo determinar tanto el orden y los átomos a insertar.

A continuación se especifica la estructura y los átomos más comunes del formato MP4, pudiendo éstos aparecer o no, cambiar su orden o incluso aparecer otros átomos no especificados.

3.3.1. Átomo ftyp

Este átomo también llamado tipo de archivo o *File Type*, contiene la información de compatibilidad del archivo con otros formatos, pudiendo

contemplar más de una compatibilidad, pero es común que indique su tipo preferido. Es un átomo opcional pero muy recomendable y si aparece debe ser el primer átomo significativo del archivo. En la Tabla 3.3 se detalla la estructura del átomo *ftyp*.

Tabla 3.3: Estructura del átomo *ftyp*

Campo	C.ASCCI	Valor	Descripción
Size		32 bits	Un entero sin signo de 32 bits que especifica el número de bytes en este átomo File Type.
Type	66747970 (ftyp)	32 bits	Un entero sin signo de 32 bits que identifica el tipo de átomo, representado como un código de cuatro caracteres (0x66747970).
Major Brand		32 bits	Un entero sin signo de 32 bits que identifica el tipo de archivo de película, representado como un código de cuatro caracteres. Si un archivo es compatible con múltiples marcas, todas estas marcas se encuentran en los campos Compatible Brands, y la Major Brand e identifica la marca preferida o de mejor uso.
Minor Version		32 bits	Un entero de 32 bits sin signo que identifica el tipo de archivo de película Minor Version, representado como un número de cuatro bytes representado en forma decimal codificado en binario (BCD); lo que indica año y mes, seguido de un decimal codificado en binario cero. Por ejemplo, para el Minor Versión de julio de 2007, este campo se establece en los valores de la BCD 20 07 07 00.
Compatible Brands		Lista de 32 bits	Una serie de enteros sin signo de 32 bits, representados como un código de cuatro caracteres cada uno, que anuncia los formatos de archivo compatibles.

3.3.2. Átomo free

Este átomo muestra el espacio no utilizado en el archivo y puede sobrescribirse si fuera necesario. En la Tabla 3.4 se detalla la estructura del átomo *free*.

Tabla 3.4: Estructura del átomo *free*

Campo	C.ASCCI	Valor	
Size		32 bits	Un entero sin signo de 32 bits que especifica el número de bytes en este átomo free.
Type	66726565	32 bits	Un entero sin signo de 32 bits que identifica el tipo de

	(free)		átomo, representado como un código de cuatro caracteres (0x66726565).
Free Space			Contiene los bytes de espacio libre y todos tienen el valor 0.

3.3.3. Átomo mdat

Este átomo también llamado datos de los medios o *media data* contiene grupos de muestras de audio y vídeo. Del contenido de éste átomo como se observa en los capítulos posteriores se extraerá el audio y los fotogramas, este proceso se logrará interpretando el recurso de película, que es el átomo *moov*. En la Tabla 3.5 se detalla la estructura del átomo *mdat*.

Tabla 3.5: Estructura del átomo *free*

Campo	C.ASCCI	Valor	Descripción
Size		32 bits	Un entero sin signo de 32 bits que especifica el número de bytes en esta porción de datos media del archivo de película MP4.
Type	6D646174 (mdat)	32 bits	Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres (0x6D646174).
Data			Contiene los datos de audio y vídeo de la película.

3.3.4. Átomo moov

Este átomo también llamado película o *movie* describe dónde encontrar los datos de la película y cómo interpretarlos desde el átomo *mdat*. Una característica importante a comentar es que este átomo posee una pista de vídeo y otra de audio con el mismo nombre denominado *trak*, asimismo los sub átomos contenidos de cada uno llevan los mismos nombres. Por tanto para el uso de los algoritmos que se observan posteriormente se le distingue por un apóstrofe. En la Tabla 3.6 se detalla la estructura básica del átomo, seguidamente en la Figura 3.2 se ilustra la estructura completa y por último para un mejor entendimiento sobre el átomo *moov* en el Anexo A se muestra el análisis y descripción de los átomos *moov*.

Tabla 3.6: Estructura del átomo *moov*

Campo	C.ASCCI	Valor	Descripción
Size		32 bits	Un entero sin signo de 32 bits que especifica el número de bytes en este átomo de película.
Type	6D6F6F76 (moov)	32 bits	Un entero sin signo de 32 bits que identifica el tipo, representada como un código de cuatro caracteres (0x6D6F6F76).
Mvhd		Átomo	Átomo que contiene cabecera de la película.
Ctab		Átomo	Átomo que contiene una tabla de colores.
Trak		Átomo	Átomo que contiene los datos de audio y vídeo de la película.
Trak		Átomo	Átomo que contiene los datos de audio y vídeo de la película.
Udta		Átomo	Átomo que contiene los datos de los usuarios.

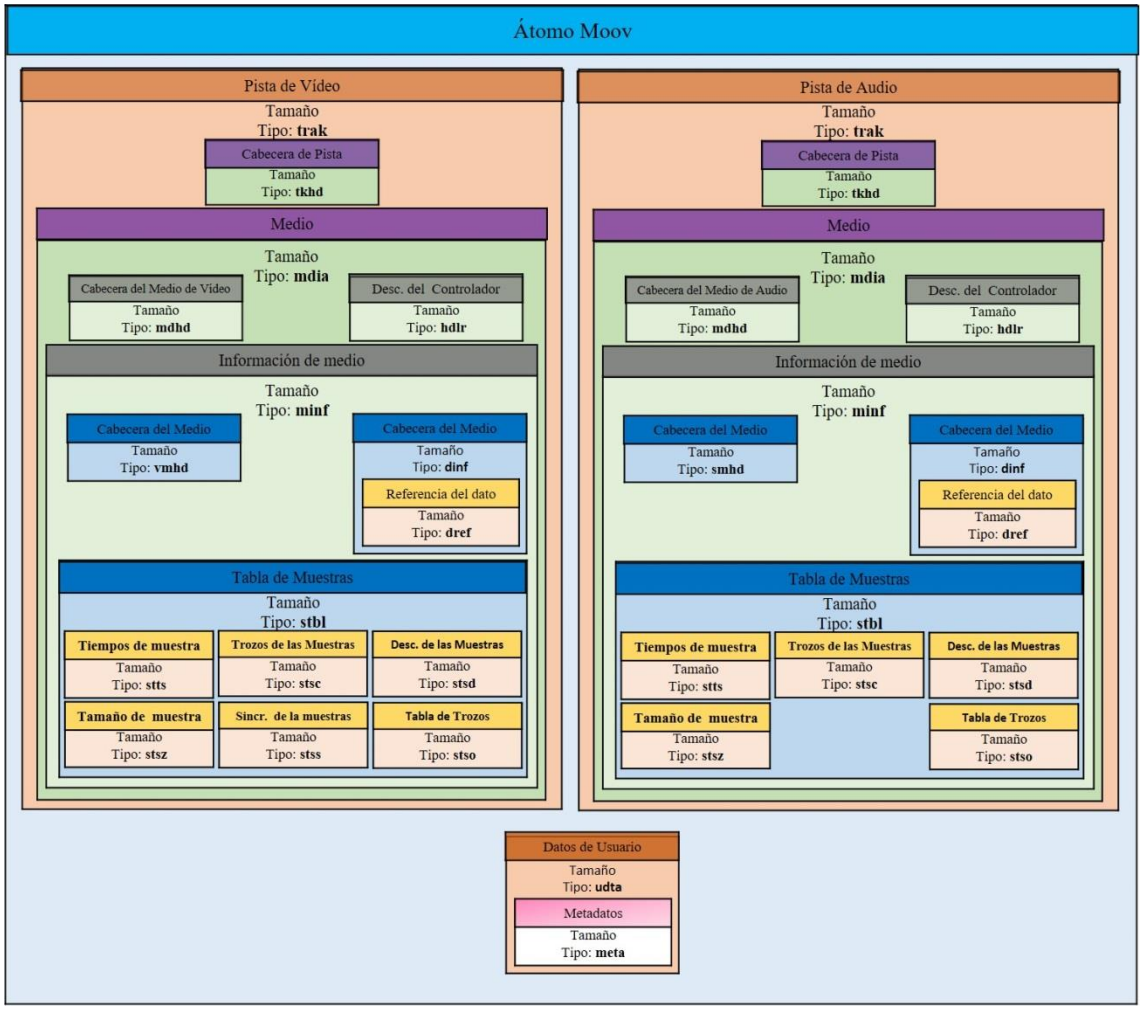


Figura 3.2: Estructura de átomos que contiene el átomo *moov*

4. ANÁLISIS FORENSE EN VÍDEOS DIGITALES DE DISPOSITIVOS MÓVILES

Este capítulo presenta el estado del arte de las técnicas de análisis forense de vídeos clasificando los trabajos relacionados en los dos grandes grupos: técnicas de análisis forense de vídeos clasificados según su objetivo y técnicas anti-forenses en vídeos. Se comienza con la exposición de los trabajos relacionados referentes a la identificación de la fuente de adquisición de vídeos digitales. Seguidamente se muestran los trabajos relacionados referentes al análisis de compresión de un vídeo la identificación de la fuente de adquisición de vídeos digitales. Posteriormente se exponen los trabajos referentes al análisis de manipulación de un vídeo. Finalmente, se presentan las técnicas anti-forenses en vídeos digitales. Cabe destacar que aunque este trabajo este centrado en los dispositivos móviles, en el estado del arte se aúnan referencias a técnicas sobre vídeos de todo tipo de dispositivos, ya que su conocimiento puede ser válido para la aplicación o adaptación a vídeos de dispositivos móviles.

4.1. Importancia del Análisis Forense

La necesidad de realizar un análisis forense en dispositivos móviles surge a partir de las bondades y características tecnológicas que pueden ofrecer este tipo de dispositivos. Por ejemplo, la facilidad de implementar aplicaciones para estos dispositivos en poco tiempo [AZ06], la capacidad de almacenar, editar, eliminar e imprimir documentos electrónicos, el uso de mensajes de texto, mensajes multimedia y conversaciones a través de aplicaciones de redes sociales (lo más requerido en la actualidad), el uso de plataformas en línea para realizar operaciones bancarias, compras a través de la web y todas aquellas operaciones realizadas con datos sensibles. Por tanto, hoy en día, los dispositivos móviles son elementos que proveen información útil que deberá tratarse con cautela y precisión.

En cuanto al análisis forense en vídeos de dispositivos móviles no hay duda de la importancia que puede tener su aplicación en casos judiciales. Los dispositivos móviles pueden contener vídeos almacenados de carácter personal o con contenidos de delitos flagrantes. Todos estos tipos de vídeos pueden ser evidencias de un hecho y elementos potenciales de uso en procesos judiciales y, consecuentemente, elementos de estudio del análisis forense.

4.2. Técnicas de Análisis Forense

Según [BFM+12], las técnicas forenses en vídeos se agrupan en herramientas forenses según el objetivo a cumplir: herramientas forenses para el análisis de adquisición, compresión y falsificación de un vídeo.

4.2.1. Herramientas Forenses para el Análisis de Adquisición

El análisis de adquisición de imágenes es uno de los primeros problemas que surgieron en la ciencia forense multimedia, que tiene como objetivo identificar la fuente u origen de una imagen. Este análisis captura información como tipo de dispositivo (cámara, escáner, ordenador, etc.), marca y modelo del dispositivo usado. A la fecha las investigaciones sobre técnicas de análisis de adquisición en imágenes se han realizado a más profundidad en comparación que las de vídeo digital que no alcanza aún un estado de madurez.

La mayoría de las técnicas que se pueden aplicar a una imagen se pueden emplear con los diferentes fotogramas de un vídeo. En [SOAGRC+13] se realiza una comparación detallada de los principales grupos de técnicas de identificación de fuente de adquisición. Estas se dividen en cinco grupos y están basadas en: metadatos, características de la imagen, defectos de la matriz CFA e interpolación cromática, imperfecciones del sensor y las transformadas *wavelet*.

En cuanto a la técnica basada en metadatos, es la más sencilla de analizar, aunque depende en gran medida de los datos que inserta el fabricante.

Asimismo la agregación de metadatos a la imagen no es obligatoria. En [RCC+08][BL05][Tes05][BL04] se utilizan los metadatos con fines de clasificación de imágenes digitales.

Estas técnicas utilizan un conjunto de características extraídas del contenido de la imagen para hacer identificar la fuente. Estas características se dividen en tres grupos: características de color, métricas de calidad de la imagen o *Image Quality Metrics* (IQM) y estadísticas del dominio *wavelet*. En [TLL07] se identifica la fuente utilizando tres tipos de características de la imagen: características de color, características de calidad o IQM y características de la imagen en el dominio de la frecuencia. La clasificación de las imágenes se realiza a través de una Maquina de Soporte Vectorial o SVM. El resultado obtenido para una clasificación de cuatro cámaras de dos fabricantes distintos con contenidos similares en la imágenes fue del 100%, mientras que para la clasificación de imágenes con contenidos distintos entre sí fue 93.05%.

En cuanto a la técnica de defectos de la matriz CFA y la especificación de los algoritmos de interpolación cromática algunos autores precisan que generan ciertas diferencias marcadas entre los diferentes modelos de cámaras [BSM06] [CAS+06] [LH06] [BSM08a]. En [CAS+06] [CAS+06] se utiliza una técnica que se basa en los algoritmos propietarios de interpolación cromática, los cuales dejan correlaciones a través de los planos de bits adyacentes de una imagen. Estos pueden ser representados mediante un conjunto de 108 métricas de similitud binarias y 10 métricas de calidad de la imagen IQM. Con un clasificador *k-Nearest Neighbors* (KNN) se realizan experimentos utilizando 9 cámaras de teléfonos móviles y 200 fotos de cada una. Para el entrenamiento se utilizaron 100 fotos de cada cámara y las 100 restantes para las pruebas. Se obtuvo un rendimiento promedio del 93,4% de 16 experimentos que se realizaron. Hay diversos grupos de investigación que han aportado en esta área, en donde se presentan buenos resultados, por ejemplo en [LH06] [BSM08a].

Dentro de los métodos existentes que se basan en las imperfecciones del sensor, hay dos grandes ramas de las cuales se pueden trabajar: defectos del pixel o patrón de ruido del sensor. En [LFG06] se demostró que los sensores de las cámaras generan un patrón de ruido *Sensor Pattern Noise* (SPN) que podría ser utilizado como método único de identificación. En [Li10] se demostró que el ruido del sensor extraído de las imágenes podría ser severamente contaminado por los detalles de las escenas concretas. Para lidiar con ese problema, se propuso un nuevo enfoque para la atenuar la influencia del detalle de las escenas en el ruido del sensor mejorando la tasa de acierto. En los experimentos se tomaron 9 cámaras y 320 fotos de cada una, variando las escenas al aire libre e interiores. En [GBK+01][VCEK07][CESR12] se presentan otros métodos de identificación de fuente basados en las imperfecciones del sensor.

Por último, en el área de la transformada *wavelet* existen diversos enfoques. Por ejemplo en [WHL12] se propone una nueva técnica de identificación basada en las características de probabilidad condicional. Este tipo de características fueron propuestas inicialmente para detectar mensajes ocultos en imágenes [WBSH09]. Se obtuvieron unos resultados del 98,6%, 97,8% y 92,5% de acierto en la clasificación de 2, 3 y 4 IPhones respectivamente con un recorte de imagen de 800x600. En [RCAGSO+13] se determina que el uso del patrón de ruido del sensor conjuntamente con la transformada *wavelet* es un método efectivo para la identificación de fuente, alcanzando una tasa de éxito promedio del 87,21%.

En el caso del desarrollo de técnicas para la identificación de fuente de vídeo, existen pocas referencias al respecto. Algunas se basan directamente en la secuencia de codificación y otras en la extracción de fotogramas aplicando algún método de clasificación para imágenes fijas.

En [SXD09], se propone un algoritmo en base a la información del vector de movimiento en el flujo codificado. En los experimentos realizados se utilizaron 100 secuencias de vídeo, de las cuales 20 de ellas procedentes de *Video Quality*

Experts Group (VQEG) y 80 de disco versátil digital o *Digital Versatile Disc* (DVD). Todos los vídeos fueron codificados por diferentes aplicaciones de edición de vídeo conocidos. Mediante un experimento se obtuvo un 74,63% de precisión en la identificación del software que se utilizó en la codificación.

En [YHW12] propone un método de identificación utilizando los fotogramas extraídos de vídeos. Las características de probabilidad condicional se extraen directamente de los fotogramas del vídeo. En las pruebas realizadas se utilizaron 4 modelos diferentes de cámaras y un clasificador SVM, obteniendo, en un primer experimento aplicado en el dominio del espacio con los valores de luminancia, un 82,6% de precisión. En un segundo experimento usando el mismo conjunto de vídeos, tomando el valor de luminancia, el promedio de clasificación fue de 100%. En un tercer experimento en donde se utilizaron un conjunto de vídeos con mayores cambios en las escenas se obtuvo un 97,2% de acierto.

Un problema importante en la protección de los derechos de autor es la proliferación de vídeos piratas. En muchos casos las copias ilegales de películas se publican en Internet incluso antes de su lanzamiento oficial. Una gran parte de estas copias falsas se producen mediante la grabación de películas con cámaras de vídeo en salas de cine. Las técnicas que forenses que contribuyen a hacer frente a estos problemas son: las técnicas de detección de re-adquisición y las técnicas de detección de copia.

Las técnicas de detección de re-adquisición se usan en vídeos capturados a partir de una secuencia de vídeo que se reproduce en una proyección. Para hacer frente a la re-adquisición se proponen diversos enfoques basados en marca de agua, tanto para la identificación del vídeo pirata [LKL+08] como para localizar la posición del pirata en salas de cine [LKL12]. En [WF08] se muestran muchos experimentos realizados dando buenos resultados; estos vídeos re-adquiridos se detectaron con un 88% de precisión y con un 0,4% de

probabilidad de falsos positivos.

En cuanto a la de detección de copia del vídeo, es común extraer características más destacadas del contenido visual que no dependen del dispositivo utilizado para capturar el vídeo. Sin embargo, en [BSM08b] se señala que las firmas sólidas basadas en contenidos pueden obstaculizar la capacidad de distinguir entre los vídeos que son similares, aunque no son copias de uno al otro. Por esta razón, se propone utilizar las características del dispositivo fuente extraídas de los vídeos para la construcción de una técnica de detección de copia. En [BSM08b], una firma de vídeo se obtiene mediante la estimación de las huellas digitales PRNU de videocámaras involucradas en la generación del vídeo. Como consecuencia, esto produce automáticamente una media ponderada de los diferentes patrones PRNU, en el que el mayor número de fotogramas tomados con la misma cámara se traducirá en un mayor peso asignado a la misma. Los autores también muestran que la huella digital es robusta frente a un conjunto de operaciones de tratamiento común, es decir, la mejora del contraste, desenfoque, irregularidades de la imagen, subtítulos, ajuste de brillo, de compresión, etc. Los experimentos realizados en vídeos descargados de YouTube muestran una tasa de detección del 96% y una probabilidad de falsa alarma de 5%.

4.2.2. Herramientas Forenses para el Análisis de Compresión

El contenido del vídeo suele estar disponible en un formato de compresión con pérdidas debido a la velocidad de bits de gran tamaño y es necesario representar las imágenes en movimiento, ya sea en un formato no comprimido o sin pérdidas. La compresión con pérdida deja huellas características, que pueden ser detectados por el analista forense. Las arquitecturas de codificación del vídeo son más complejas que las adoptadas para imágenes. La mayoría de los estándares de codificación ampliamente utilizados como MPEG o las familias H.26x, heredan el uso de codificación en bloque por la transformada de

la norma JPEG. Sin embargo, la arquitectura de codificación es complicada por varios procesos adicionales, por ejemplo, la predicción espacial y temporal. Cuando cada trama se considera como una sola imagen, es posible aplicar técnicas de análisis forense basadas en las imágenes. Sin embargo, para permitir un análisis más completo, es necesario tener en cuenta las operaciones de codificación a lo largo de la dimensión temporal. A continuación, se detallan técnicas forenses destinadas a la codificación del contenido del vídeo.

La elección de los parámetros de codificación depende de la implementación específica del códec y de las características de la señal codificada. En la compresión de vídeo, el número de parámetros de codificación que se pueden ajustar es significativamente amplio. Como consecuencia de ello, el analista forense debe tener en cuenta un mayor grado de libertad cuando se detecta la identidad del códec. Esta información podría permitir identificar las implementaciones de otros proveedores que dependen de los códecs de vídeo. Los métodos con el objetivo de estimar diferentes parámetros de codificación y elementos de sintaxis que caracterizan a cada códec se pueden agrupar en tres categorías principales: detección de bloques, detección de parámetros de cuantificación e identificación de vectores de movimiento.

En [HS09] se calcula el tamaño de bloque en una secuencia de vídeo comprimida mediante el análisis de la imagen reconstruida en el dominio de la frecuencia y la detección de los picos que están relacionados con las discontinuidades en los límites de bloque, en lugar de las características intrínsecas de la imagen subyacente.

Cuando un vídeo ha sido comprimido y posteriormente se edita la escala, brillo, etc., el vídeo en mención es re-comprimido. Las técnicas para detectar las huellas dejadas por la doble compresión de vídeo se han enfocado en el estándar MPEG y explotan las mismas ideas usadas para la doble compresión JPEG. En [LWH08] se propone un método para la detección de doble

compresión de MPEG basada en el bloqueo de los artefactos. Se define un sistema de medición para calcular el *Block Artifact Strength* (BAS) de cada trama. La media BAS se calcula para la eliminación de secuencias obtenidas a partir de 1 a 11 tramas, obteniendo un vector de características de los valores BAS. Si la secuencia se ha manipulado previamente, sea eliminación de fotograma o re-compresión, el vector de características presenta un comportamiento característico.

Aunque la detección de doble compresión de las imágenes es un tema ampliamente investigado, la compresión doble de vídeo aún es un problema de investigación actual, debido a la complejidad y diversidad de arquitecturas de codificación de vídeo.

Cuando se transmite un vídeo a través de un canal ruidoso, éste deja huellas en el contenido del vídeo. Un claro ejemplo son las pérdidas de paquetes y errores que pueden afectar el flujo de bits que se recibe. Por tanto, algunos datos codificados harán falta o en su defecto estarán dañados.

En [RP07], los autores presentan un algoritmo basado en varias métricas de evaluación de calidad para estimar la pérdida de paquetes en el vídeo reconstruido. Sin embargo, la solución propuesta adopta métricas de calidad de referencia completa que requieren la disponibilidad de la secuencia de vídeo original sin comprimir. La solución propuesta en [GMT10] se basa en la estimación de calidad no-referenciada pero se lleva a cabo sin tener en cuenta la disponibilidad de la corriente de bits. Por lo tanto, la solución propuesta procesa sólo los valores de los píxeles, identificando las rebanadas que se perdieron del vídeo y produciendo un valor que presenta una correlación con el valor *Mean Square Error* (MSE). El método supone que las rebanadas se corresponden con las filas de macrobloques.

4.2.3. Herramientas Forenses para el Análisis de Manipulaciones

A pesar de ser más complicado que para las imágenes, la creación de un vídeo falsificado o manipulado actualmente es más fácil que antes, debido a la disponibilidad de los medios. Existen muchas maneras diferentes de manipulación de un vídeo como la sustitución o eliminación de algunos fotogramas, la replicación de un conjunto de fotogramas y la inserción o eliminación de objetos de la escena. A continuación se presentan algunas técnicas para detectar la falsificación o manipulación de los vídeos.

Los dispositivos móviles suelen dejar una huella característica en los vídeos grabados. Aunque estas huellas normalmente son explotadas sólo para la identificación de dispositivos, algunas investigaciones como [MCP+07] [HHLH08] [KTY10] han sido direccionadas para detectar las manipulaciones.

En [MCP+07] se propone una técnica basada a PRNU, en concreto a las secuencias de vídeo. El patrón PRNU característico de los dispositivos se estima en los primeros fotogramas de vídeo, y se utiliza para detectar varios tipos de ataques. Los autores evalúan el coeficiente de tres correlaciones y cada uno de estos coeficientes de correlación es el umbral para obtener un evento binario, y diferentes combinaciones de eventos permiten detectar diferentes tipos de manipulación, entre los cuales están la inserción de fotogramas, inserción de objetos dentro de un fotograma (cortar y pegar), la replicación de fotogramas, etc. Los experimentos se realizaron en vídeos MPEG sin comprimir; los resultados muestran que el método es fiable aunque solo informa de algunos estudios de casos, no los valores promediados.

La codificación del vídeo introduce artefactos que pueden ser aprovechados para investigar la integridad del contenido. Estos artefactos son notables en el contenido. En los últimos años, algunos analistas forenses investigan la presencia o inconsistencia de estos artefactos para evaluar la integridad de un vídeo y para localizar las regiones que no son originales. En [WF06], se exploran

dos fenómenos en vídeos MPEG comprimidos, uno estático es decir entre fotogramas y una temporal es decir intra-fotogramas. El fenómeno estático se basa en el hecho de que un vídeo MPEG manipulado casi seguro es comprimido dos veces, el primero que se realiza cuando se crea el vídeo, y la segunda cuando el vídeo se vuelve a guarda después de ser manipulada. Los fenómenos temporales se basan en la estructura GOP de archivos MPEG. Cuando un vídeo se vuelve a comprimirse después de la eliminación o la adición de un grupo de fotogramas, se producirá una desincronización en el patrón de GOP. La literatura indica que aún quedan muchos aspectos por descubrir en la detección de manipulación basada a la codificación de vídeos. Esto se debe a que los algoritmos de codificación de vídeo son mucho más complejos que la compresión JPEG. Esto hace que la detección de artefactos introducidos sea más difícil, ya que los modelos matemáticos no son fáciles de obtener.

Como se ha podido evidenciar es relativamente difícil entender la geometría y las propiedades físicas de una escena cuando es consistente o no. Para un analista forense sería más asequible comprobar las consistencias en una imagen. Lo contrario ocurre al comprobar las consistencias geométricas de un vídeo que contiene una gran cantidad de fotogramas. En [COF12] se propone un algoritmo para detectar trayectorias físicamente inverosímiles de objetos en secuencias de vídeo. La idea es modelar la trayectoria parabólica tridimensional de objetos en vuelo libre, es decir, como cuando una pelota vuela o desplaza hacia la canasta y la correspondiente proyección bidimensional en el plano de la imagen. El objeto volador se extrae del vídeo, compensando el movimiento de la cámara si es necesario. Después el movimiento en el espacio 3D se estima a partir marcos de 2D y se compara con una trayectoria satisfactoria. Si la desviación entre las trayectorias observada y esperada es grande, el objeto se clasifica como manipulado. Aunque el análisis es un escenario muy específico, el método hereda todas las ventajas que caracterizan técnicas forenses basadas

en aspectos físicos y geométricos; por ejemplo, el rendimiento no depende de la compresión y la calidad de vídeo.

Los ataques de copiar - mover en un vídeo afectan a los niveles intra e inter-fotograma. Un ataque de copiar - mover intra-fotograma es conceptualmente idéntico a las de las imágenes, y consiste en la replicación de una parte del fotograma en el propio fotograma, cuyo objetivo normalmente es ocultar o reproducir un objeto. Un ataque de copiar - mover inter-fotograma, consiste en la sustitución de algunos fotogramas con una copia de los anteriores, por lo general para ocultar algo entró en la escena en el vídeo original. En [WF07] se realiza la detección el procedimiento de copiar-mover en un vídeo. Para ello el vídeo se divide en sub partes y se calculan diferentes tipos de coeficientes de correlación con el fin de resaltar las similitudes entre las diferentes partes de la secuencia. También se efectúa un método para detectar la duplicación de regiones, tanto para la inter-fotograma e intra-fotograma. Los resultados logran una precisión superior al 90% para una cámara fija, y 80% para una cámara en movimiento.

4.3. Técnicas Anti-Forenses

En la actualidad un vídeo puede ser fácilmente manipulado por un atacante gracias a la infinidad de técnicas y software de edición existentes. Para hacer frente a este problema surgen las estrategias forenses enfocadas a los vídeos para detectar el origen, la manipulación y verificación de la autenticidad. Pero cuando el atacante tiene mayor conocimiento y un objetivo concreto, puede usar y crear técnicas anti-forenses sofisticadas para manipular vídeos sin dejar rastro alguno, es decir ocultar la huella del procedimiento realizado sobre el vídeo con la finalidad de engañar al analista forense y conllevar a tomar decisiones no acertadas. En este sentido, la ciencia forense debe hacer frente a las técnicas anti-forenses. Como se comentó en la sección 1.1, la forma de combatir a las técnicas anti-forenses es conocerlas en profundidad. Sin

embargo, esto se logra implementando dichas técnicas, y así aplicar medidas de prevención a la hora de desarrollar técnicas capaces de detectar este tipo de operaciones anti-forenses.

Las investigaciones realizadas sobre el análisis anti-forense se ha realizado en gran parte sobre imágenes fijas, pero alguna de las técnicas se pueden aprovechar para el análisis de los vídeos a partir de la extracción de fotogramas. Las técnicas anti-forenses dirigidas a vídeos son relativamente escasas por diversos factores como la complejidad y diversidad de arquitecturas de codificación del vídeo.

En cuanto a técnicas anti-forenses dirigidas a un vídeo, en [PPML15] se estudian técnicas forenses y anti-forenses aplicables a los procesos de edición de vídeos con codificación H.264/AVC. El trabajo se basa en el hecho de que la excesiva predicción residual en la codificación puede aparecer en los fotogramas que se han codificado con predicción intra e inter. En el primer método implementado los autores evalúan el resultado residual después del filtrado de desbloqueo para revelar las operaciones de edición realizadas. En el segundo, se utiliza un mecanismo de control de frecuencia para comprobar los parámetros de cuantificación. En [SRL11] se propone una técnica anti-forense para suprimir la huella digital temporal que surge en secuencias de vídeo MPEG cuando se agregan o eliminan fotogramas después de la re-compresión. Para ejecutar esta técnica inicialmente los autores identificaron las propiedades de la huella digital temporal. Estas propiedades fueron utilizadas para modelar el efecto de supresión y adición de fotogramas. La técnica funciona aumentando del error de predicción en ciertos *P-frames* del vídeo, mediante el establecimiento de los vectores de movimiento de algunos macrobloques dentro de ese fotograma a cero; para posteriormente calcular el error de predicción del fotograma. Después de realizar sus experimentos los autores consiguen eficientemente eliminar la huella digital temporal de los vídeos MPEG que se han sometido a una eliminación o adición de fotogramas.

En cuanto a imágenes fijas, en [RC13] [GKWB07] realizan una clasificación de los ataques a las técnicas de análisis forense de imágenes según el objetivo a cumplir: (1) camuflaje de post-procesamientos maliciosos sobre la imagen, (2) destrucción de la identificación correcta del origen de la imagen y (3) falsificación del origen de imagen.

4.3.1. El Camuflaje de Post-Procesamientos

El objetivo principal de estas técnicas es camuflar algún proceso que al que haya sido sometido una imagen mediante el análisis de rasgos que dejan dichos procesos para ser contrarrestados. Existe investigaciones enfocados a estos rasgos de los algoritmos en imágenes como [PF05] [LF03] [LF03] [CSS07] [GKWB07].

En [GKWB07] se presenta una técnica para ocultar el proceso de re-muestreo *resampling*. El re-muestreo es el redimensionamiento con interpolación de las imágenes. Para detectar el re-muestreo los algoritmos realizan una búsqueda de dependencias sistemáticas y periódicas entre píxeles vecinos. Para ocultar el re-muestreo se elimina las equidistancias periódicas insertando distorsiones geométricas. El ataque consiste básicamente en generar una imagen (y) a partir de una imagen (x) aplicando el re-muestreo, para esto se calcula el componente de baja frecuencia de (x), se calcula el componente de alta frecuencia, y por último se obtiene un la imagen (y) que es la suma de los dos pasos anteriores. Los autores obtuvieron una tasa de falsos positivos *False Acceptance Rate* (FAR) inferior al 1%.

4.3.2. Manipulación de la Identificación de la Fuente

Imaginemos que un analista forense utiliza una técnica que extrae el ruido del sensor de la imagen para identificar la fuente, un contraataque a esta técnica sería la eliminación del ruido del sensor de la imagen, que se podría complementar con la sustitución del ruido del sensor de otra cámara. La

manipulación de la identificación de la fuente, según [RC13] se puede dar a través de dos procedimientos: (1) la destrucción de la identidad de una imagen, y (2) la falsificación de identidad de la imagen.

Para la destrucción de la identidad de una imagen existe métodos como la resta de las características del dominio *wavelet* que no es suficiente para eliminar el ruido de una imagen y deja rasgos visibles en la imagen [GKWB07]. Otro método de eliminación del ruido de una imagen es la corrección de sensibilidad o *flatfielding*, usado en la astronomía y escaneado de planos para mejorar la calidad de las imágenes. Esta corrección de sensibilidad se ejecuta utilizando el ruido de patrón fijo *Fixed Pattern Noise* (FPN) y el ruido de respuesta no uniforme PRNU. En [LFG06] [GKWB07], los autores manifiestan que los atacantes pueden evitar la identificación correcta de la fuente por el hecho de existir la posibilidad de eliminar y extraer la huella de una imagen.

Para la falsificación de la identidad de una imagen también se puede eliminar el ruido utilizando la técnica de corrección de sensibilidad, se puede incrustar el ruido de la imagen de otra cámara diferente mediante la corrección de sensibilidad inversa con la ecuación 4.1 [GKWB07].

$$\bar{y} = \bar{x} \cdot f_{falsa} + d_{falsa} \tag{4.1}$$

Donde, f_{falsa} y d_{falsa} corresponden a la cámara a plagiar y $\bar{x}$ es la imagen original sin ruido.

En [SLFK10] se propone el Algoritmo 1 para falsificar la identidad de una cámara.

Algoritmo 1: Falsificación de la identidad de una cámara

1. Calcular el promedio de las huellas *F(C1)* de la cámara *C1* con la que se atacará;
2. Tomar una fotografía *P* con la segunda cámara *C2*;
3. Sumar *F(C1)* a la fotografía P;

Si las dimensiones de ambas imágenes no coinciden se recorta o reconstruye. Adicionalmente en el Algoritmo 2 se propone una mejora al algoritmo de falsificación anterior para enmascarar los rasgos de la cámara *C2*.

Algoritmo 2: Falsificación de la identidad de una cámara para imágenes con dimensiones diferentes

1. Calcular el promedio de las huellas *F(C1)* de la cámara *C1* con la que se atacará;
2. Calcular el promedio de las huellas *F(C2)* de la cámara *C2*;
3. Sumar *F(C1)* a la fotografía *P*;
4. Tomar una fotografía P con la cámara *C2*;
5. Restar F*(C2)* a P;

Cuando se resta *F(C2)* se intenta eliminar la correlación entre la fotografía *P* y la cámara *C2*.

4.3.3. Detección de Falsificación de la Identidad de una Imagen

Según [SLFK10] [GFC11] es posible detectar un ataque de falsificación de la huella de una imagen y la inyección de ésta en otra imagen, mediante el análisis de las diferencias entre las propiedades de un patrón de ruido copiado. En la Figura 4.1 se presenta diagrama de procesos de un escenario de ataque basado a las ideas de [RC13], con la finalidad de explicar el proceso de detección de la falsificación de identidad de una imagen.

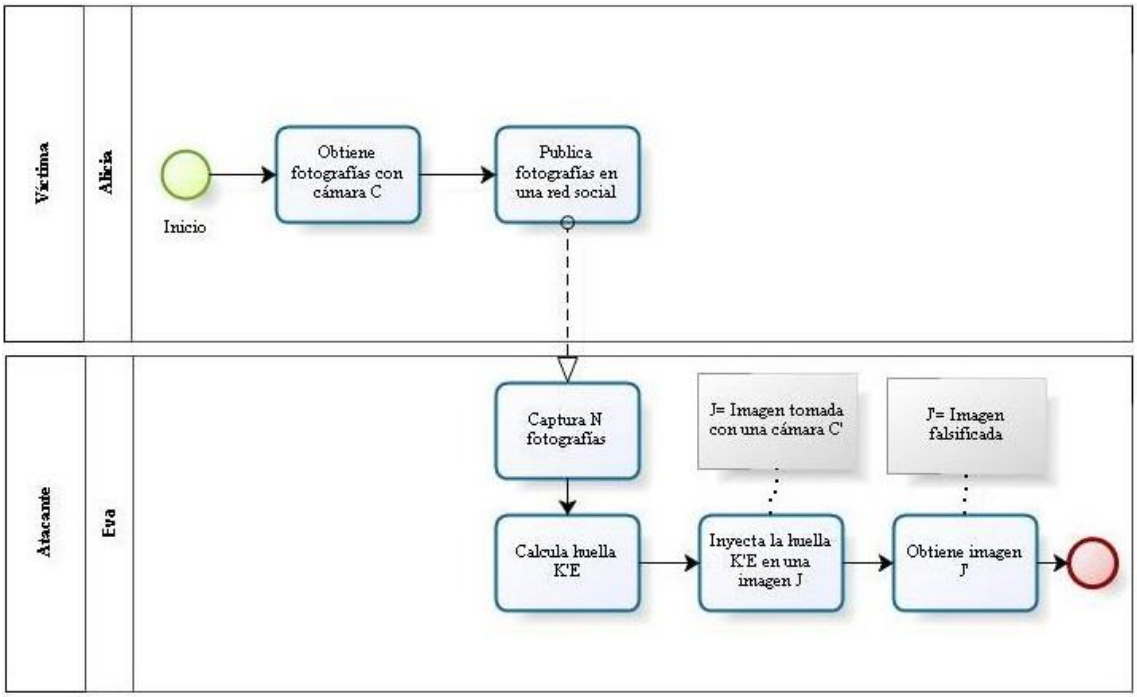


Figura 4.1: Escenario de ataque de falsificación de identidad

Alicia, la víctima publicó sus fotografías tomadas con su cámara ***C*** en una red social como muchas personas lo hacen en la actualidad, pero no se imaginó que **Eva** la atacante, descargaría sus ***N*** fotografías para calcular la huella ***K'E*.** **Eva** inyecta la huella obtenida ***K'E*** en una imagen ***J***, que fue tomada con una cámara ***C'***. Producto de los procesos anteriores se genera una imagen falsificada ***J'***. La finalidad de **Eva** al realizar este trabajo es hacer creer que **Alicia** fue la que generó la imagen falsificada.

En un plano legal, en caso que la imagen falsificada este inmiscuido en un delito, **Alicia** podría ser acusada y condenada. Los recursos para su defensa claramente serían un conjunto de fotografías ***C***, que son obtenidas de la cámara de **Alicia**, otro conjunto ***F*** de fotografías que **Eva** robó, y algunas de su propiedad.

En [GFC11] [RC13] se propone un escenario de defensa de Alicia, que comúnmente lo realiza una analista forense. Para mejor detalle en la Figura 4.2 se presenta el mencionado escenario.

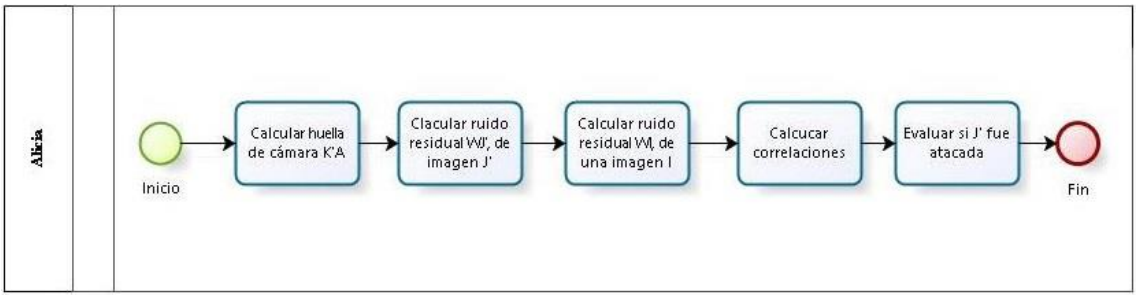


Figura 4.2: Escenario de defensa de falsificación de identidad

Alicia deberá calcular la huella de su cámara ***K'A*** utilizando imágenes planas inocentes para obtener una mejor estimación de la huella. Después calculara el ruido residual **PRNU** ***WJ'*** de la imagen ***J'***. Seguidamente calculara el ruido residual **PRNU** ***WI*** de una de las imágenes ***I*** utilizadas por **Eva** para realizar el ataque. Posteriormente **Alicia** calcula las correlaciones según la Tabla 4.1. Por último se evaluara si ***J'*** fue atacada realizando la prueba del triángulo como se muestra en la Figura 4.3, que se basa en el hecho de que el valor de correlación ***CI,J'*** de las imágenes ***I*** que no fueron utilizadas para falsificar ***J'*** puede ser estimado de las correlaciones ***CI,K'A*** y ***CJ',K'A***. En caso de que la imagen ***I*** haya sido utilizada para la falsificación la correlación ***CI,J'*** tendría un valor mayor que ***CI,K'A*** y ***CJ',K'A***.

Tabla 4.1: Calculo de correlaciones

Formula
$CI,J'=\text{corr}(WI, WJ')$
$CI,K'A=\text{corr}(WI, K'A)$
$CJ',K'A=\text{corr}(WJ',K'A)$

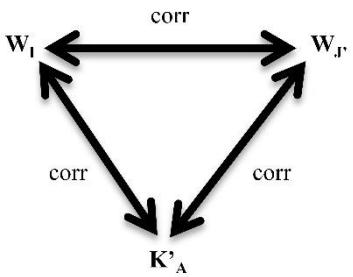


Figura 4.3: Correlaciones de la prueba del triángulo

En resumen esta propuesta refuerza los elementos necesarios para no culpar a alguien inocente, aunque el contraataque no es al 100% efectivo, porque los autores malintencionados pueden generar imágenes ilegales con características similares a las fotografías de la víctima.

5. CONTRIBUCIONES

El objetivo de este capítulo es presentar las contribuciones de este trabajo que consiste en dos técnicas anti-forenses: una para realizar la anonimización de un vídeo con formato MP4 y otra para realizar la falsificación de la fuente de un vídeo con formato MP4. Para la comprensión de los algoritmos, se presentan algunas consideraciones generales del funcionamiento de las técnicas anti-forenses propuestas, para luego detallar el funcionamiento interno de cada algoritmo mediante su flujo de procesos: extracción del flujo elemental AAC (audio), extracción de fotogramas desde los átomos del vídeo con formato MP4, extracción de la huella del sensor basado a PRNU y recomposición del vídeo modificado.

5.1. Consideraciones Generales

La lectura y análisis de los componentes básicos de un vídeo con formato MP4, denominados átomos, permiten inicialmente el mapeo de la información (vídeo, audio o metadatos) que posteriormente se utilizan para realizar correctamente la descomposición, anonimización y falsificación del vídeo. Cabe señalar que en la actualidad existen pocas investigaciones que hacen uso de los átomos para estos fines. Esto se debe al carácter de información estrictamente confidencial que le dan los fabricantes y al hecho de ser un tema relativamente complejo.

El patrón del ruido del sensor en imágenes y vídeos se define como los rasgos que se impregnan en el proceso de generación, que se puede usar como medio de identificación de la fuente [LFG06]. En base al análisis realizado de las diferentes técnicas para la anonimización y falsificación de la fuente se detectó que para el caso particular de los dispositivos móviles las técnicas más adecuadas son las basadas en el ruido del sensor por el tipo de sensor que

utilizan. Sin embargo, la mayoría de los trabajos realizados se han enfocado únicamente en cámaras tradicionales olvidándose de las cámaras de dispositivos móviles.

En función a las consideraciones anteriores, este trabajo propone implementar dos técnicas anti-forenses que permiten la anonimización y falsificación de la fuente en vídeos MP4. Las técnicas están basadas a la descomposición del vídeo, imperfecciones en el sensor y la transformada *wavelet*. En la Figura 5.1 se muestra gráficamente el proceso general para efectuar la anonimización y falsificación de la fuente en vídeos con formato MP4.

5.2. Anonimización de un Vídeo con formato MP4

El algoritmo de anonimización tiene como objetivo eliminar toda la información del sensor de la cámara de dispositivo móvil presente en los fotogramas del vídeo para impedir que se identifique el dispositivo móvil usado para grabar un vídeo determinado. El algoritmo se compone, a su vez, de tres procesos: (1) un algoritmo de extracción de fotogramas, (2) un algoritmo de extracción del flujo elemental AAC (audio), y (3) un algoritmo de eliminación del ruido de cada fotograma extraído del vídeo. El Algoritmo 3 muestra el pseudocódigo del mismo.

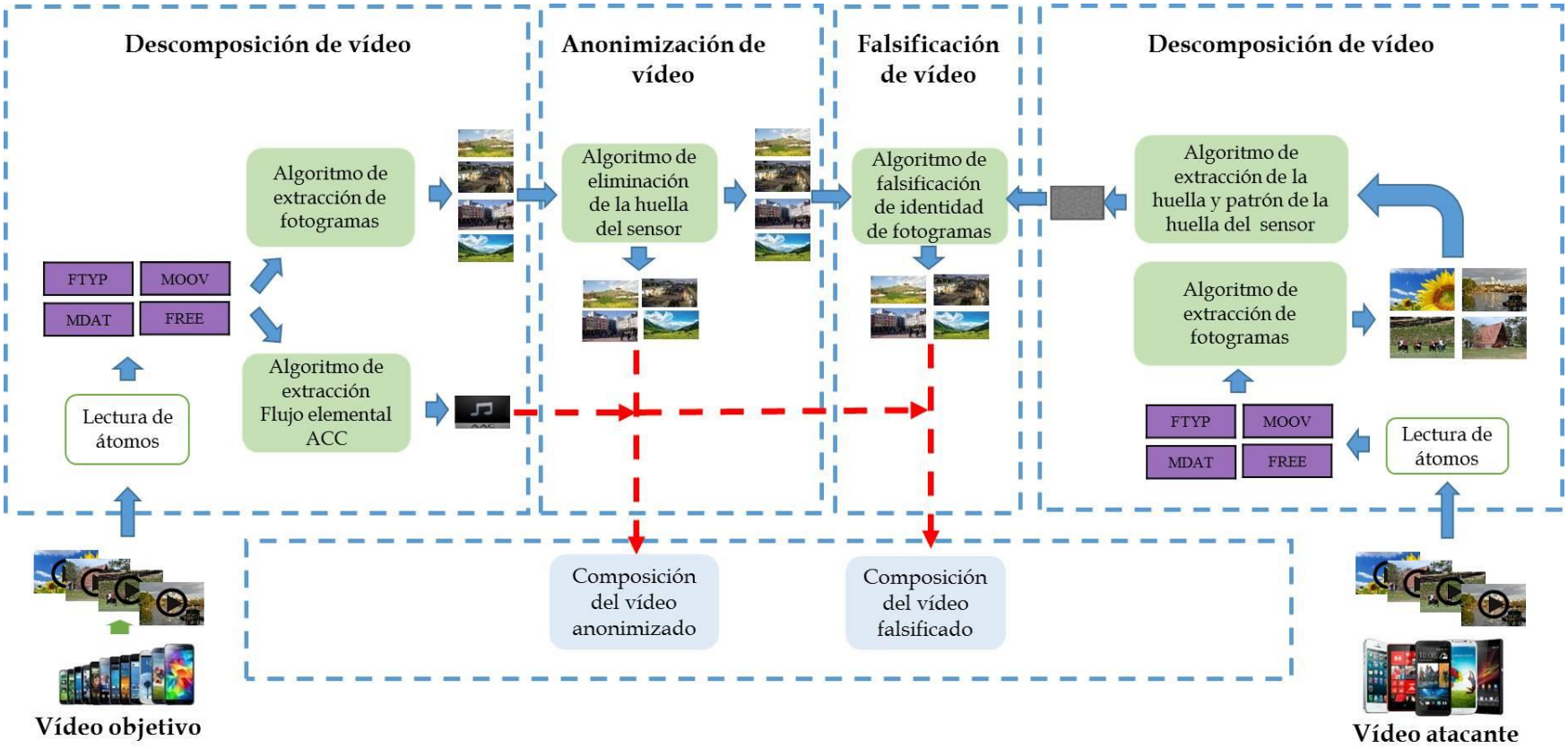


Figura 5.1: Proceso general de anonimización y falsificación de la fuente en vídeos MP4

Algoritmo 3: Anonimización de un vídeo con formato MP4

Input: $Video_{original}$: Es el vídeo objetivo MP4
Result: $Video_{anonimizado}$: Es el vídeo anonimizado MP4

1. **Procedure** ANONIMIZARVIDEO($Video_{original}$)
2. Leer $Video_{original}$;
3. $Audio$ ← EXTRAERFLUJOAAC($Video_{original}$);
4. $Fotogramas_{objetivo}$ ← EXTRAERFOTOGRAMAS($Video_{original}$);
5. **Foreach** *fotograma* in $Fotogramas_{objetivo}$ **do**
6. $Fotogramas_{sinruido}$ ← ELIMINARPRNU(*fotograma*);
7. $Vídeoanonimizado = Audio + Fotogramas_{sinruido}$;
8. **end procedure**

A continuación se describen detalladamente los pasos del mismo:

El algoritmo se inicia con la lectura del vídeo para verificar si la estructura cumple con las especificaciones detalladas en la Sección 3.3. En caso afirmativo, se obtienen los datos necesarios de los átomos contenedores *mdat*, *moov*, *trak* y de sus respectivos átomos hijos. Seguidamente con el Algoritmo 4 se extrae el flujo elemental AAC (audio) desde el vídeo original $Video_{original}$ conservando los datos en bruto para evitar re-comprimir. Después, se realiza la extracción de los fotogramas objetivos $Fotogramas_{objetivo}$ del vídeo original $Video_{original}$ usando el Algoritmo 5.

A continuación, se elimina la huella del ruido PRNU presente en cada fotograma extraído del vídeo $Fotogramas_{objetivo}$. Como resultado se obtiene un conjunto de fotogramas sin ruido $Fotogramas_{sinruido}$. Este proceso se realiza en el Algoritmo 6.

Finalmente, se realiza la recomposición del vídeo con los fotogramas sin ruido, dando como resultado un Vídeo anonimizado $Video_{anonimizado}$. Este proceso necesita dos elementos fundamentales: Por un lado el audio extraído $Audio$, y por otro el conjunto de fotogramas sin ruido $Fotogramas_{sinruido}$.

Para el proceso de recomposición del vídeo anonimizado se utiliza la librería *Libx264* de la API de software libre *FFmpeg* que ayuda sustancialmente al tratamiento de ajuste, conversión y creación de audio y vídeo *Ffmpeg*. Para que el vídeo anonimizado esté lo más sincronizado posible hay que tener en cuenta los siguientes aspectos: el número de fotogramas por segundo o *frame rate*, el tipo de códec, la tasa de bits de vídeo y audio, el perfil del vídeo, el tipo de formato de píxel YUV420P, información de metadatos como la fecha de creación del vídeo y por último la más importante, la sincronización entre el audio y vídeo. Esto se debe a que cada vídeo anonimizado tiene sus propias particularidades en cuanto a la configuración de estos parámetros.

Para realizar la sincronización entre un audio y vídeo, se utiliza *itsoffset* de *FFmpeg*, que permite mover hacia atrás o hacia delante la hora de inicio tanto de la secuencia de datos de audio o de la secuencia de datos de vídeo. Una cámara típica de un dispositivo móvil registra una secuencia de datos de audio y una de vídeo que se fusionan en un solo archivo. Por lo general *itsoffset* se utiliza conjuntamente con *map*, que indica a *FFmpeg* la secuencia de datos se quiere afectar y la secuencia de datos se desea combinar en un nuevo archivo de salida.

A continuación se presentan los algoritmos que componen la anonimización de vídeos MP4.

5.2.1. Algoritmos de Descomposición de un Vídeo con Formato MP4

La descomposición de un vídeo con formato MP4 consiste en separar la pista de vídeo (para extraer los fotogramas) y la pista de audio, a partir del flujo elemental H.264 y del flujo elemental AAC respectivamente.

En la descomposición se tienen en cuenta las principales ideas de [SZC+03], que determinan que un vídeo con formato MP4 está compuesto principalmente

por: los metadatos y la comunicación entre los datos. Éstos, se encuentran en los átomos *moov* y *mdat* respectivamente. Los metadatos deben interpretarse como se muestra en la Figura 5.2. Posteriormente, se extraen los datos del vídeo como se presenta en la Figura 5.3.

A continuación se detalla el procedimiento general para extraer una muestra de vídeo:

- Identificar las pista de vídeo y audio mediante la comprobación de los átomos *moov* y *mdat*.
- Obtener la tabla de muestras de los átomos *stbl* contenidos en las pistas de vídeo y audio. Incluir la ubicación de los datos de los medios y su respectiva descripción.
- Obtener información de la tabla de muestras de los átomos *stco*, las muestras por *chunk* del átomo *stsc* y tamaño de la muestra *stsz*.

A partir del procedimiento escrito anteriormente, se obtiene una tabla que ayuda entender el proceso de identificación de las muestras y a realizar el seguimiento de la información y ubicación de cada muestra.

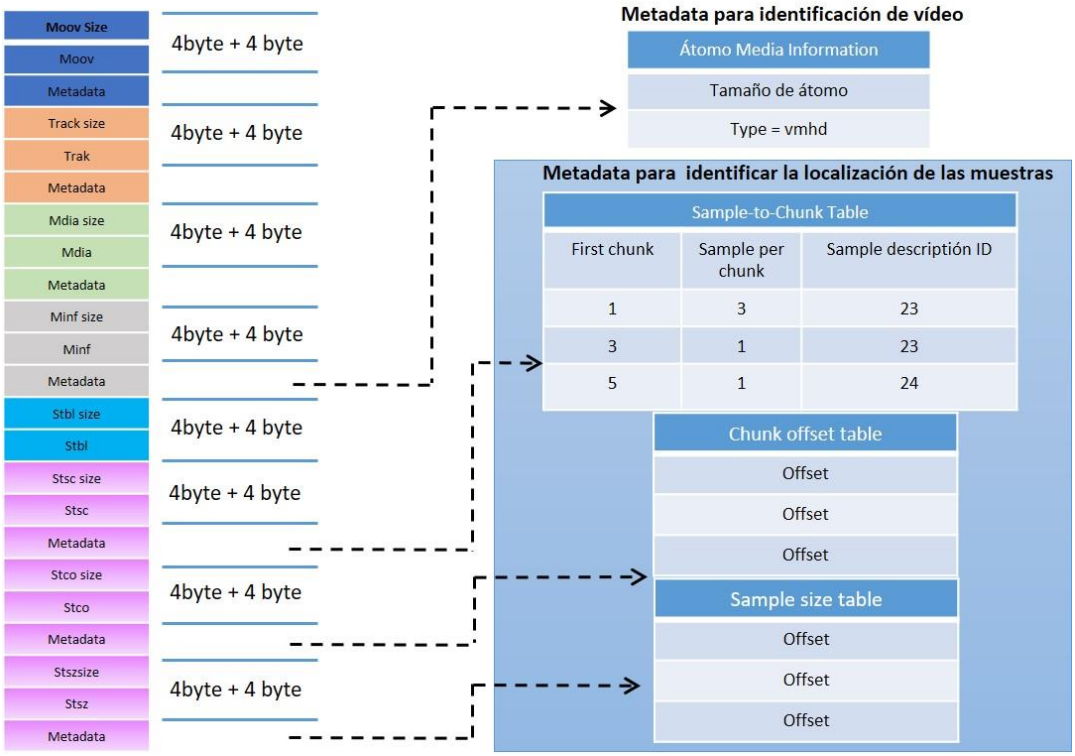
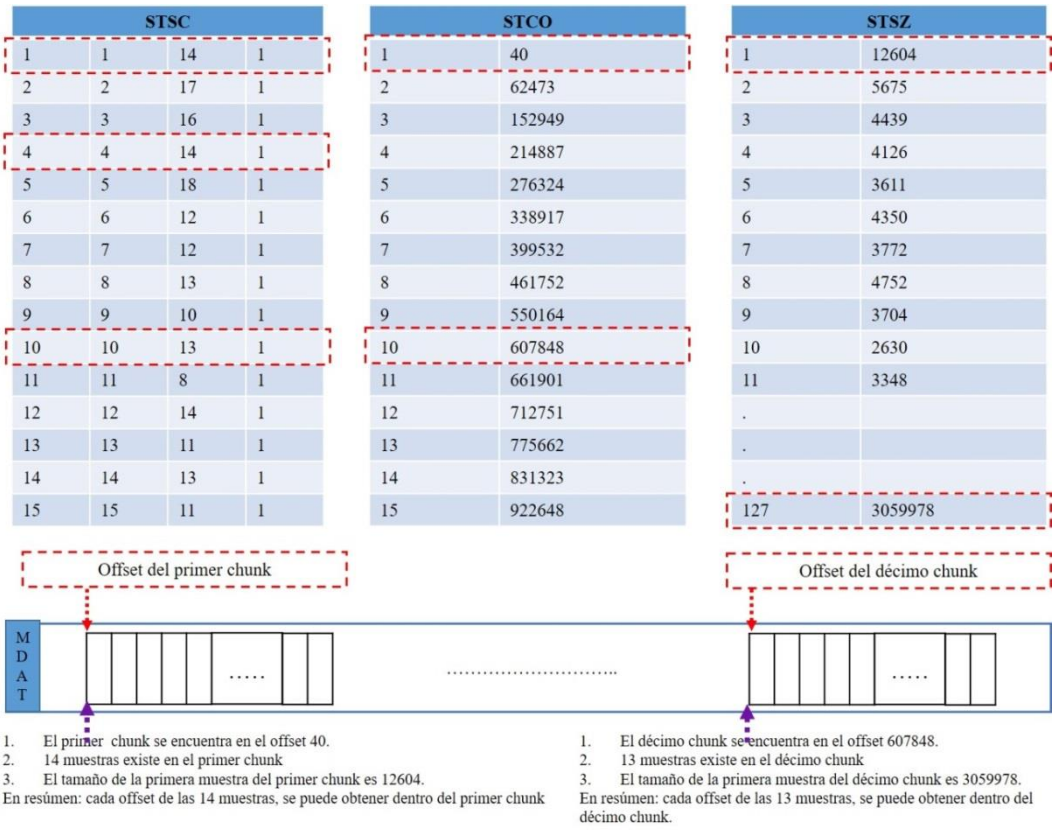


Figura 5.2: Interpretación de los metadatos de un vídeo con formato MP4



STSC			
1	1	14	1
2	2	17	1
3	3	16	1
4	4	14	1
5	5	18	1
6	6	12	1
7	7	12	1
8	8	13	1
9	9	10	1
10	10	13	1
11	11	8	1
12	12	14	1
13	13	11	1
14	14	13	1
15	15	11	1

STCO	
1	40
2	62473
3	152949
4	214887
5	276324
6	338917
7	399532
8	461752
9	550164
10	607848
11	661901
12	712751
13	775662
14	831323
15	922648

STSZ	
1	12604
2	5675
3	4439
4	4126
5	3611
6	4350
7	3772
8	4752
9	3704
10	2630
11	3348
.	
.	
.	
127	3059978

1. El primer chunk se encuentra en el offset 40.
2. 14 muestras existe en el primer chunk
3. El tamaño de la primera muestra del primer chunk es 12604.

En resúmen: cada offset de las 14 muestras, se puede obtener dentro del primer chunk

1. El décimo chunk se encuentra en el offset 607848.
2. 13 muestras existe en el décimo chunk
3. El tamaño de la primera muestra del décimo chunk es 3059978.

En resúmen: cada offset de las 13 muestras, se puede obtener dentro del décimo chunk.

Figura 5.3: Procedimiento general para obtener datos del átomo *mdat*

5.2.1.1. Algoritmo de Extracción Flujo Elemental AAC

En el primer paso, se realiza una lectura recursiva de todos los átomos existentes en el vídeo para después obtener los datos y la información que poseen los átomos *mdat* y *trak* de la pista de audio. Los átomos necesarios para la extracción de la pista de audio son: *trak*, *stbl*, *stsd*, *stsc*, *stco* y *stsz*. Los datos e información necesaria de cada átomo son los siguientes:

- *trak*: Permite diferenciar los elementos que pertenecen a la pista de audio de los que pertenecen a la pista de vídeo. Esto es necesario debido a que ambas pistas tienen el mismo nombre.

- *stbl*: Provee los átomos que contienen los datos a explotar.

- *stsd*: Contiene el número de canales que posee la frecuencia. La mayoría de dispositivos móviles utilizan 2 canales, y la frecuencia (*sample rate*) puede estar en el rango 8 - 96 kHz; pero generalmente se considera 48000 Hz.

- *stsc*: Contiene el número de muestras en un *chunk*.

- *stco*: Contiene la ubicación del *chunk*. Este desplazamiento, también llamado *offset*, se conoce desde el principio del archivo.

- *stsz*: Este átomo provee el tamaño de cada muestra de un *chunk*.

Una vez identificados estos elementos, se crea un fichero con extensión .aac que almacenará el audio. Posteriormente, se obtiene el número de entradas del átomo *stsc*, del diccionario *entries*, que es básicamente una tabla que almacena el número de muestras en cada *chunk*.

El siguiente ejemplo muestra las entradas del átomo *stsc* y el número de muestras en cada *chunk* (ver Figura 5.2).

[(1, 49, 1), (2, 47, 1), (26, 13, 1)]

Donde,

(1, 2, 26) Es el número de *chunk*

(49, 47, 13) Es el número de muestras por *chunk*

(1, 1, 1) Es la descripción del identificador de la muestra.

En la Figura 5.4 se esquematiza gráficamente los requerimientos para la extracción del audio.

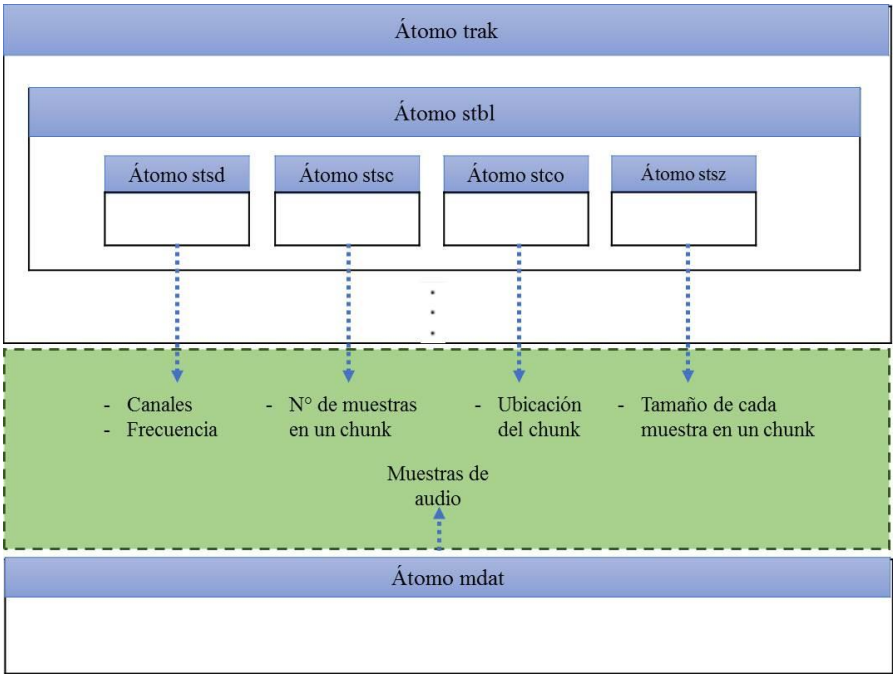


Figura 5.4: Requerimientos para la extracción del audio

A continuación, se obtienen tres datos importantes: el número de *chunk*, el número de muestras que contiene cada *chunk* y el número de iteraciones donde se determina si existen o no más *chunks* en el diccionario para ser leídos. Este procedimiento se realiza calculando la diferencia entre el *chunk* y el número de *chunk*. En caso que fuera la última, se itera una sola vez. Este dato es fundamental a fin de tener identificado el total de *chunks* a ser procesados.

El siguiente paso es obtener el desplazamiento para cada muestra desde las

iteraciones del *chunk* calculado. Con esa información se obtiene el tamaño de cada muestra del *chunk* para escribir la cabecera de cada muestra con los siguientes elementos: Tamaño de la muestra, frecuencia del audio y el canal. La ubicación de cada muestra se desplaza según la información del átomo *stsc* para obtener el bloque de datos del átomo *mdat* del vídeo con formato MP4 y se guardan los datos en un fichero con extensión .aac. El pseudocódigo de este proceso se presenta en el Algoritmo 4.

Algoritmo 4: Extracción del Flujo elemental AAC (Audio)

Input: $Video_{original}$: Es el vídeo objetivo MP4
Result: $Flujo_{elementalaac}$: Es el audio elemental con codificación MP4

```
1.  Procedure EXTRAERFLUJOACC(Vídeo_original)
2.      Leer Atomo_stbl del track de la pista de audio de Vídeo_original;
3.      Obtener datos de Atomo_stsd; Atomo_stsc, Atomo_stco; Atomo_stsz de Atomo stbl;
4.      Obtener Frecuencia_audio de Atomo_stsd;
5.      Obtener Canal_audio de Atomo_stsd;
6.      Establecer extensión de Flujo_elementalaac;
7.      Calcular Numero_entradas de Atomo_stsd;
8.      foreach chunk in Numero_entradas do
9.          Obtener Numerochunk (chunk);
10.         Obtener NumeromuestrasporChunk(chunk);
11.         if Chunk > Numeroentradas then
12.             Iteracioneschunk = (Chunk + 1) - (Numerochunk);
            else
13.             Iteracioneschunk = 1;
14.         foreach chunkleido in Iteracioneschunk do
15.             Obtener Desplazamientochunk (chunkleido);
16.         foreach muestra in NumeromuestrasporChunk do
17.             Obtener Tamañomuestra (muestra);
18.             Escribir Cabeceramuestra = Tamañomuestra+Frecaudio+Canalaudio;
19.             Obtener bloque de datos (muestra);
20.             Volcar datos (muestra);
21. end procedure
```

5.2.1.2. Algoritmo de Extracción de Fotogramas

Uno de los pasos más importantes para anonimizar un vídeo es extraer los fotogramas del mismo. Este proceso se inicia con una lectura recursiva de todos los átomos existentes en un vídeo, para obtener los datos e información que poseen los átomos *mdat* y *trak* de la pista de vídeo. Los átomos necesarios de la pista de vídeo son: *trak*, *stbl*, *stsd*, *stsc*, *stco* y *stsz*. Los datos necesarios de cada átomo son los siguientes:

- *trak*: Este átomo tiene el mismo nombre que el átomo que contiene el audio. Sin embargo, los átomos contienen información relacionada al vídeo almacenado.

- *stbl*: Provee los átomos que contienen los datos a explotar.

- *stsd*: Contiene los datos de registro de configuración del decodificador *(decoderConfigurationRecord)* que forma parte del átomo avcC.

- *stsc*: Contiene el número de muestras en un *chunk*.

- *stco*: Contiene la ubicación del *chunk*. Este desplazamiento se conoce desde el principio del archivo.

- *stsz*: Contiene el tamaño de cada muestra de un *chunk*.

Una vez identificados estos elementos, se crea el fichero con extensión .H264 que almacenará la pista de vídeo. Se obtiene el número de entradas del átomo *stsc* y del diccionario *entries* donde se almacena el número de muestras en cada *chunk*.

El siguiente ejemplo muestra las entradas del átomo *stsc* y el número de muestras en cada *chunk* (ver Figura 5.2).

[(1, 32, 1), (2, 30, 1), (3, 4, 1)]

Donde,

(1, 2, 3) Es el número de *chunk*

(32, 30, 4) Es el número de muestras por *chunk*

(1, 1, 1) Es la descripción del identificador de la muestra.

Los siguientes pasos son similares a los realizados en el algoritmo de extracción de audio hasta calcular el tamaño de cada muestra del *chunk*. De cada una de estas muestras, se obtiene el bloque de datos y se verifica el campo de longitud *nalu size*, que comúnmente está constituido por 4 bytes. La primera parte de la cabecera de la unidad de acceso *(access unit)* se agrega al fichero H.264. En esta cabecera se insertan los siguientes datos:

- Un prefijo de inicio de la unidad de acceso *(start of access unit)* representado por (00 00 00 01).
- La cabecera de la unidad NAL que contiene los campos *forbidden_zero_bit*, *nal_ref_idc* (0) y *nal_unit_type* (9) identificado como *access unit delimiter*.
- El tipo de rebanada *(slice types):* (E0) que representa cualquiera de los siguientes tipos de *frames (I, P, B, SI y SP).*
- Un delimitador (00 00 00 01). Esta cabecera se añade al principio de cada unidad de acceso.

A partir de la posición 7 del *decoderConfigurationRecord* se extrae el tamaño en bytes de la secuencia con los datos que se tienen que leer del decodificador y se agregan estos datos seguidos por el delimitador (00 00 00 01) al fichero .H264. De igual forma, para la segunda parte de la cabecera, es necesario obtener el tamaño en bytes de la siguiente secuencia de datos (a partir de la posición 26) del decodificador y se agregan estos datos seguidos por delimitador (00 00 01). Finalmente, se realiza la escritura de los datos que representan la información

del vídeo. En la Figura 5.5 se muestra la primera parte de la cabecera de cada unidad (NAL).

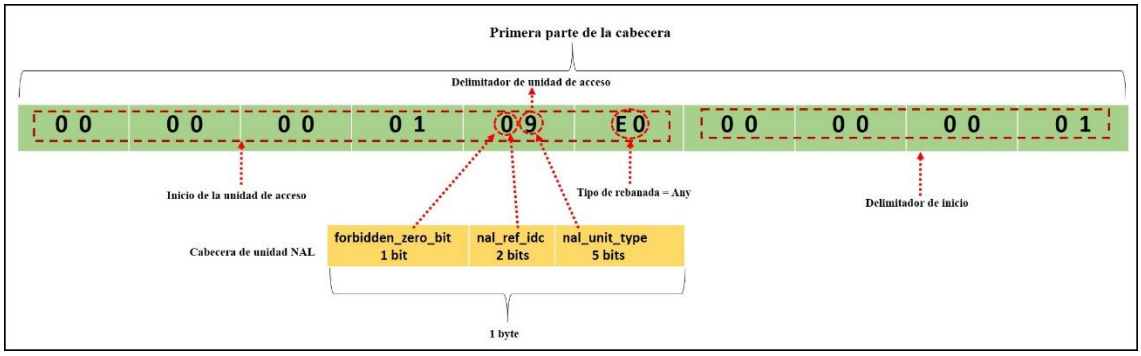


Figura 5.5: Primera parte de la cabecera de cada unidad NAL

Una vez extraído el flujo elemental H.264 del contenedor MP4, se realiza la extracción de todos los fotogramas del fichero .H264. En primer lugar se extrae el conjunto de matrices de valores [R, G, B] de cada fotograma, es decir de cada unidad de acceso del flujo elemental H.264, para posteriormente ser convertidos en imágenes con formato JPEG. En el Algoritmo 5 muestra el pseudocódigo de la extracción de fotogramas.

Algoritmo 5: Extracción de Fotogramas

Input: $Video_{mp4}$: Es el vídeo con formato MP4 sea objetivo o atacante
Result: $Conjunto_{fotogramas}$: Es número de fotogramas extraídos en un vídeo con formato MP4

1. **Procedure** EXTRAERFOTOGRAMAS($Video_{mp4}$)
2. Leer $Atomo_{stbl}$ del track de la pista de vídeo de $Vídeo_{mp4}$;
3. Obtener datos de $Atomo_{stsd}$; $Atomo_{stsc}$, $Atomo_{stco}$; $Atomo_{stsz}$ de *Atomo stbl*;
4. Establecer extensión de $Flujo_{elementalh264}$;
5. Calcular $Numero_{entradas}$ de $Atomo_{stsd}$;
6. **Foreach** *chunk* in $Numero_{entradas}$ **do**
7. Obtener *Numerochunk* (*chunk*);
8. Obtener *NumeromuestrasporChunk*(*chunk*);
9. **if** *Chunk* > *Numeroentradas* then
10. *Iteracioneschunk* = (*Chunk* + 1) - (*Numerochunk*);
    else
11. *Iteracioneschunk* = 1;
12. **foreach** *chunkleido* in *Iteracioneschunk* **do**
13. Obtener *Desplazamientochunk* (*chunkleido*);
14. **foreach** *muestra* in *NumeromuestrasporChunk* **do**
15. Obtener *Tamañomuestra* (*muestra*);
16. Obtener bloque de datos de (*muestra*);
17. Verificar longitud Nalu Size de (*muestra*);
18. Agregar en fichero la primera parte de cabecera de (*muestra*);
19. Agregar en fichero segunda parte de cabecera de (*muestra*);
20. Agregar en fichero delimitador de (*muestra*);
21. Volcar datos (*muestra*);
22. Extraer fotogramas desde $Flujo_{elementalh264}$;
23. **end procedure**

5.2.2. Algoritmo de Eliminación de la Huella del Sensor

El algoritmo de eliminación de la huella del sensor en fotogramas propuesto en este trabajo se basa en el presentado [OVG+15]. Sobre este algoritmo se realizaron algunas modificaciones, con el objeto de mejorar la huella PRNU que

se ve afectada por los parámetros de codificación del vídeo. En la Figura 5.6 se presenta el diagrama funcional de eliminación de la huella del sensor de un fotograma.

En el algoritmo propuesto es necesario utilizar la resolución original del vídeo. Asimismo, se debe calcular la varianza local de forma no adaptativa ya que al procesar todos los fotogramas del vídeo en su resolución original, los requerimientos computacionales son mucho mayores. En la literatura se han obtenido buenos resultados calculando la varianza local para un solo tamaño de ventana, considerando W=3. Posteriormente, se calculan los componentes wavelet sin ruido utilizando el filtro de Wiener.

$$c_{limpio}(i,j) = c(i,j)\frac{\hat{\sigma}^2(i,j)}{\hat{\sigma}^2(i,j)+\hat{\sigma}_0^2} \tag{5.1}$$

Por último, el fotograma final sin ruido se obtiene aplicado la transformada wavelet inversa a los componentes wavelet sin ruido de cada uno de los niveles. El Algoritmo 6 presenta el pseudocódigo.

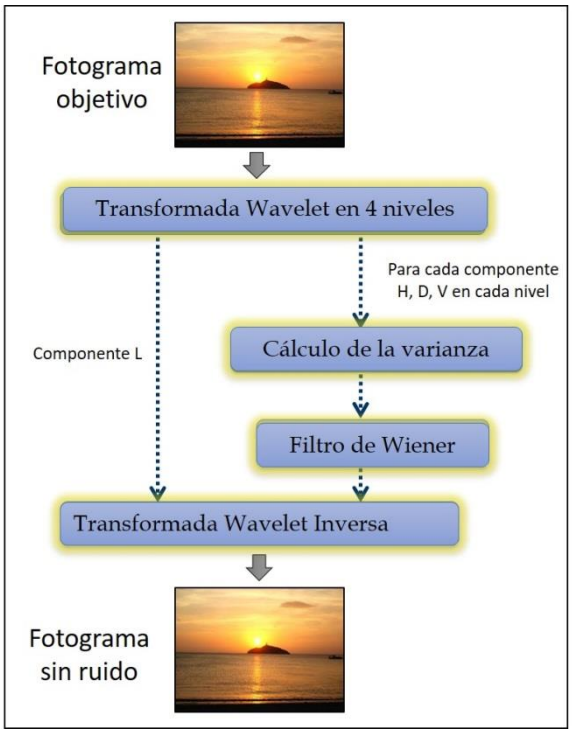


Figura 5.6: Diagrama funcional de eliminación de la huella del sensor de un fotograma

Algoritmo 6: Eliminación de la huella del sensor

Input: *I*: Es el fotograma objetivo con extensión JPEG
Result: $Fotograma_{sinruido}$: Es el fotograma sin ruido con extensión JPEG

1. **Procedure** ELIMINARPRNU(*I*)
2. Realizar una descomposición wavelet de 4 niveles de I_n;
3. **foreach** *nivel de la descomposición wavelet* **do**
4. **foreach** $c \in \{H, V, D\}$ **do**
5. Calcular la varianza local con una ventana tamaño 3;
6. Calcular los componentes wavelet sin ruido aplicando el filtro Wiener a la varianza;
7. Obtener el fotograma limpio del ruido del sensor aplicando la transformada wavelet Inversa;
8. **end procedure**

5.3. Falsificación de un Vídeo con Formato MP4

Es fundamental precisar que para efectuar este procedimiento intervienen dos actores: un vídeo atacante y un vídeo objetivo como se ilustra en la Figura 5.1. El pseudocódigo general de falsificación de la fuente de un vídeo con formato MP4 se muestra en el Algoritmo 7.

Algoritmo 7: Falsificación de la fuente de un vídeo con formato MP4

Input: $Video_{original}$: Es el vídeo objetivo con formato MP4
$Video_{atacante}$: Es el vídeo atacante con formato MP4
Result: $Video_{falsificado}$: Es el vídeo falsificado con formato MP4

1. **Procedure** FALSIFICARMP4($Video_{original}$, $Video_{atacante}$)
2. Leer $Video_{original}$;
3. $Fotogramas_{objetivo}$ ← EXTRAERFOTOGRAMAS($Video_{original}$);
4. *Audio* ← EXTRAERFLUJOAAC($Video_{original}$);
5. **Foreach** *fotogramaO* in $Fotogramas_{objetivo}$ **do**
6. $Fotogramas_{sinruido}$ ← ELIMINARPRNU(*fotogramaO*);
7. Leer $Video_{atacante}$;
8. $Fotogramas_{atacantes}$ ← EXTRAERFOTOGRAMAS($Video_{atacante}$);
9. **Foreach** *fotogramaA* in $Fotogramas_{atacantes}$ **do**
10. $Huellas_{atacante}$ ← EXTRAERHUELLA(*fotogramaA*);
11. Calcular el patrón de ruido $patron_{atacante}$ = promedio($Huellas_{atacante}$);
12. **Foreach** *FotogramaH* in $Fotogramas_{sinruido}$ **do**
13. $Fotogramas_{falsos}$ ← FALSIFICACIONFOTOGRAMA*(FotogramaH);*
14. $Video_{falsificado}$ = *Audio*+$Fotogramas_{falsos}$;
15. **end procedure**

Primero, el algoritmo realiza una lectura del vídeo objetivo para verificar si la estructura cumple con las especificaciones detalladas en la sección 3.3. Si coincide, se extraen los datos e información de los átomos *mdat, moov*, *trak* y de sus respectivos átomos hijos, de las pistas de audio y vídeo.

Posteriormente, se realiza la extracción del flujo elemental AAC con el Algoritmo 4 para obtener el audio original del vídeo y la extracción de los fotogramas con el Algoritmo 5. Cada fotograma del vídeo objetivo contiene la huella intrínseca que deja el sensor del dispositivo móvil. Por tanto, el siguiente paso es eliminar la huella del sensor en cada uno de los fotogramas del vídeo original con el Algoritmo 6.

Análogamente al vídeo objetivo, se verifica si la estructura del vídeo atacante cumple con las especificaciones detalladas en la sección 3.2. Se siguen los

mismos pasos hasta extraer tanto los fotogramas como el audio del vídeo.

A continuación, se realiza la extracción del patrón del ruido del sensor del vídeo atacante, utilizando los fotogramas extraídos del vídeo y el Algoritmo 6 de eliminación de la huella del sensor con tres pasos adicionales.

1. Se calcula la huella o ruido residual $I_{ruido} = I - F(I)$ de un fotograma I eliminando el contenido de la escena del fotograma mediante un filtro de eliminación de ruido F.

   Entre los diferentes filtros que existen para la eliminación del ruido de las imágenes fijas los que usan la transformada wavelet dan mejor resultado, debido a que el ruido residual que se obtiene con este filtro contiene la menor cantidad de rasgos de la escena. Generalmente, las áreas que están alrededor de los bordes son mal interpretadas cuando se utilizan únicamente filtros de eliminación de ruido menos robustos, tales como el filtro de Wiener o el filtro de mediana. Por este motivo se selecciona el filtro de eliminación de ruido basado en la transformada wavelet.

   La huella calculada I_{ruido} contiene todos los elementos que se presentan sistemáticamente en cada uno de los fotogramas, incluyendo algunos que no son causados por el sensor como las características afectadas por la interpolación de colores o la compresión JPEG. La mayoría de estas características son generadas por el proceso de interpolación cromática dependiendo de la CFA utilizada por la cámara. Debido a que estas características tienen una naturaleza periódica se pueden eliminar mediante el tercer paso.

2. Se limpia la huella de las características que no son intrínsecas al sensor aplicando un promediado a cero (también denominado *zero mean*) de filas y columnas como se sugiere en [CFGL08], de tal manera que los promedios de las filas y de las columnas sean iguales a cero. Esto se logra

restando el promedio de la columna a cada píxel de la columna y posteriormente restando el promedio de la fila a cada píxel de la fila. Esta operación se aplica a todas las filas y columnas de la imagen.

3. Por último, se da mayor peso al canal verde ya que éste, debido a la configuración de la matriz de color, contiene más información sobre la imagen que el resto de los canales de color [APS98] [McK07] [CSA08]: $I_{ruido} = 0{,}3 \, . \, I_{ruidoR} + 0{,}6 \, . \, I_{ruidoG} + 0{,}1 + I_{ruidoB}$.

   Seguidamente se calcula patrón del ruido *Pixel Non-Uniformity* (PNU) mediante el promedio del ruido residual del conjunto de fotogramas del vídeo atacante. Este patrón del ruido, es un elemento esencial que será incrustado a todos los fotogramas extraídos del vídeo objetivo, como se observa en los apartados posteriores.

   Después se realiza la falsificación de cada fotograma sin ruido del vídeo objetivo, incrustando el patrón del atacante a cada fotograma sin ruido. Producto de esto se obtiene un conjunto de fotogramas falsos $Fotogramas_{falsos}$. Este procedimiento se realiza mediante el Algoritmo 6 de falsificación de fotogramas presentado posteriormente.

Finalmente, se realiza la recomposición del $Video_{falsificado}$; este proceso necesita dos elementos fundamentales: Por un lado el audio extraído *Audio* del vídeo original, y por otro el conjunto de fotogramas falsos $Fotogramas_{falsos}$ obtenidos en el paso anterior.

A continuación en la Figura 5.7 se muestra gráficamente los pasos para falsificar un vídeo con formato MP4.

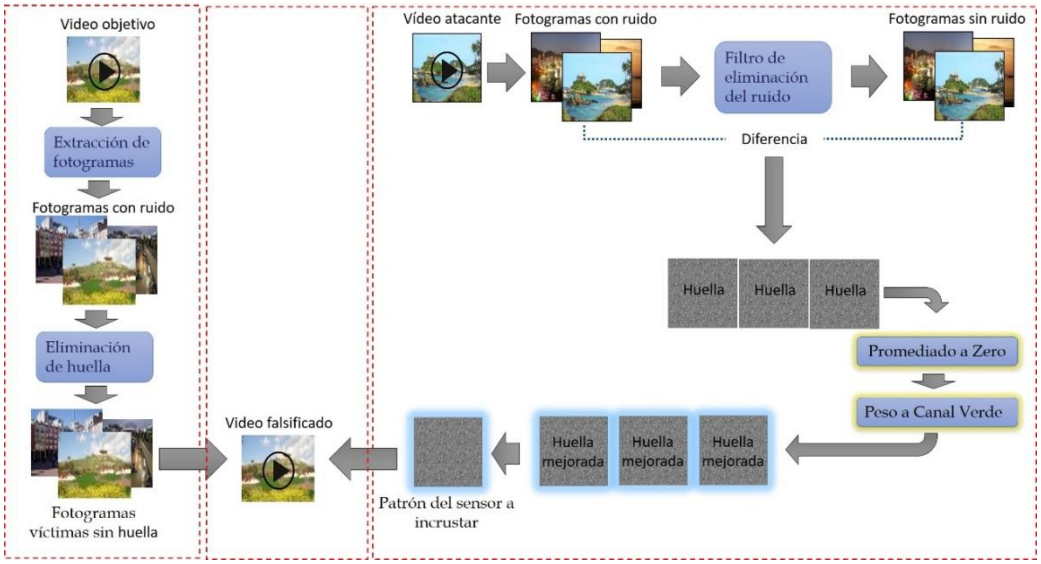


Figura 5.7: Pasos para falsificar un vídeo con formato MP4

5.3.1. Algoritmo de Falsificación de Identidad de un Fotograma

Una vez eliminado el ruido del sensor del conjunto de fotogramas del vídeo objetivo y de extraer el patrón del ruido del sensor del conjunto de fotogramas del vídeo atacante se procede a realizar la falsificación de la identidad de cada uno de los fotogramas del vídeo objetivo. El Algoritmo 6 muestra los pasos a seguir para falsificar la identidad de cada fotograma.

Algoritmo 6: Falsificación de identidad de un fotograma

Input: $Fotograma_{sinruido}$: Fotograma sin ruido de vídeo objetivo con formato MP4
$Patron_{atacante}$: Patrón del ruido del vídeo atacante con formato MP4
Result: $Fotograma_{falso}$: Fotograma con falsa identidad

1. **Procedure** FALSIFICARFOTOGRAMA($Fotograma_{sinruido}$)
2. Realizar una descomposición wavelet de un nivel de $Fotograma_{sinruido}$ obteniendo los componentes L_I, H_I, V_I, D_I;
3. Realizar una descomposición wavelet de un nivel de $Patron_{atacante}$ obteniendo los componentes H_P, V_P, D_P;
4. Calcular los componentes wavelet falsificados mediante $C_F=C_I + C_P$ Donde $c \in \{H, V, D\}$;
5. Obtener $Fotograma_{falso}$ aplicando la transformada wavelet inversa con L_I, H_F, V_F, D_F;
6. **end procedure**

6. EXPERIMENTOS Y RESULTADOS

En este capítulo se describen los experimentos realizados para evaluar la estructura e información de los átomos de vídeos MP4 y la eficacia de las técnicas anti-forenses para la anonimizar y falsificar los vídeos multimedia con formato MP4.

6.1. Análisis de las Estructuras de los Átomos de Vídeos MP4

El presente análisis tiene como objetivo consolidar los conocimientos sobre la especificación y comprobar si ésta es utilizada por los fabricantes. Los vídeos analizados fueron obtenidos por dispositivos móviles de personas conocidas lo que asegura que los vídeos no han sido objeto de post-procesamiento alguno. El conjunto de vídeos está conformado por de 60 vídeos grabados de 6 modelos de 5 fabricantes diferentes. En la Tabla 6.1 se muestra detalladamente teléfonos móviles utilizados clasificados por marca y modelo.

Tabla 6.1: Teléfonos móviles clasificados por marca y modelo

Marca	Modelo
Huawei	Y635-L01
Motorola	Nexus 6
Samsung	Galaxy S5
	Galaxy S6
Sony	Xperia M2
Xiomi	Mi3

El primer átomo encontrado en todos los vídeos es el *fytp* cómo indica la especificación. Cuando el vídeo contiene el átomo *free*, el siguiente átomo que se encuentra es el *moov* seguido de sus átomos hijos, terminando con los átomos *free* y *mdat*. Cuando el vídeo no tiene el átomo *free*, el siguiente átomo que se observa es el *mdat*, seguido del átomo *moov* con sus respectivos átomos hijos.

Al respecto, la especificación no contempla un orden determinado de los átomos, como tampoco la aparición de un átomo en concreto; esto dependerá de cada fabricante o modelo, aunque después de realizar el análisis sobre varios vídeos se aprecia que generalmente siguen el mismo patrón en cuanto a orden y tipos de átomos contenidos.

Se observa que el átomo *moov* posee la misma estructura, pero varían en algunos casos los tipos de átomos que contienen, como el caso de los móviles Motorola Nexus 6 tiene el átomo hijo *meta* que a su vez contiene átomos hijos *hdlr*, *keys* e *ilst*; como los móviles Samsung S5, Samsung S6, y Sony Xperia M2 que tienen el átomo *udta*. El primer átomo *trak* encontrado es de la pista de vídeo y el segundo es de la pista de audio. Esto se determina porque el átomo *hdlr* del primer *trak* contiene los campos *component type* y *subtype* con valor *vide* y *videohandle* y el átomo *hdlr* del segundo *track* contiene los campos *component type* y *subtype* con valor *soun* y *soundhandle*.

Tras analizar el primer átomo *trak*, pista de vídeo, se observa que la mayoría de átomos contenidos son iguales a excepción del átomo hijo *minf*, que contiene los átomos *vmhd* y *dinf*, este último siempre tiene el átomo hijo *dref* y éste a su vez el átomo *url*, como se indica en la especificación. Seguidamente se encuentra el átomo *stbl* que contiene los átomos *stsd*, *stts*, *stss*, *stsz*, *stsc* y *stco*. En el átomo *stsd* contiene el átomo *avc1* que indica el tipo de códec, este átomo a su vez contiene un átomo de extensión *avcC* y en algunos casos aparece un átomo denominado *pasp*.

El átomo *trak*, la pista de audio, contiene como átomo hijo a *minf*, y este a su vez los átomos hijos *smhd*, en sustitución del *vmhd* y *dinf*, que es igual al del primer átomo *trak* descrito anteriormente. El átomo *stbl* es igual al del primer átomo *trak*, con las excepciones de no existir un átomo *stss* y que el átomo *stsd*, contiene el átomo *mp4a* que indica el tipo de códec, que a su vez contiene al átomo de extensión *esds*. Todos estos detalles se pueden apreciar en la Figura

6.1, donde se consolida las estructuras de átomos de vídeos por marca y modelo.

Producto del análisis de los 60 vídeos en cuestión, se ha consolidado la información de sus respectivos átomos en la Tabla 6.2. Por tanto, después de terminar con el análisis de las estructuras e información de los átomos de vídeos MP4, se puede concluir, que un mismo teléfono móvil siempre tiene los mismos átomos de vídeo y que estos mantienen el mismo orden; casi todos los fabricantes siguen la misma especificación con algunas excepciones, como el formato preferido o la lista de formatos compatibles, si contienen datos de usuario o metadatos, entre otros.

Figura 6.1: Estructuras de átomos de vídeos por marca y modelo

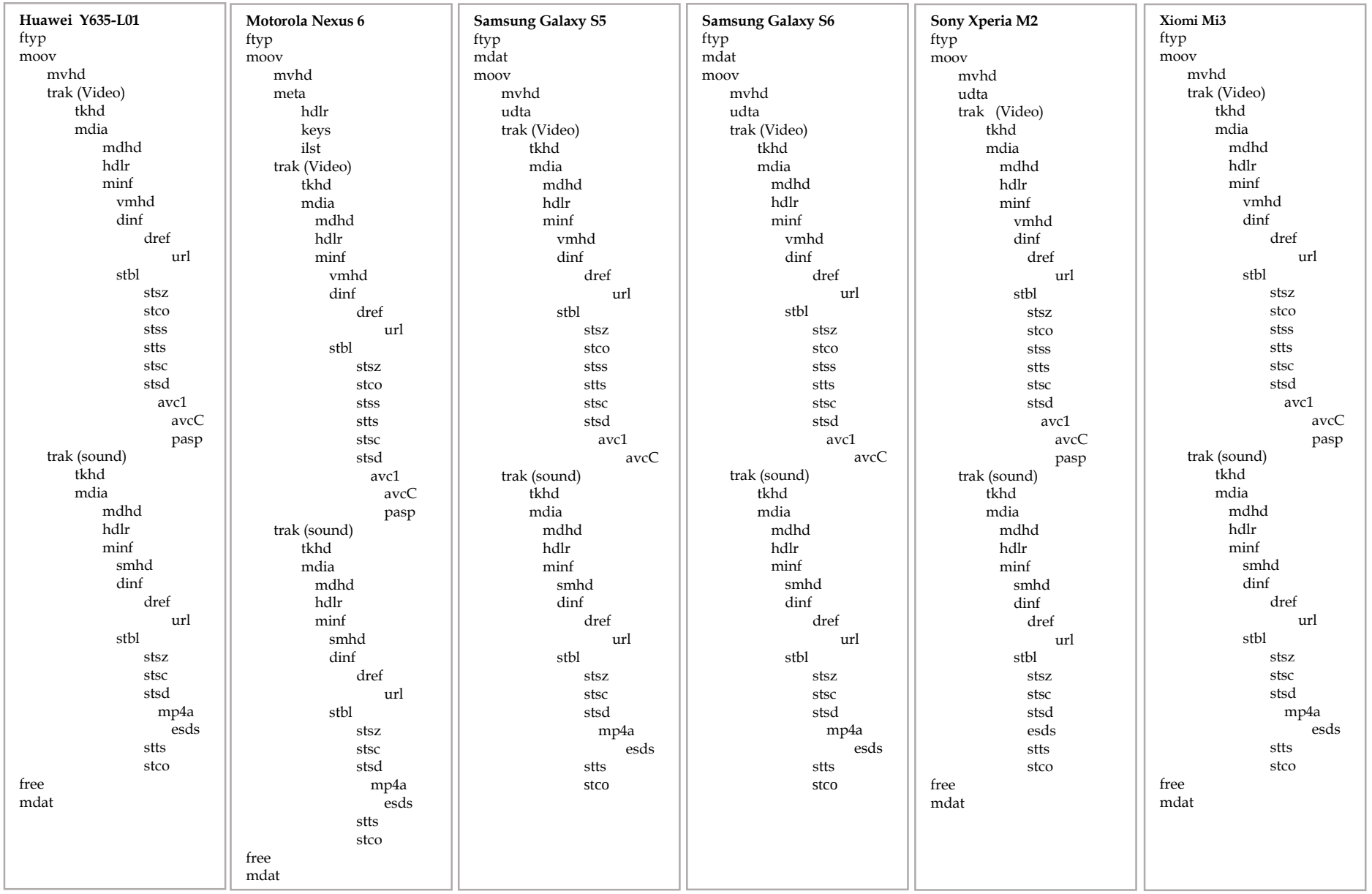


Tabla 6.2: Estructuras de átomos de vídeos por marca y modelo

Marca			**Huawei**	**Motorola**	**Samsung**		**Sony**	**Xiomi**
Modelo			**Y635-L01**	**Nexus 6**	**Galaxy S5**	**Galaxy s6**	**Xperia M2**	**Mi3**
Formato	Preferido		isom	mp42	mp42	mp42	mp42	isom
	Compatibles		[isom,3gp4]	[isom,mp42]	[isom,mp42]	[isom,mp42]	[isom,mp42]	[isom,3gp4]
Película	Escala de tiempo		1000	1000	1000	1000	1000	1000
	Volumen		1.0	1.0	1.0	1.0	1.0	1.0
	Velocidad		1.0	1.0	1.0	1.0	1.0	1.0
	Datos de usuario		-	-	SDLN,smrd,smta	SDLN,smrd,smta	XYZ	-
	Metadatos	Campo	-	com.android.version	-	-	-	-
Pista de vídeo	Escala de tiempo		90000	90000	90000	90000	90000	90000
	Color		[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]
	Modo gráfico		0	0	0	0	0	0
	Manipulador		VideoHandle	VideoHandle	VideoHandle	VideoHandle	VideoHandle	VideoHandle
	S. manipulador		vide	vide	vide	vide	vide	vide
	Ancho		1280	1920	1920	1920	1920	1280
	Alto		720	1080	1080	1080	1080	720
	ID		1	1	1	1	1	1

Pista de vídeo	**Marca**		**Huawei**	**Motorola**	**Samsung**	**Sony**	**Xiomi**	**Marca**
	Modelo		**Y635-L01**	**Nexus 6**	**Galaxy S5**	**Galaxy s6**	**Xperia M2**	**Mi3**
	Volumen		0.0	0.0	0.0	0.0	0.0	0.0
	Tipo		avc1	avc1	avc1	avc1	avc1	avc1
	Resolución	Horizontal	72.0	72.0	72.0	72.0	72.0	72.0
		Vertical	72.0	72.0	72.0	72.0	72.0	72.0
	Códec	Tabla de color	65535	65535	65535	65535	65535	65535
	Ancho		1280	1920	1920	1920	1920	1280
	Alto		720	1080	1080	1080	1080	720
	Extensiones		[avcC,pasp]	[avcC,pasp]	[avcC]	[avcC]	[avcC,pasp]	[avcC,pasp]
	E. pixeles	Horizontal	65536	65536	-	-	65536	65536
		Vertical	65536	65536	-	-	65536	65536
	Escala de tiempo		4800	4800	4800	4800	4800	4800
	Manipulador		SoundHandle	SoundHandle	SoundHandle	SoundHandle	SoundHandle	SoundHandle
	S. manipulador		soun	soun	soun	soun	soun	soun
	Ancho		0	0	0	0	0	0
Pista de audio	Alto		0	0	0	0	0	0
	ID		2	2	2	2	2	2
	Volumen		1.0	1.0	1.0	1.0	1.0	1.0
	Códec	Tipo	mp4a	mp4a	mp4a	mp4a	mp4a	mp4a
	Extensiones		[esds]	[esds]	[esds]	[esds]	[esds]	[esds]

6.2. Identificación de la Fuente de Adquisición con Vídeos sin Post-Procesamiento

Como el objetivo de las técnicas de anti-forenses propuestas en este trabajo es impedir la correcta identificación correcta de la fuente de adquisición de los vídeos procesados. Se utilizó un algoritmo de identificación de la fuente de adquisición de basado en el agrupamiento [VOC15] para realizar dos experimentos de clasificación.

En la Tabla 6.3 se muestra detalladamente los teléfonos móviles clasificados por marca y modelo utilizados en los siguientes 4 experimentos.

Tabla 6.3: Teléfonos Móviles utilizados en los experimentos

Marca	Modelo
Samsung	GalaxyA3
	GalaxyAceStyle
	GalaxyS5Neo
	GalaxyS6
	GalaxyGT-I9000

6.2.1. Experimento 1

En este primer experimento se llevó a cabo una clasificación mediante técnicas de agrupamiento de los fotogramas sin anonimizar para verificar que los fotogramas se agruparan en las categorías a las que se conocía a priori que pertenecen. Para el presente experimento se ha utilizado un conjunto de vídeos pertenecientes a 1 marca y 5 modelos diferentes. De cada uno de los 5 modelos se utilizaron 4 vídeos y de éstos se extrajeron 50 fotogramas con resolución de 640x480 píxeles, resultando un total de 200 fotogramas para cada modelo de la marca analizada. Como se puede apreciar en la Tabla 6.4 al realizar la clasificación se obtuvieron 5 grupos con una tasa de acierto promedio 76.5%.

Tabla 6.4: Matriz de confusión de vídeos no anonimizados (640x480 píxeles)

Teléfono Móvil		Samsung					Tasa de acierto
		Galaxy A3	Galaxy Ace Style	Galaxy S5 Neo	Galaxy S6	GT I9000	
Samsung	Galaxy A3	**159**	18	19	2	2	**80%**
	Galaxy Ace Style	18	**133**	26	10	13	**67%**
	Galaxy S5 Neo	21	8	**170**	1	0	**85%**
	Galaxy S6	2	4	0	**171**	23	**86%**
	GT I9000	4	30	6	28	**132**	**66%**

6.2.2. Experimento 2

Con el propósito de identificar si la resolución afecta los resultados se repitió la misma clasificación que en el experimento 1 pero con fotogramas con una resolución de 1280x720 píxeles. Los resultados se resumen en la Tabla 6.5.

Tabla 6.5: Matriz de confusión de vídeos no anonimizados (1280x720 píxeles)

Teléfono Móvil		Samsung					Tasa de acierto
		Galaxy A3	Galaxy Ace Style	Galaxy S5 Neo	Galaxy S6	GT I9000	
Samsung	Galaxy A3	**192**	8	0	0	0	**96%**
	Galaxy Ace Style	4	**192**	0	0	4	**96%**
	Galaxy S5 Neo	0	0	**200**	0	0	**100%**
	Galaxy S6	0	0	0	**200**	0	**100%**
	GT I9000	6	4	0	4	**186**	**93%**

Como se puede apreciar en la tabla al realizar la clasificación se obtuvieron 5 grupos con una tasa de acierto promedio del 97%. Los resultados de estos dos experimentos servirán más adelante para evaluar la efectividad de la anonimización y falsificación de la fuente de los vídeos.

6.3. Anonimización de Vídeos MP4

Para evaluar la efectividad de la anonimización de vídeos MP4 generados por dispositivos móviles, se anonimizó un vídeo de cada modelo de los móviles detallados en la Tabla 6.3. Una vez anonimizados los vídeos se ejecutan los experimentos 1 y 2 incluyendo los videos anonimizados.

En estos experimentos se utilizaron 3 vídeos sin anonimizar de cada modelo de la Tabla 6.3 y un vídeo anonimizado, para un total de 4 vídeos por modelo. De los 3 vídeos sin anonimizar se extrajeron 50 fotogramas (150 fotogramas por modelo) y del vídeo anonimizado por modelo se extrajeron 50 fotogramas, para un total de 200 fotogramas por modelo. Los resultados de estos experimentos se detallan a continuación.

6.3.1. Experimento 3

En este experimento se realizó la clasificación con una resolución de fotogramas de 640x480 píxeles. En la Tabla 6.6 se muestra como se agruparon los 200 fotogramas de los diferentes modelos. Del modelo Galaxy A3, de 150 fotogramas sin alteraciones en la identidad, se clasificaron 110 correctamente obteniendo un 73,2% de acierto. Del modelo Galaxy Ace Style, de los 150 fotogramas, se clasificaron correctamente 120 con un porcentaje de acierto del 80%. El resto de los modelos se interpretan de la misma forma alcanzando una tasa de acierto promedio del 73.2%. Se observa que después de la clasificación aparecieron 3 nuevas clases: Grupo 1, Grupo 2 y Grupo 3. Un total de 260 fotogramas fueron clasificados en estas últimas 3 clases, por lo que se puede concluir que el algoritmo de anonimización tuvo éxito ya que 250 fotogramas fueron anonimizados. La tasa de acierto promedio de la clasificación con respecto al experimento 1 disminuyó solo un 3,3%. Esto muestra que los vídeos anonimizados no fueron asociados con su modelo real hay que tener en cuentaque la baja resolución también afecta la clasificación.

Tabla 6.6: Matriz de confusión de vídeos anonimizados (640x 480 píxeles)

Teléfono Móvil		Samsung					Grupo 1	Grupo 2	Grupo 3	Tasa de acierto
		Galaxy A3	Galaxy Ace Style	Galaxy S5 Neo	Galaxy S6	GT I9000				
Samsung	Galaxy A3	**110**	8	23	4	0	29	6	20	**73%**
	Galaxy Ace Style	5	**120**	17	2	6	28	12	10	**80%**
	Galaxy S5 Neo	16	9	**118**	0	0	33	10	13	**79%**
	Galaxy S6	3	15	0	**121**	12	8	38	3	**81%**
	GT I9000	10	27	10	23	**80**	10	3	37	**53%**

6.3.2. Experimento 4

En la Tabla 6.7 se muestra el resultado de la clasificación con fotogramas de una resolución de 1280x720. La interpretación de estos resultados se realiza de forma análoga al experimento anterior, obteniendo una tasa de acierto promedio en la clasificación del 95.7%. Como se puede observar en la tabla, dos nuevas clases aparecieron en la clasificación: Grupo 1 y Grupo 2. Del total de los 250 fotogramas anonimizados 229 no se clasificaron como parte del grupo al que pertenecían originalmente alcanzando un 91.6% de efectividad en la anonimización. En esta ocasión, la tasa de acierto promedio de la clasificación con respecto al experimento 1 disminuyó solo un 1,3%. Se aprecia en la Tabla 6.6 que en la clasificación aparecen tres grupos desconocidos, a diferencia de los dos grupos desconocidos que aparencen en este experimento. Esto se debe a que la clasificación es más precisa cuando la resolución es más alta. Entre menos grupos desconocidos detectados, la anonimización es aún más efectiva ya que elimina características que posibiliten su clasificación

Tabla 6.6: Matriz de confusión de vídeos anonimizados (640x 480 píxeles)

Teléfono Móvil		Samsung					Grupo 1	Grupo 2	Tasa de acierto
		Galaxy A3	Galaxy Ace Style	Galaxy S5 Neo	Galaxy S6	GT I9000			
Samsung	**Galaxy A3**	**142**	8	3	1	0	20	26	95%
	Galaxy Ace Style	5	**140**	0	0	4	36	15	93%
	Galaxy S5 Neo	0	3	**150**	0	0	29	18	100%
	Galaxy S6	7	0	1	**144**	3	42	3	96%
	GT I9000	5	6	1	6	**142**	17	23	95%

6.4. Evaluación de Falsificación de la Fuente de Vídeos MP4

Para evaluar la efectividad de la falsificación de la fuente de vídeos MP4 generados por dispositivos móviles se realizó la falsificación de un vídeo (vídeo objetivo) haciendo que éste pareciera haber sido grabado por otro modelo de dispositivo móvil (vídeo atacante). Para este propósito se utilizaron los mismos modelos del experimento 1 descritos en la Tabla 6.3. Los roles que jugaron los modelos involucrados en la falsificación se detallan en la Tabla 6.8. Ningún vídeo de los 4 modelos restantes fue falsificado.

Tabla 6.8: Dispositivos utilizados en el experimento 1

Vídeo Objetivo			Vídeo Atacante		
Marca	**Modelo**	**Resolución**	**Marca**	**Modelo**	**Resolución**
Samsung	Galaxy Ace Style	1280x720	Samsung	Galaxy GT-I9000	1280x720

De forma similar a los experimentos anteriores se utilizaron 4 vídeos de cada modelo de la Tabla 6.3. Cada vídeo se descompuso en 50 fotogramas (200 fotogramas para cada modelo). Del vídeo falsificado como Galaxy GT-I9000 se extrajeron 50 fotogramas y 150 fotogramas de los 3 vídeos restantes del mismo modelo para completar el total de los 200 fotogramas del modelo. Los resultados de la clasificación después de llevar a cabo la falsificación se detallan en la siguiente Tabla 6.9.

Tabla 6.9: Matriz de confusión con resultados de la clasificación con falsificación

Teléfono Móvil		Samsung					% de acierto
		Galaxy A3	Galaxy Ace Style	Galaxy S5 Neo	Galaxy S6	GT I9000	
Samsung	Galaxy A3	**191**	6	0	0	3	96%
	Galaxy Ace Style	4	**184**	0	0	12	92%
	Galaxy S5 Neo	0	0	**200**	0	0	100%
	Galaxy S6	0	0	0	**200**	0	100%
	GT I9000	4	27	1	2	**166**	83%

De la Tabla 6.9 se pueden hacer las siguientes observaciones: En comparación con la Tabla 6.4 que describe los resultados de la clasificación sin anonimización y sin falsificación se observa que los porcentajes de acierto sólo cambiaron para los modelos GT I9000 y Galaxy Ace Style.

En el caso del modelo Galaxy Ace Style el porcentaje de acierto bajó de un 96% a un 92%. Lo que quiere decir que después de anonimizar fotogramas pertenecientes al modelo Galaxy Ace Style y falsificarlos con el modelo GT I9000, el algoritmo tuvo más dificultades para agrupar correctamente esta clase.

Para el modelo GT I9000 se puede observar que el porcentaje de acierto fue del 83%, siendo un 10% menor con respecto a los experimentos de los vídeos sin alteraciones. De los 200 fotogramas que pertenecen a esta clase (150 originales y 50 falsificados) se clasificaron correctamente 166. En la clasificación de la Tabla

6.4 se clasificaron correctamente 186 de los 200 fotogramas. Por tanto, se puede concluir que 30 de los 50 fotogramas falsificados lograron su objetivo.

7. CONCLUSIONES Y TRABAJO FUTURO

7.1. Conclusiones

En este trabajo se han desarrollado dos técnicas anti-forenses para la alteración de un vídeo digital con formato MP4.

Inicialmente se ha realizado un estudio de los aspectos más importantes de los vídeos digitales. Se ha detallado el proceso de generación de un vídeo dependiendo del tipo de dispositivo, haciendo énfasis en los tipos de codificación y la compresión que intervienen en el procesamiento. Asimismo, se ha realizado un análisis de los contenedores multimedia que almacenan los datos e información del vídeo, señalando los aspectos más importantes de los átomos, su estructura y tipos existentes, Se ha detallado la especificación del contenedor multimedia MP4 con compresión de vídeo H.264 al ser el formato más utilizado por la mayoría de los dispositivos móviles.

Seguidamente se han comentado las técnicas de análisis forense orientadas a vídeos digitales, las razones que justifican el estudio de este tipo de técnicas forenses, así como las diversas ramas del mismo.

Posteriormente, se han revisado los principales trabajos relacionados sobre las distintas técnicas forenses enfocadas a vídeos digitales y de las técnicas anti-forenses que afectan la autenticidad de un vídeo. Las técnicas forenses se han clasificado en tres grupos: técnicas para el análisis de adquisición, técnicas para el análisis de compresión y análisis de manipulaciones. Por otro lado, las técnicas anti-forenses en vídeos se han clasificado de acuerdo al objetivo de la manipulación.

A continuación se han presentado las contribuciones de este trabajo que consiste de dos técnicas anti-forenses para vídeos digitales. En primer lugar se

presenta una técnica anti-forense para realizar la anonimización de un vídeo con formato MP4 y en segundo lugar una técnica anti-forense para falsificar la fuente de un vídeo con formato MP4. Ambas técnicas están compuestas por una serie de algoritmos que están basados en la descomposición del vídeo, el ruido del sensor y la transformada wavelet. Los algoritmos propuestos tienen como objetivo la extracción de fotogramas y del audio del vídeo, la eliminación de la huella del sensor de cada fotograma, la extracción del ruido residual o huella del sensor de cada fotograma, el cálculo del patrón del ruido del sensor y por último la falsificación de la identidad de un vídeo inyectando el patrón del ruido de otra cámara diferente a la que grabó el vídeo.

Finalmente, la evaluación de las dos técnicas anti-forenses se ha realizado con dos experimentos de identificación de la fuente donde se utilizaron los vídeos generados por cada técnica. Los resultados obtenidos fueron satisfactorios. En cuanto a la anonimización se alcanzó un 91.6% de acierto y en la falsificación se alcanzó un 60%. Por tanto, el presente trabajo contribuye notablemente a la investigación forense para evitar ser víctimas de robo de identidad y la posibilidad de ser culpados de manera injusta.

7.2. Trabajo Futuro

Como posibles trabajos futuros pueden señalarse los siguientes:

- Realización de más experimentos con el algoritmo de falsificación para realizar mejoras y alcanzar una tasa de acertividad mayor.

- Desarrollo de algoritmos forenses para detectar cuando un vídeo ha sido procesado con técnicas anti-forenses.

- Ampliar las funcionalidades de las técnicas anti-forenses presentadas, abordando otro tipos de contenedores multimedia como MOV que en la actualidad también es muy usado por los fabricantes de dispositivos

móviles.

- Crear métodos eficientes y efectivos, para detectar la edición, inserción, eliminación y sustitución de átomos.

- Implementar técnicas para detectar si un vídeo ha sido fragmentado a partir del análisis de la estructura de átomos que posee cada vídeo de una determinada marca y modelo.

- Crear métodos o herramientas integrales para identificar de forma precisa la manipulación de un vídeo, tanto de los metadatos como del contenido de audio y vídeo; de esta forma el usuario realizará una gestión centralizada, coherente que conllevará a tomar buenas decisiones bajo cualquier circunstancia.

7.3. Publicaciones

Además de los resultados obtenidos de los experimentos, la elaboración de este trabajo dio lugar a la siguiente publicación.

- "**Análisis de Metadatos en Vídeos Digitales de Dispositivos Móviles**" (con Esteban Alejandro Armas Vega, Ana Lucila Sandoval Orozco, Luis Javier García Villalba), en las Actas del *VIII Congreso Internacional de Computación y Telecomunicaciones*. Lima, Perú, Septiembre 21-23, 2016.

RESUMEN EN INGLÉS

8. INTRODUCTION

Technology evolves at a fast pace, originating astonishing projects and realities. For instance, mobile devices have become a must, due to their ability to provide immediate solutions to daylife needs. Mobile phone manufacturers are successful, not only because of selling bleeding technology, but also as a consequence of selling full-featured devices at affordable prices. Currently, many users own more than a single mobile devices (Smartphone, phablets, tablets etc.); as seen in a study in [Liu15] referring that at the end of 2016 approximately 44.7% of mobile device users will use a smartphone, going up to 2.060 million users (expanding the mobile devices market by 11.7%) and tablet users going up to 1,150 million.

According to a report presented by a leading manufacturer of networking equipment,Cisco Systems (Cisco) in February 2016 [CIS16], it is expected that by 2020 there will be 5,500 million users of mobile devices (70% of world population). For comparison, the estimated 2020 World population will be 7,800 million, according to the United Nations. In addition, Cisco estimates that in 2020, smartphones, laptops and tablets will generate 98% of global mobile data traffic , i.e. 366.8 exabytes per year; whilst, on present day (2015) they generate 89% of total mobile data, i.e. 44.2 exabytes per year. Mobile generated videos will cope most of mobile data usage, increasing the use of 4th and 5G enabled connections. Therefore, it is estimated that by 2020, 75% of mobile network traffic will be carrying high-def videos. That is 81 billion new images (28 images daily per person) and 7 billion videos (more than 2.5 videos per day and person) per capita worldwide.

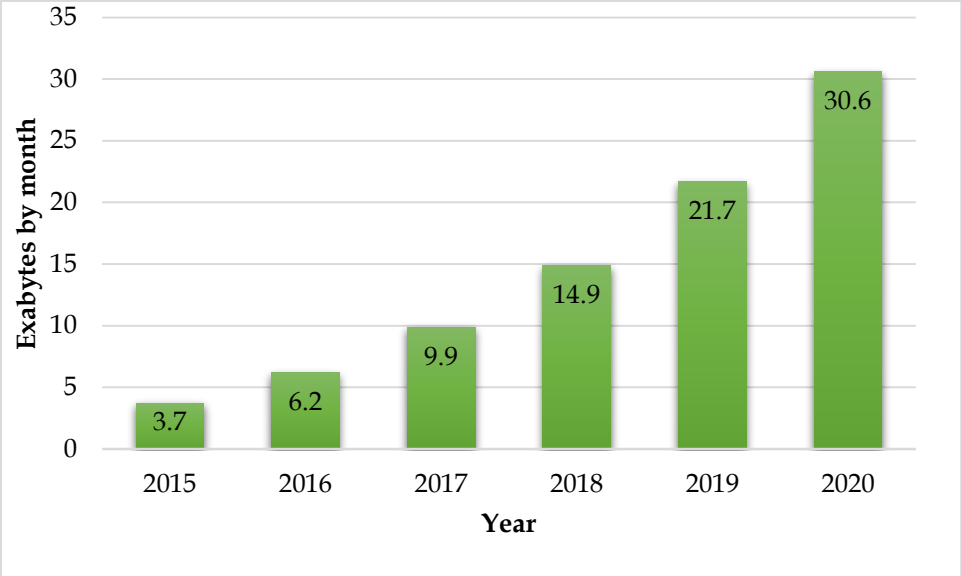


Figure 1.1 Mobile devices monthly traffic forecast, for 2015-2020.

A mobile device camera is a valuable equipment to an user, given the fact it enables video capturing scenes in fine details. Nonetheless, customers are demanding new features everyday. Manufacturers upgrade and innovate designs on regular basis: leaner devices, more natural interfaces, two main cameras, etc. Therefore, two pairs of a sensor and a lens enable a wider lens angle, a sharper picture (using a monochromatic sensor to capture light levels and another one to capture coloured components). In the near future, 3D imaging use will be widespread.

As a result, mobile devices are used to take pictures and videos, which additionally can be stored on external devices, posted on social networks, e-mailed, etc. This means that, from a forensic point of view, widespread presence of integrated cameras mobile devices and a vast amount of generated images and videos, facilitates criminal deeds such as data theft, child pornography, kidnapping, espionage, etc.,. In this regard, forgery detection and anonymization of images and videos, and source identification procedures are needed to determine criminal liabilities and collect evidence that will later be used in court. However, these forensic techniques are prone to suffer countermeasures such as anti-forensic techniques, image and video hidden

tampering, etc. More specifically, they try to hide, delete or tamper any evidence. This makes necessary to develop forensic techniques that can detect the use of these countermeasures and cope with them.

As we have seen, mobile devices pictures and videos forensic analysis and anti-forensics techniques, are extremely important issues that need thorough investigation in order to ensure the creation of appropriate forensic tools. These tools must assist criminal investigators. In the case of digital images and video, we ought to have extensive knowledge, through literature, of existing anti-forensic techniques oriented to anonymizing and tampering source of acquired image and digital videos.

7.1. Research Objective

Assuming that it want to design a lock safe enough that does not allow its opening without the original key, is necessary to know the techniques for open the lock without having the correct key. This way, it will be possible design a lock that avoids intensions of opening without having the key of the same one. Likewise, the knowledge acquired of the anti-forensic techniques to anonymize or forge the identity of the device that generated an image or digital video, they carry to the development of the forensic techniques for identification of the source more robust to avoid this type of attacks.

Research in recent years have been directed mostly to anti-forensic techniques and forensic digital imaging, leaving aside the digital videos. Both the forensic techniques of identification of the source, and the anti-forensic ones of anonimización and falsification of the identity of the device that generated, have focused solely on traditional cameras Digital Still Camera (DSC). However, cameras of mobile devices currently have replaced the DSC, for this is necessary to perform a study of them technical anti-forenses oriented to devices mobile.

The digital videos are composed by a sequence of images and audio. These videos can be generated by different devices which differ by peculiarities of each manufacturer, but all the videos look alike for having intrinsically the pattern of the noise of the sensor created during the process of generation of the same one, and used as a means of identification of the source. Specifically, the cameras digital of devices mobile have, mostly, with a type of sensor that leaves traits characteristic in each frame of a video.

The present research is focused in the study of them technical anti-forenses oriented to the anonymization and falsification of the source of a video. Specifically, it focuses on the exhaustive analysis of the atoms contained in a video format MP4, this will allow to extract the frames and the audio from the video to subsequently perform the anonymization and falsification of identity of the mobile device that generated the video in question.

The work proposes, first, develop an anti-forense technique for anonymize a MP4 format video; in this process are implemented two algorithms of decomposition of video: one for extraction of the elementary stream *Advanced Audio Coding* (AAC), which contains the audio, and another for the extract of frames from video atoms with MP4 format. One algorithm to remove the fingerprint of the sensor based on the noise *Photo Response Non Uniformity* (PRNU) and the recomposition of the anonymized MP4 format video. Secondly, one presents an anti-forensic technique to forge the source of a video with format MP4. For this purpose, implemented an algorithm to forge source of a video with format MP4, which re-uses algorithms used for anonimizar a video and an algorithm to forge the identity of the stills of a video with format MP4.

8. CONCLUSIONS AND FUTURE WORKS

8.1. Conclusions

With this project, we have developed two anti-forensic techniques, made for altering a digital video in MP4 format.

Firstly, a study has been performed on the essential aspects of a digital video. Digital Video Creation processes a video have been described on the basis of the type of device, emphasizing on types of encoding and compression used. Also, an analysis has been made, about multimedia containers which store data and video information, pinpointing the essential aspects of atoms, their structure and existing types. There is an explained specification of MP4 multimedia container, and of the video compression codec H.264, being most widely used on mobile phones.

Secondly, Video Oriented Forensic techniques and derivative work, have been described in this project, along with the reasons to study them.

Thirdly, some significant Anti-Forensic Digital Video Oriented works, techniques and derivatives, on the subject of authenticity a video, have been described in this work. The forensic techniques are being classified across three groups: acquisition analysis, compression analysis, tampering analysis. On the other hand, video anti-forensic techniques are being classified on the basis of tampering objective.

After that, Contributions are being explained, as two anti-forensic techniques. First one is n video anonymization oriented technique and second one is about tampering video source registry. Both techniques are made of a series of algorithms based on video deconstruction, sensor noise and wavelet transform. Proposed algorithms goals are audio and video extraction, sensor

fingerprint removal on every frame, the extraction of residual noise and sensor fingerprint, calculating sensor noise pattern, and lastly tampering video identity by injecting another camera noise pattern.

The evaluation of both anti-forensic techniques is being made through two source identification experiments, using videos generated by said anti-forensic techniques. Final results are satisfactory. Anonymization test got 91,6% success rate, whilst source tampering got 60%.

In conclusion, this project contribute significantly to Forensic Investigation, oriented towards avoiding identity theft and sentencing innocents

8.2. Future works

Plausible future works may be:

- Doing mode experiments and proof testing of the algorithms, in order to obtain a better assert rate
- Developing some forensic techniques in order to detect whether a video has been submitted to anti-forensic techniques
- Enhance present anti-forensic techniques, enabling support of additional video containers, such as MOV, which is also widely used nowadays
- Implement efficient and effective methods, in order to be able to detect whether one or more atoms have been edited, inserted, deleted, or replaces
- Implement techniques to detect whether a video has been fragmented, on the basis of an atoms structure analysis of a variety of brands and models
- Implement all-in-one technique or tools, for pinpointing video tampering, be it metadata or audio and video. This way, user would be able to take

faster decisions.

8.3. Publications

In addition to results in this work, its creation has produced the following publications:

- "**Análisis de Metadatos en Vídeos Digitales de Dispositivos Móviles**" (con Esteban Alejandro Armas Vega, Ana Lucila Sandoval Orozco, Luis Javier García Villalba), en las Actas del *VIII Congreso Internacional de Computación y Telecomunicaciones*. Lima, Perú, Septiembre 21-23, 2016.

ANEXOS

A. DESCRIPCIÓN DE LOS ÁTOMOS DE MOOV

En este anexo se procede a presentar los átomos contenidos en el átomo *moov* de la especificación.

A.1. Átomo Movie Header - mvhd

Este átomo es la cabecera del átomo *moov* y especifica todas las características de la película como la escala de tiempo, duración, características de visualización, etc. Contiene los campos que se indican en la Tabla A.1.

Tabla A.1: Estructura del átomo: *mvhd*

Size	Type	Version	Flags	Creation Time	Modification Time	Time Scale	Duration
0000006C	6D766864 (mvhd)	00	0			1000	
32 bits	32 bits	1 byte	3 byte	32 bits	32 bits	32 bits	32 bits

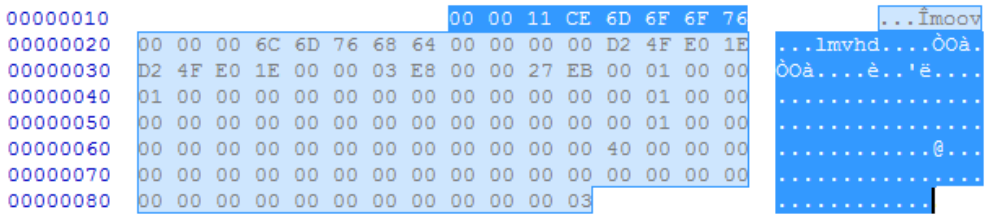
Preferred Rate	Preferred Volume	Reserved	Matrix	Predefines	Next Track ID
1.0 (normal)	1.0 (normal)	0	00010000 00000000 00000000 00000000 00010000 00000000 00000000 00000000 40000000	0	3
32 bits	16 bits	10 bytes	36 bytes (3x3 de 32 bits)	24 bytes	32 bits

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *movie header*; establecer este campo a "0x0000006C".
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representada como un código de cuatro caracteres; esta campo debe estar a *mvhd* (0x6D766864).
- **Version**: Una especificación de 1 byte de la versión de este átomo de cabecera de la película; establecer aquí a "0x00".
- **Flags**: Tres bytes de espacio para indicadores de encabezado de películas futuras; establecer aquí a "0x000000".
- **Creation Time**: Un entero de 32 bits que especifica la fecha del calendario y el tiempo (en segundos desde la medianoche del 1 de enero de 1904) cuando

el átomo película fue creado en el tiempo universal coordinado (UTC).

- **Modification Time**: Un entero de 32 bits que especifica la fecha y hora del calendario (en segundos desde medianoche, 1 de Enero, 1904) cuando el átomo película fue creada en el tiempo universal coordinado (UTC).
- **Time Scale**: Un valor de tiempo que indica la escala de tiempo para esta película, es decir, el número de unidades de tiempo que pasan por segundo en su sistema de coordenadas de tiempo.
- **Duration**: Valor de tiempo que indica la duración de la película en unidades de la escala de tiempo, derivada de las pistas de la película, lo que corresponde a la duración de la pista más larga de la película.
- **Preferred Rate**: Un número de coma fija de 32 bits que especifica la velocidad a la que reproducir esta película (un valor de 1,0 indica la velocidad normal cuyo valor sería "0x00010000").
- **Preferred Volume**: Número de coma fija de 16 bits que especifica el volumen para reproducir el sonido de esta película (Un valor de 1,0 indica que el volumen total cuyo valor sería "0x0100").
- **Reserved**: Diez bytes reservados, puestos a cero.
- **Matrix**: Una matriz de transformación que define la forma de asignar puntos de un espacio de coordenadas en otro espacio de coordenadas.
- **Predefines**: Cabecera media predefinida; se establece a cero.
- **Next Track ID**: El número ID de la pista siguiente; establecer aquí a "3".

La Figura A.1 muestra la estructura de los átomos *moov* y *mvhd* de un vídeo MP4:




```
ftyp
free
mdat
moov
    size |: 4558
    type |: moov
    mvhd
        size |: 108
        type |: mvhd
        version |: 0
        flags |: 0
        creationTime |: 23/10/2015-15:1:18
        modificationTime |: 23/10/2015-15:1:18
        timeScale |: 1000
        duration |: 10.219
        preferredRate |: 1.0
        preferredVolume |: 1.0
        reserved |: 0
        matrixStructure |: [65536, 0, 0, 0, 65536, 0, 0, 0, 1073741824]
        predefines |: 0
        NextTrackID |: 3
```

Figura A.1: Estructura de los átomos *moov* y *mvhd*

A.2. Átomos Track - trak

Este átomo define una pista de la película, cada pista es independiente de las otras pistas y lleva su propia información temporal y espacial. Cada pista lleva asociada unos átomos de medios. Encontramos dos átomos *trak*, uno de vídeo para la pista de vídeo y uno de audio para la pista de audio. Contiene los campos que se indican en la Tabla A.2:

Tabla A.2: Estructura del átomo: *trak*

Size	Type	tkhd	mdia
	7472616B (trak)		
32 bits	32 bits	átomo	átomo

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este primer átomo *Track*.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como

un código de cuatro caracteres; esta campo debe estar a *trak* (0x7472616B).

A.3. Átomo Track Header tkhd

Este átomo se encuentra en ambos átomos *Track*. Este átomo es la cabecera de las pistas de la película y contiene las características de las mismas, incluyendo información temporal, espacial y de volumen. Contiene los campos que se indican en la Tabla A.3.

Tabla A.3: Estructura del átomo: *tkhd*

Size	Type	Version	Flags	Creation Time	Modification Time	Track ID
0x0000005C	746B6864 (tkhd)	0x00	0x000007			
32 bits	32 bits	1 byte	3 byte	32 bits	32 bits	32 bits

Reserved	Duration	Reserved	Layer	Alternative Group	Volume	Reserved
0		0	0x0000	0x0000	1.0 (normal)	0
32 bits	32 bits	8 bytes	16 bits	16 bits	16 bits	16 bits

Matrix Structure	Track Width	Track Height
0x00010000 0x00000000 0x00000000 0x00000000 0x00010000 0x00000000 0x00000000 0x00000000 0x40000000	0x05000000 (1080)	0x02D00000 (720)
36 bytes (3x3 de 32 bits)	32 bits	32 bits

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes del átomo *track header*; establecer aquí a "0x0000005C'"
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representada como un código de cuatro caracteres; este campo debe estar a *tkhd*.
- **Version**: Una especificación de 1 byte de la versión del átomo de cabecera de esta primera pista de película; establecer aquí a "0x00".
- **Flags**: Tres bytes de especificación de la versión del átomo *track header*; establecer aquí a "0x000007".
- **Creation Time**: Un entero de 32 bits que especifica la fecha y hora del calendario (en segundos desde medianoche del 1 de enero de 1904) cuando el

átomo *track* fue creado en el tiempo universal coordenado (UTC).

- **Modification Time**: Un entero de 32 bits que especifica la fecha y hora del calendario (en segundos desde medianoche del 1 de enero de 1904) cuando el átomo *track* fue modificado en el tiempo universal coordinado (UTC).
- **Track ID**: Un número entero de 32 bits que identifica de forma exclusiva la pista; el valor "0" no se puede utilizar.
- **Reserved**: Un entero de 32 bits que está reservado; este campo se establece a "0x00000000".
- **Duration**: Un valor de tiempo que indica la duración de esta pista (en el sistema de coordenadas de tiempo de la película). Tenga en cuenta que esta propiedad se deriva de las modificaciones de la pista: el valor de este campo es igual a la suma de las duraciones de todas las ediciones de la pista y que, si no hay ninguna lista de edición, la duración es la suma de las duraciones de la muestra, convertida en la escala de tiempo de la película.
- **Reserved**: Un valor de 8 bytes que está reservado; este campo se establece a "0".
- **Layer**: Un entero de 16 bits que indica la prioridad espacial de esta pista (La caja de herramientas de la película QuickTime utiliza este valor para determinar cómo rastrea una superposición de otra). Las pistas con menor valor de capas se muestran frente a las pistas con los valores de capas superiores.
- **Alternative Group**: Un entero de 16 bits que especifica un conjunto de pistas que contienen datos alternativos del uno al otro; establecido aquí a "0x0000".
- **Volume**: Valor de coma fija de 16 bits que indica cómo de fuerte se va reproducir esta pista de audio; un valor de 1.0 indica que el volumen es normal.
- **Reserved**: Un entero de 16 bits que está reservado; este campo se establece a "0x0000".
- **Matrix Structure**: La estructura de la matriz asociada a esta pista.
- **Track Width**: Un número de coma fija de 32 bits que especifica el ancho de

esta pista en píxeles; establecer aquí a "0x05000000" para 1080.

- **Track Height**: Un número de coma fija de 32 bits que especifica la altura de esta canción en píxeles; establecer aquí a "0x02D00000" para 720.

La Figura A.2 muestra la estructura de los átomos *trak* y *tkhd* de un vídeo MP4.

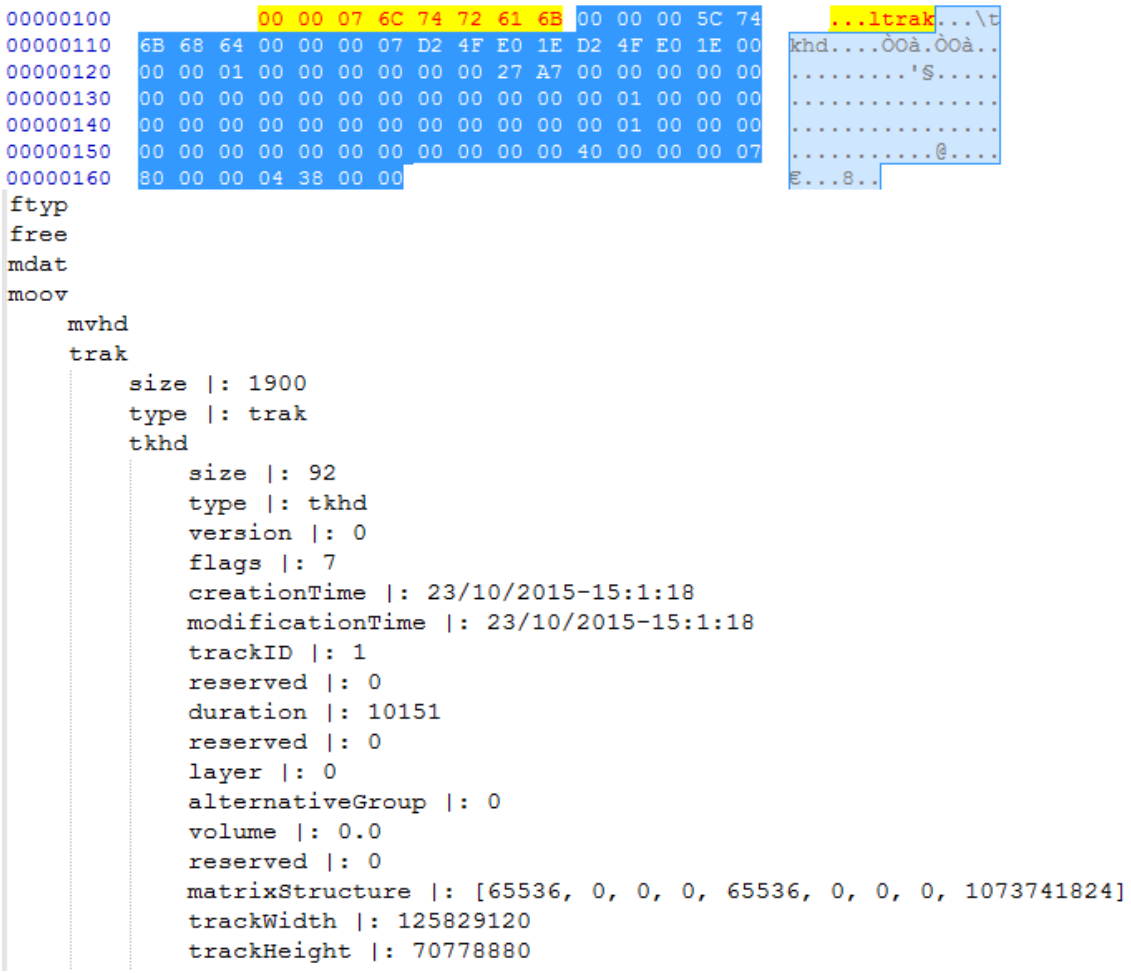


Figura A.2: Estructura de los átomos *trak* y *tkhd*

A.4. Átomo Media - mdia

Este átomo se encuentra en ambos átomos *track*. Define el tipo de medio y los datos de muestra de cada pista. Contiene los campos que se indican en la Tabla A.4.

Tabla A.4: Estructura del átomo: *mdia*

Size	Type	mdhd	hdlr	minf
	0x6D646961 (mdia)			
32 bits	32 bits	átomo	átomo	átomo

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este primer átomo *movie media*.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; esta campo debe estar a *mdia* (0x6D646961).

A.5. Media Header - mdhd

Este átomo se encuentra en ambos átomos *track*. Este átomo es la cabecera del átomo *media* y especifica las características de un medio, incluyendo escala de tiempo y duración. Contiene los campos que se indican en la Tabla A.5.

Tabla A.5: Estructura del átomo: *mdhd*

Size	Type	Version	Flags	Creation Time
00000020	0x6D646864 (mdhd)	0x00	0	
32 bits	32 bits	1 byte	3 byte	32 bits
Modification Time	**Time Scale**	**Duration**	**Languaje**	**Predefined**
				0x0000
32 bits	32 bits	32 bits	16 bits	16 bits

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este primer átomo *movie media header*; establecer aquí a "0x00000020".
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *mdhd* (0x6D646864).
- **Version**: Especificación de 1 byte de la versión de este primer átomo *movie media header*; establecer aquí a "0x00".
- **Flags**: Tres bytes que especifican la versión de este primer átomo *movie media header*; establecer aquí a "0x000000".
- **Creation time**: Un entero de 32 bits que especifica la fecha y hora del

calendario (en segundos desde medianoche del 1 de enero de 1904) cuando el átomo *movie media* fue creado en el tiempo universal coordinado (UTC).

- **Modification time**: Un entero de 32 bits que especifica la fecha del calendario y el tiempo (en segundos desde la medianoche del 1 de enero de 1904) cuando el átomo *movie media* fue creado en el tiempo universal coordinado (UTC).
- **Time Scale**: Un valor de tiempo que indica la escala de tiempo para este medio, es decir, el número de veces que pasan por segundo en sus unidades tiempo del sistema de coordenadas; para 30 fps establecer aquí a "0x0000001E".
- **Duration**: Duración de los medios en unidades de la escala de tiempo; Por ejemplo si vale "0x00002A17" representa 10.775 frames que son 10.775*30(*time scale* del átomo *mdhd*)/1000(*time scale* del átomo *mvhd*)=323,250 segundos.
- **Languaje**: Un entero de 16 bits que especifica el código de idioma para este medio.
- **Predefined**: Establecer aquí a "0x0000".

A.6. Átomo Media Handler References hdlr,

Este átomo se encuentra en ambos átomos *track*. Este átomo, referencia del manejador del medio, especifica el manejador que va a usar para interpretar los datos. Este átomo se mantiene por motivos históricos pero puede ser ignorado con seguridad. Contiene los campos que se indican en la Tabla A.6.

Tabla A.6: Estructura del átomo: *hdlr*

Size	Type	Version	Flags	Component Type	Component Subtype	Component Name
0000002C	0x68646C72 (hdlr)	0x00	0		vide soun	VideoHandle SoundHandle
32 bits	32 bits	1 byte	3 byte	32 bits	32 bits	24 bytes

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este

átomo *media handler references*; establecer aquí a "0x0000002C".

- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *hdlr* (0x68646C72).
- **Version**: Una especificación de 1 byte de la versión de este átomo; establecer aquí a "0x00".
- **Flags**: Tres bytes que especifican la versión de este átomo *media handler references*; establecer aquí a "0x000000".
- **Component Type**: Un código de cuatro caracteres que identifica el tipo del manipulador (*handler*) (normalmente sólo dos valores son válidos para este campo: *mhlr* para los manipuladores de los medios de comunicación y *dhlr* para los manipuladores de datos); establecer aquí a "0x6D686C72" para *mhlr* o "0x64686C72" para *dhlr*.
- **Component Subtype**: Un código de cuatro caracteres que identifica el tipo de manipulador de medios de comunicación o el manipulador de datos. Para los manipuladores de los medios de comunicación, este campo define el tipo de datos, por ejemplo, "0x76696465" (*vide*) para los datos de vídeo del primer átomo *track*, pista de vídeo o "0x76696465" (*soun*) para datos de sonido del segundo átomo *track*, pista de audio.
- **Component Name**: Una cadena que especifica el nombre del componente; establecer aquí a *videohandle* para el primer *track* (pista de vídeo) y *soundhandler* para el segundo *track* (pista de audio).

La Figura A.3 muestra la estructura de los átomos *mdia, mdhd* y *hdlr* de un vídeo MP4.

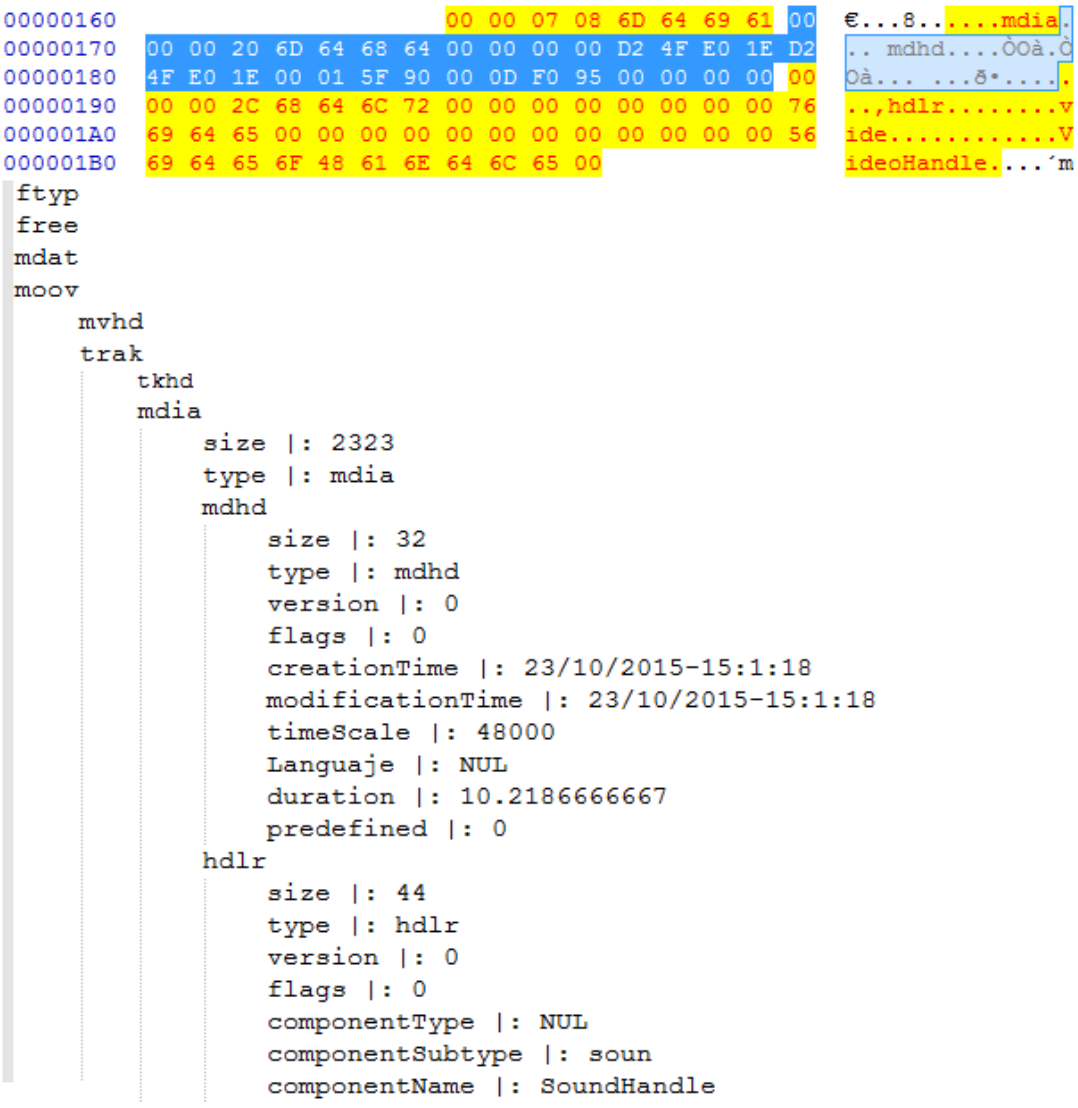


Figura A.3: Estructura de los átomos *mdia*, *mdhd* y *hdlr*

A.7. Átomo Media Information - minf

Este átomo contiene la información del medio. El manejador del medio usará esta información para asignar el tiempo del medio a los datos del medio y así poder procesar los datos de la pista. Contiene los campos que se indican en la Tabla A.7.

Tabla A.7: Estructura del átomo: *minf*

Size	Type	vmhd (vídeo track) smhd (sound track)	dinf	stbl
	6D696E66 (minf)			
32 bits	32 bits	átomo	átomo	átomo

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *minf* (0x6D696E66).

A.8. Átomo Vídeo Media Information Header - vmhd

Se encuentra en el *track* de vídeo. Este átomo es la cabecera del átomo *media information*, en este caso para la pista de vídeo y define las características específicas de los datos del medio de vídeo. Contiene los campos que se indican en la Tabla A.8.

Tabla A.8: Estructura del átomo: *vmhd*

Size	Type	Version	Flags	Graphics Mode	Opcolor
0x00000014	0x766D6864 (vmhd)	0x00	0x000001	0x0000	0x0000 0x0000 0x0000
32 bits	32 bits	1 byte	3 byte	16 bits	3x 16 bits

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *vídeo media information header*; establecer aquí a "0x00000014".

- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *vmhd* (0x766D6864).

- **Version**: Una especificación de 1 byte de la versión de este *vídeo media information header*; establecer aquí a "0x00".

- **Flags**: Un espacio de 3 bytes para indicadores de información sobre los medios de comunicación de vídeo; establecer aquí a "0x000001".

- **Graphics Mode**: Un entero de 16 bits que especifica el modo de transferencia; establecido aquí a "0x0000".

- **Opcolor**: Valores de 16 bits de tres colores, que especifica los colores rojo, verde y azul para la operación del modo de transferencia indicada en el campo de modo gráfico; todos establecidos a "0x0000".

A.9. Átomo Sound Media Information Header - smhd

Se encuentra en el *track* de audio. Este átomo es la cabecera del átomo *media information*, en este caso para la pista de audio y define las características específicas de los datos del medio de audio Contiene los campos que se indican en la Tabla A.9.

Tabla A.9: Estructura del átomo: *smhd*

Size	Type	Version	Flags	Balance	Reserved
0x00000014	0x736D6864 (smhd)	0x00	0	0x0000	0x0000
32 bits	32 bits	1 byte	3 byte	16 bits	16 bits

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *sound media information header*; establecer aquí a "0x00000014".
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *smhd* (0x736D6864).
- **Version**: Una especificación de 1 byte de la versión de este *sound media information header*; establecer aquí a "0x00".
- **Flags**: Un espacio de 3 bytes para indicadores de información sobre los medios de comunicación de audio; establecer aquí a "0x000001".
- **Balance**: Un entero de 16 bits que especifica el balance de sonido de este medio de sonido. El balance de sonido es el ajuste que controla la mezcla del sonido entre los dos altavoces de un ordenador. Este campo se establece normalmente a "0".
- **Reserved**: Un campo de 16 bits reservados. Establecer a "0x0000".

A.10. Átomo Data Information - dinf

Este átomo se encuentra en ambos átomos *track*. Este átomo es utilizado por el manejador del medio para interpretar los datos de dicho medio. Contiene los campos que se indican en la Tabla A.10:

Tabla A.10: Estructura del átomo: *dinf*

Size	Type	dref
0x00000024	0x64696E66 (dinf)	
32 bits	32 bits	Átomo

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *media data information*; establecer aquí a "0x00000024".
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *dinf* (0x64696E66).

A.11. Átomo Data References - dref

Este átomo se encuentra en ambos átomos *track*. Este átomo indica al componente del manejador (*handler component*) cómo acceder a los datos del medio. Contiene los campos que se indican en la Tabla A.11:

Tabla A.11: Estructura del átomo: *dref*

Size	Type	Version	Flags	Number of Entries	Data Reference
g0x0000001C	0x64726566 (dref)	0x00	0x000000	0x00000001	
32 bits	32 bits	1 byte	3 byte	32 bits	átomo

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *media data reference*; establecer aquí a "0x0000001C".
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *dref* (0x64726566).
- **Version**: Una especificación de 1 byte de la versión de este *media data reference*; establecer aquí a "0x00".

- **Flags**: Un espacio de 3 bytes para las banderas de este átomo *media data reference*.
- **Number of Entries**: Un entero de 32 bits que contiene el recuento de referencias de datos que siguen; establecer aquí a "0x000001".
- **Data References**: Una serie de referencias de datos: cada referencia de datos está formateado como un átomo de tipo *url* (para QuickTime existen otros dos tipo más: *alis* y *rsrc*), es una cadena que especifica una dirección URL y contiene los campos que se indican en la Tabla A.12.

Tabla A.12: Estructura del átomo: *url*

Size	Type	Version	Flags
0x0000000C	0x75726C20 (url)	0x00	0x000001
32 bits	32 bits	1 byte	3 byte

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo; establecer aquí a "0x0000000C".
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo se establece en *url* (0x75726C20).
- Version: Una especificación de 1 byte de la versión de este átomo; establecer aquí a "0x00".
- Flags: Un espacio de 3 bytes para las banderas; establecer aquí a "0x000001".

La Figura A.4 muestra la estructura de los átomos *minf*, *vmhd*, *dinf* y *dref* de un vídeo MP4:

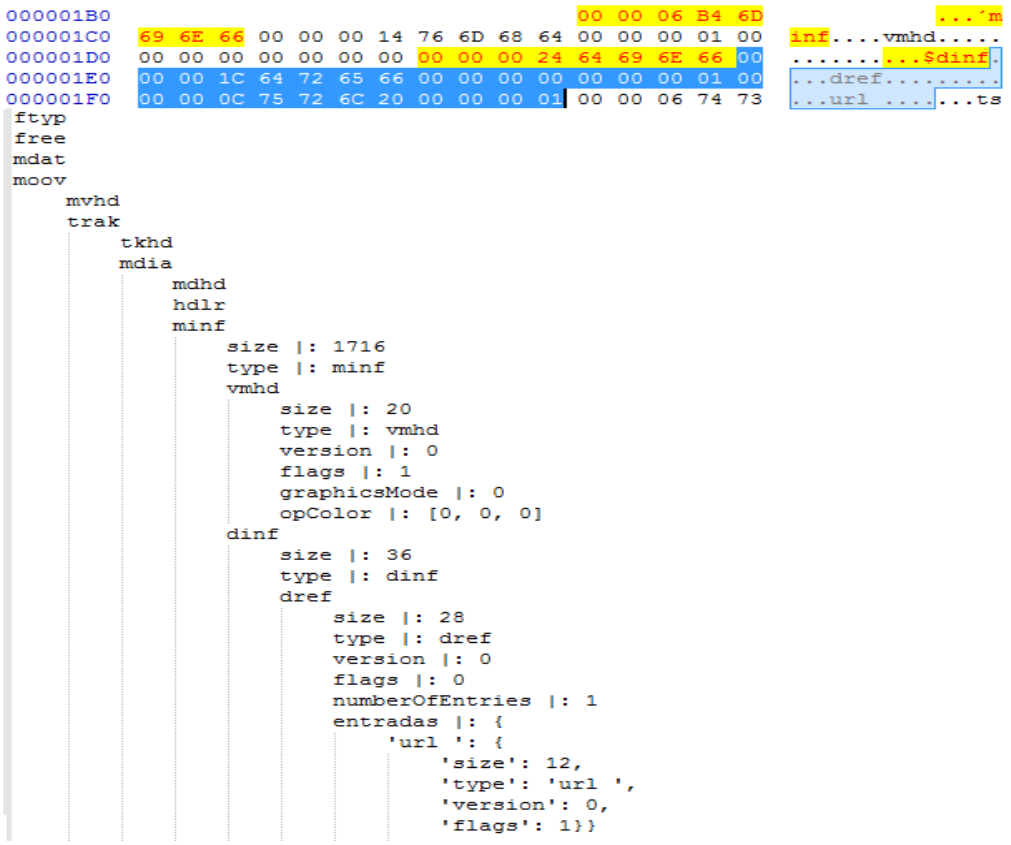


Figura A.4: Estructura de los átomos *minf*, *vmhd*, *dinf* y *dref*

A.12. Átomo Sample Table stbl

Este átomo se encuentra en ambos átomos *track*. Este átomo contiene información para la conversión de tiempo de medio a número de muestra para localizar la muestra. Este átomo indica cómo interpretar la muestra y describe el contenido y el formato de la tabla de muestras. Contiene los campos que se indican en la Tabla A.13:

Tabla A.13: Estructura del átomo: *stbl*

Size	Type	stsd	stts	stss	stsz	stsc	stco
	0x7374626C (stbl)						
32 bits	32 bits	átomo	átomo	átomo	átomo	átomo	átomo

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *sample table*.

- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *stbl* (0x7374626C).

A.13. Átomo Sample Description - stsd

Este átomo se encuentra en ambos átomos *track*. En la pista de vídeo Este átomo permite decodificar las muestras del medio de vídeo. Contiene los campos que se indican en la Tabla A.14.

Tabla A.14: Estructura del átomo *stsd* de la pista de vídeo

Size	Type	Version	Flags	Number of Entries	Sample Description Table	Data Format/Type
	73747364 (stsd)	0x00	0	00000001		0x61766331 (avc1)
32 bits	32 bits	1 byte	3 byte	32 bits	32 bits	32 bits

Reserved	Data Reference Index	Predefines	Reserved	Media Width	Media Height	Horizontal Resolution
0	1	0	0	0780 (1920)	0438 (1080)	00480000 (72.0)
6 bytes	16 bits	32 bits	12 bytes	16 bits	16 bits	32 bits

Vertical Resolution	Reserved	Frame Count	Size Compressor Name	Compressor Name	Depth	Color Table ID
00480000 (72.0)	0	1				
32 bits	32 bits	16 bits	16bits		16 bits	16 bits

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *vídeo sample description*.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *stsd* (0x73747364).
- **Version**: Una especificación de 1 byte de la versión de este átomo *sample description*; establecer aquí a "0x00".
- **Flags**: Un espacio de 3 bytes para las banderas del átomo *sample description*; establecer aquí a "0x000000".
- **Number of Entries**: Un entero de 32 bits que contiene el recuento de las

descripciones de muestras de vídeo que siguen; establecer aquí a "0x000001".

- **Sample Description Size**: Un entero de 32 bits que indica el número de bytes en este átomo *sample description*; establecer aquí a "0x0000FF".
- **Data Format/Type**: Un valor de 32 bits que indica el formato de los datos almacenados: esto depende del tipo de medio, pero es por lo general el formato de compresión o el tipo de soporte. La Tabla A.15 muestra los tipos de compresión de vídeo y una descripción de dicho tipo:
- **Reserved**: Seis bytes que se deben establecer a "0x00000000".
- **Data Reference Index**: Un entero de 16 bits que contiene el índice de la referencia de los datos a utilizar para recuperar datos asociados con las muestras que utiliza esta muestra de descripción (las referencias de datos se almacenan en los átomos de referencia de datos); establecido aquí a "0x0001".
- **Predefines**: Valores definidos aquí a "0".
- **Reserved**: Un campo de 12 bytes reservados, establecidos a "0".
- **Media Width**: Anchura de vídeo, "0x0780" para 1920 píxeles.
- **Media Height**: Altura de vídeo, "0x0438" para 1080 píxeles.
- **Horizontal Resolution**: Resolución horizontal de vídeo, "0x00480000" para 72 píxeles/pulgada.
- **Vertical Resolution**: Resolución vertical de vídeo, "0x00480000" para 72 píxeles/pulgada.
- **Reserved**: Campo de 32 bits reservados, establecidos a "0".
- **Frame Count**: Un entero de 16 bits que indica cómo se almacenan los frames de datos comprimidos en cada muestra; establecido aquí a "1".

- **Size Compressor Name**: Un entero sin signo de 16 bits que indica el tamaño del nombre del compresor (en este tamaño se incluye los 16 bits de este campo).

- **Compressor Name**: Cadena de caracteres que indican el nombre del compresor, su tamaño se indica en el campo anterior (menos 16 bits del propio campo).

- **Depth**: Un entero sin signo de 16 bits que indica la profundidad de píxeles de la imagen comprimida. Los valores de 1, 2, 4, 8, 16, 24, y 32 indican la profundidad de las imágenes en color. El valor 32 se debe utilizar sólo si la imagen contiene un canal alfa. Los valores de 34, 36 y 40 indican la escala de grises de 2, 4, y 8 bits, respectivamente, para las imágenes en escala de grises.

- **Color Table ID**: Un entero sin signo de 16 bits que indica qué tabla de colores se va a utilizar.

Tabla A.15: Tipos de compresión y sus respectivas descripciones

Tipo	Descripción		Tipo	Descripción		Tipo	Descripción
'cvid'	Cinepak		**'tiff'**	Tagged Image File Format		**'v216'**	Sin comprimir Y´CbCr, 10, 12, 14, o 16-bit por componente 4:2:2
'rle '	Animation		**'rpza'**	Apple vídeo		**'v210'**	Sin comprimir Y´CbCr, 10-bit-por componente 4:2:2
'SVQ3'	Sorenson vídeo 3		**'dvc '**	NTSC DV-25 vídeo		**'2vuY´**	Sin comprimir Y´CbCr, 8-bit-por componente 4:2:2
'dvcp'	PAL DV-25 vídeo		**'mjpb'**	Motion-JPEG (format B)		**'v408'**	Sin comprimir Y´CbCr, 8-bit-por componente 4:4:4:4
'avc1'	H.264 vídeo		**'raw '**	Uncompressed RGB		**'yuv2'**	Sin comprimir Y´CbCr, 8-bit-por componente 4:2:2
'smc '	Graphics		**'SVQ1'**	Sorenson vídeo, version 1		**'v410'**	Sin comprimir Y´CbCr, 10-bit-por componente 4:4:4
'jpeg'	JPEG		**'mp4v'**	MPEG-4 vídeo		**'png '**	Portable Network Graphics
'h263'	H.263 vídeo		**'mjpa'**	Motion-JPEG		**'v308'**	Sin comprimir Y´CbCr, 8-bit-por componente

				(format A)			4:4:4
			'kpcd'	Kodak Photo CD		**'gif '**	CompuServe Graphics Interchange Format

Análogamente, el átomo *sample description* en el *track* de audio permite decodificar las muestras del medio de audio. Contiene los campos que se indican en la Tabla A.16.

Tabla A.16: Estructura del átomo *stsd* de la pista de sonido

Size	**Type**	**Version**	**Flags**	**Number of Entries**	**Sample Description Table**	**Data Format/Type**
91 bytes	0x73747364 (stsd)	0x00	0	0x00000001	0x0000004B	0x6D703461 (mp4a)
32 bits	32 bits	1 byte	3 byte	32 bits	32 bits	32 bits

Reserved	**Data Reference Index**	**Version**	**Revision Level**	**Vendor**
	1		0	0
6 bytes	16 bits	16 bits	16 bits	32 bits

Channels	**Sample Size**	**Commpression ID**	**Packet Size**	**Sample Rate**
	8 o 16	0	0	
16 bits	16 bits	16 bits	16 bits	32 bits

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *sound sample description*; establecer aquí a "0x0000010F".
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *stsd* (0x73747364).
- **Version**: Una especificación de 1 byte de la versión de este átomo *sample description*; establecer aquí a "0x00".
- **Flags**: Un espacio de 3 bytes para las banderas del átomo *sample description*; establecer aquí a "0x000000".
- **Number of Entries**: Un entero de 32 bits que contiene el recuento de las descripciones de muestras de vídeo que siguen; establecer aquí a "0x000001".
- **Sample Description Size**: Un entero de 32 bits que indica el número de bytes en este *sample description*; establecer aquí a "0x0000FF".

- **Data Format/Type**: Un valor de 32 bits que indica el formato de los datos almacenados: esto depende del tipo de medio, pero es por lo general el formato de compresión o el tipo de soporte; establecer aquí a *mp4a*. La Tabla A.17 muestra los tipos de compresión de audio y una descripción de dicho tipo.
- Reserved: Seis bytes que se deben establecer a "0x00000000".
- Data Reference Index: Un entero de 16 bits que contiene el índice de la referencia de los datos a utilizar para recuperar datos asociados con las muestras que utiliza esta muestra de descripción (las referencias de datos se almacenan en los átomos de referencia de datos); establecido aquí a "0x0001".
- Version: Un entero de 16 bits que indica la versión del códec *mp4a*.
- Revision Level: Entero de 16 bits puestos a "0".
- Vendor: Entero de 16 bits puestos a "0".
- Channels: Entero de 16 bits que indica el número de canales de audio usados por la muestra de sonido. 1 para sonido monoaural y 2 para sonido estéreo. Número de canales más altos no son soportados.
- Sample Size: Un entero de 16 bits que indica el número de bits de cada muestra de audio sin comprimir. Los valores soportados son 8 o 16.
- Compression ID: Un entero de 16 bits puestos a "0" para la versión 0.
- Packet Size: Un entero de 16 bits puestos a "0".
- Sample Rate: Un número de 32 bits sin signo de punto fijo (16,16) que indica la velocidad a la que las muestras de sonido deben ser adquiridas.

  La mayoría de los archivos tienen valores enteros, como "44100". *sample rates* mayores que 2 ^ 16 no son compatibles.

Tabla A.17: Tipos de compresión y sus respectivas descripciones

Formato	Código	Descripción
No especificado	0x00000000	Este formato no debe ser utilizado pero lo podemos encontrar en algunos archivos. La muestra es almacenada con el formato "raw" o "twos", dependiendo del campo "size" de la muestra en la descripción del audio.
kSoundNotCompressed	'NONE'	Este formato no debe ser utilizado pero lo podemos encontrar en algunos archivos. La muestra es almacenada con el formato "raw" o "twos", dependiendo del campo "size" de la muestra en la descripción del audio.
k8BitOffsetBinary- Format	'raw '	Las muestras son almacenadas sin comprimir, en un formato de desplazamiento binario (rango de valores: 0-255; 128 es silencio). Estos son almacenados como desplazamientos binarios de 8 bits.
k16BitBigEndianFormat	'twos'	Las muestras son almacenadas sin comprimir, el formato "twos-complement" (los valores van de 128-127 para audio de 8 bits y de -32768-32767 para audio de un bit; 0 siempre es silencio). Estas muestras son almacenadas en formato big-endian de 16 bits.
k16BitLittleEndian- Format	'sowt'	16-bit little-endian, twos-complement
kMACE3Compression	'MAC3 '	Las muestras son comprimidas usando MACE 3:1. (Obsoleto.)
kMACE6Compression	'MAC6 '	Las muestras son comprimidas usando MACE 6:1. (Obsoleto.)
kIMACompression	'ima4'	Las muestras son comprimidas usando IMA 4:1.
kFloat32Format	'fl32'	32-bit coma flotante
kFloat64Format	'fl64'	64-bit coma flotante
k24BitFormat	'in24'	24-bit entero
k32BitFormat	'in32'	32-bit entero
kULawCompression	'ulaw'	uLaw 2:1
kALawCompression	'alaw'	uLaw 2:1
kMicrosoftADPCMFormat	0x6D730002	Microsoft ADPCM-ACM code 2
kDVIIntelIMAFormat	0x6D730011	DVI/Intel IMAADPCM-ACM code 17
kDVAudioFormat	'dvca'	DV Audio
kQDesignCompression	'QDMC'	QDesign music
kQDesign2Compression	'QDM2'	QDesign music version 2
kQUALCOMMCompression	'Qclp'	QUALCOMM PureVoice
kMPEGLayer3Format	0x6D730055	MPEG-1 capa 3, CBR solo (pre-QT4.1)
kFullMPEGLay3Format	'.mp3'	MPEG-1 capa 3, CBR & VBR (QT4.1 y posteriores)
kMPEG4AudioFormat	'mp4a'	MPEG-4, Advanced Audio Coding (AAC)
kAC3AudioFormat	'ac-3'	Digital Audio Compression Standard (AC-3, Enhanced AC-3)

A.14. Átomos Sample Description Extension

Estos átomos son una extensión del átomo *stsd* y por la tanto están contenidos dentro del mismo. Cada uno de estos átomos puede aparecer o no. En la Tabla A.18 se puede encontrar los posibles átomos *sample description extension*.

Tabla A.18: Tipo de átomos *sample description extension* y sus respectivas descripciones

Tipo de extensión	Descripción
'gama'	Un número de coma fija de 32 bits que indica el nivel de gamma en la que se capturó la imagen. El descompresor puede utilizar este valor para el gamma-correct en tiempo de visualización.
'fiel'	Dos números enteros de 8 bits que definen el campo "handling". Esta información es utilizada por las aplicaciones para modificar los datos de imagen descomprimidos o por componentes del descompresor para determinar el orden de visualización de campo. Esta extensión es obligatoria para todos los formatos de datos no comprimidos YCbCr. El primer byte especifica el número de campos, y puede ser ajustado a 1 ó 2. Un valor de 1 se utiliza para las imágenes de escaneo progresivo; un valor de 2 indica que las imágenes entrelazadas. Cuando el recuento de campo es 2, el segundo byte especifica el orden de campo: qué campo contiene la línea de exploración más alta, qué campo debe mostrarse antes, y cuál se almacena primero en cada muestra. Cada muestra se compone de dos imágenes comprimidas diferentes, cada una con una codificación: el campo con la línea de exploración superior, T, y el otro campo, B. A continuación se definen las variantes permitidas: 0 - Sólo hay un campo. 1 - T se muestra más temprana, T se almacena primero en el archivo. B se visualiza como muy pronto, B se almacena primero en el archivo - 6. B se visualiza como muy pronto, T se almacena primero en el archivo - 9. 14 - T se muestra más temprana, B se almacena primero en el archivo.
'mjqt'	La tabla de cuantificación por defecto para un flujo de datos Motion-JPEG.
'mjht'	La tabla de Huffman por defecto para un flujo de datos Motion-JPEG.
'esds'	Un átomo elemental para MPEG-4. Se requiere esta extensión para vídeo MPEG-4.
'avcC'	Se requiere esta extensión para vídeo H.264 como se define en la norma ISO / IEC 14496-15.
'pasp'	Pixel aspect ratio. Esta extensión es obligatoria para los formatos de vídeo que utilizan píxeles no cuadrados, si no es opcional.
'colr'	Color parameters. Una extensión de la descripción de la imagen requerida para todos los tipos de vídeo sin comprimir Y´CbCr.
'clap'	Clean apertura. Relación espacial de los componentes YCbCr con relación a un centro de la imagen canónica. Esto permite una alineación precisa para la composición de imágenes de vídeo capturadas utilizando diferentes sistemas. Esta es una extensión obligatoria para todos los formatos de datos no comprimidos YCbCr.

Como en este documento se está analizando los vídeos mp4, con compresión H.264, se detallarán los átomos *esds*, *avcC* y *pasp*: El átomo *esds* aparece como extensión del átomo *stsd* del *track* de sonido, cuando el tipo de códec es *mp4a* (puede aparecer también en el *track* de vídeo, si el tipo de códec del átomo *stsd* es *mp4v*). El átomo *avcC* aparece como extensión del átomo *stsd* del *track* de vídeo cuando el tipo de códec es *avc1*, al igual que el átomo *pasp*, que como especificamos anteriormente es opcional ya que se utilizan pixeles cuadrados.

Elementary Stream Descriptor - esds: Este átomo contiene un descriptor de flujo elemental para MPEG-4. Este átomo es una extensión requerida para la muestra de vídeo MPEG-4. Contiene los campos que se indican en la Tabla A.19.

Tabla A.19: Estructura del átomo: *esds*

Size	Type	Version	Flags	Elementary Stream Descriptor
	0x65736473 (esds)	0	0	
32 bits	32 bits	8 bits	24 bits	

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *elementary stream descriptor*.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *esds* (0x65736473).
- **Version**: Una especificación de 1 byte de la versión de este *elementary stream descriptor*; establecer aquí a "0x00".
- **Flags**: Un espacio de 3 bytes para las banderas de *elementary stream descriptor*; establecer aquí a "0x000000".
- **Elementary Stream Descriptor**: Campo donde se describe el *elementary stream descriptor*.

AVC Decoder Configuration - avcC: Este átomo contiene un átomo de configuración del decodificador MP4, es una extensión requerida para la descripción de la muestra de vídeo de los vídeos H.264. Esta extensión sólo aparece cuando el tipo de códec de vídeo es *avc1*. Contiene los campos que se indican en la Tabla A.20.

Tabla A.20: Estructura del átomo: *avcC*

Size	Type	AVC Decoder Configuration Record
	0x61766343 (avcC)	
32 bits	32 bits	

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *AVC decoder configuration*;
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *avcC* (0x61766343).
- **AVC Decoder Configuration Record**: Campo donde se describe el *AVC decoder configuration record*.

Pixel Aspect Ratio, abreviatura *pasp*. Este átomo especifica la relación entre altura y anchura de los píxeles de la muestra de vídeo. Es una extensión requerida para los formatos de vídeo MPEG-4 e Y'CbCr sin comprimir cuando se utilizan píxeles no cuadrados, cuando los píxeles son cuadrados esta extensión es opcional. Contiene los campos que se indican en la Tabla A.21.

Tabla A.21: Estructura del átomo: *pasp*

Size	Type	hSpacing	vSpacing
0x00000010	0x70617370 (avcC)	0x00010000	0x00010000
32 bits	32 bits	32 bits	32 bits

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *pixel aspect ratio*;
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *pasp* (0x70617370).
- **hSpacing**: Un float sin signo de 32 bits (16,16) que especifica el espaciado

horizontal de píxeles.

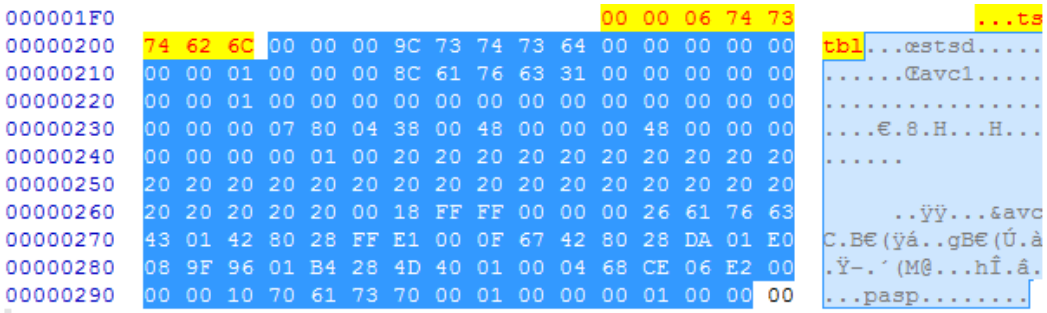
- **vSpacing**: Un float sin signo de 32 bits (16,16) que especifica el espaciado vertical de píxeles.

En la Tabla A.22 se encuentran las relaciones de aspecto de píxeles más comunes.

Tabla A.22: Relaciones de aspecto de píxeles más comunes

4:3 pixeles cuadrados (NTSC compuesto o PAL)	1	1
4:3 no cuadrados 525 (NTSC)	10	11
4:3 no cuadrados 625 (PAL)	59	54
16:9 análogo (NTSC compuesto o PAL)	4	3
16:9 digital 525 (NTSC)	40	33
16:9 digital 625 (PAL)	118	81
1920x1035 HDTV (para SMPTE 260M-1992)	113	118
1920x1035 HDTV (para SMPTE RP 187-1995)	1018	1062
1920x1080 HDTV o 1280x720 HDTV	1	1

La figura A.5 muestra la estructura de los átomos *stbl* y *stsd* de un vídeo MP4.



```
000001F0                                        00 00 06 74 73          ...ts
00000200  74 62 6C 00 00 00 9C 73 74 73 64 00 00 00 00 00  tbl...œstsd.....
00000210  00 00 01 00 00 00 8C 61 76 63 31 00 00 00 00 00  ......Œavc1.....
00000220  00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000230  00 00 00 07 80 04 38 00 48 00 00 00 48 00 00 00  ....€.8.H...H...
00000240  00 00 00 00 01 00 20 20 20 20 20 20 20 20 20 20  ......
00000250  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000260  20 20 20 20 20 00 18 FF FF 00 00 00 26 61 76 63       ..ÿÿ...&avc
00000270  43 01 42 80 28 FF E1 00 0F 67 42 80 28 DA 01 E0  C.B€(ÿá..gB€(Ú.à
00000280  08 9F 96 01 B4 28 4D 40 01 00 04 68 CE 06 E2 00  .Ÿ–.´(M@...hÎ.â.
00000290  00 00 10 70 61 73 70 00 01 00 00 00 01 00 00 00  ...pasp........
```

```
ftyp
free
mdat
moov
    mvhd
    trak
        tkhd
        mdia
            mdhd
            hdlr
            minf
                dinf
                    dref
                stbl
                    size |: 1652
                    type |: stbl
                    stsd
                        size |: 156
                        type |: stsd
                        version |: 0
                        flags |: 0
                        numberOfEntries |: 1
                        avc1
                            size |: 140
                            type |: avc1
                            reserved |: 0
                            dataReferenceIndex |: 1
                            Predefines |: 0
                            reserved |: 0
                            mediaWidth |: 1920
                            mediaHeight |: 1080
                            horizontalResolution |: 72.0
                            verticalResolution |: 72.0
                            reserved |: 0
                            frameCount |: 1
                            SizeCompressorName |: 32
                            CompressorName |:
                            Depth |: 24
                            ColorTableID |: 65.535
                            avcC
                                size |: 38
                                type |: avcC
                                AVC Decoder Configuration Record |:
                            pasp
                                size |: 16
                                type |: pasp
                                hSpacing |: 1
                                vSpacing |: 1
```

Figura A.5: Estructura de los átomos *stbl* y *stsd*

A.15. Átomo Sample-to-Time Table - stts

Éste átomo se encuentra en ambos *trak*. Este átomo contiene la información de la duración de las muestras del medio, proporcionando en un solo mapeo el tiempo que corresponde a los datos de las muestras. Cada entrada de la muestra indica el número de muestras consecutivas con el mismo delta de tiempo y el delta de esas muestras. Mediante la adición de los deltas se consigue un mapa de tiempo completo de las muestras, siendo DT(n+1)=DT(n)+STTS(n). Contiene los campos que se indican en la Tabla A.23.

Tabla A.23: Estructura del átomo: *stts*

Size	Type	Version	Flags	Number of Entries	Entries[Sample Count,Sample Duration]	
	0x73747473 (stts)	0x00	0x000000			
32 bits	32 bits	1 byte	3 byte	32 bits	32 bits	32bits

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *sample-to time table*;
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe establecerse a *stts* (0x73747473).
- **Version**: Una especificación de 1 byte de la versión de esta *sample-to time table*; establecer aquí a "0x00".
- **Flags**: Un espacio de 3 bytes para las banderas de *sample-to time table*; establecer aquí a "0x000000".
- **Number of Entries**: Un entero de 32 bits que contiene el recuento de las entradas de *sample-to time table* que seguir. Cada entrada se compone de dos campos: *sample duration* y *sample count*.
- **Sample Duration**: Un entero de 32 bits que especifica la duración de cada muestra.
- **Sample Count**: Un número entero de 32 bits que especifica el número de muestras consecutivas que tienen la misma duración.

A.16. Átomo Sync Sample - stss

Este átomo sólo está presente en el *track* de vídeo. Este átomo identifica los frames clave para el medio (Los I-frames descritos en el capítulo 2 de este documento). Si esta muestra no está presente todos los frames son clave (I-frames). Contiene los campos que se indican en la Tabla A.24:

Tabla A.24: Estructura del átomo: *stss*

Size	Type	Version	Flags	Number of Entries	Sample Duration
	0x73747373 (stss)	0x00	0x000000		
32 bits	32 bits	1 byte	3 byte	32 bits	X entradas de 32 bits

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *sync sample*.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe establecerse a *stss* (0x73747373).
- **Version**: Una especificación de 1 byte de la versión de este átomo *sync sample*; establecer aquí a "0x00".
- **Flags**: Un espacio de 3 bytes para las banderas del átomo *sync sample*; establecer aquí a "0x000000".
- **Number of Entries**: Un entero de 32 bits que contiene el recuento de las entradas de *sync sample table* que seguir.
- **Sample Duration**: Un entero de 32 bits que especifica la duración de cada muestra.

La Figura A.6 muestra la estructura de los átomos *stts* y *stss* de vídeos MP4:

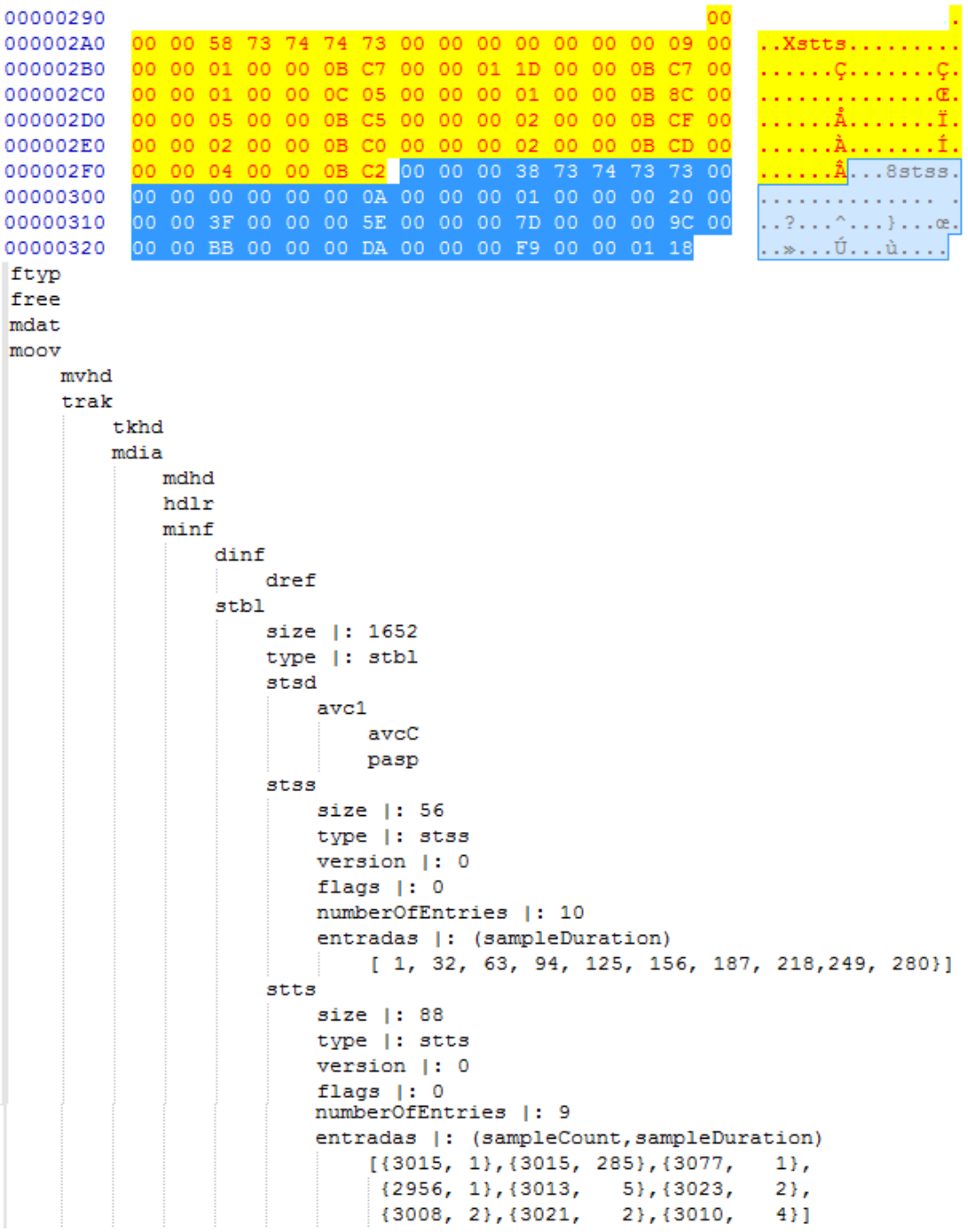


Figura A.6: Estructura de los átomos *stts* y *stss*

A.17. Átomo Sample-to-Chunk Table - stsc

Este átomo se encuentra en ambos átomos *track*. Las muestran se recogen en trozos permitiendo así un acceso óptimo a los datos. Un trozo contiene una o más muestras, estos trozos pueden ser de diferentes tamaños al igual que las

muestras que pueden contener. Este átomo contiene una tabla que asigna muestras a trozos en el flujo de datos del medio, con él se puede determinar el trozo que contiene una muestra específica. Contiene los campos que se indican en la Tabla A.25.

Tabla A.25: Estructura del átomo: *stsc*

Size	Type	Version	Flags	Number of Entries	Sample-to-Chunk Table
	0x73747363 (stsc)	0x00	0		
32 bits	32 bits	1 byte	3 byte	32 bits	Lista de x entradas de 96 bits (32 bits primer campo, 32 bits muestra, 32 bits ID)

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *sample-to-chunk*.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *stsc* (0x73747363).
- **Version**: Una especificación de 1 byte de la versión de este átomo *sample-to-chunk*; establecer aquí a "0x00".
- **Flags**: Un espacio de 3 bytes para las banderas del *sample-to-chunk*; establecer aquí a "0x000000".
- **Number of Entries**: Un entero de 32 bits que contiene el recuento de las entradas *time-to-chunk* a seguir.
- **Sample-to-Chunk Table**: La tabla que asigna muestras a trozos. Cada átomo de muestra a trozo contiene una tabla de este tipo, que identifica el trozo para cada muestra en un medio de comunicación. Cada entrada de la tabla contiene un primer campo trozo, un trozo muestras por campo, y un campo ID de descripción de la muestra. A partir de esta información, se puede determinar que las muestras residen en los datos de los medios.

A.18. ÁtomoSample Sizes - stsz

Este átomo se encuentra en ambos átomos *track*. Este átomo especifica el tamaño

de cada muestra del medio Contiene los campos que se indican en la Tabla A.26.

Tabla A.26: Estructura del átomo: *tkhd*

Size	Type	Version	Flags	Sample Size	Number of Entries	Sample-to-Chunk Table
	0x7374737A (stsz)	0x00	0			
32 bits	32 bits	1 byte	3 byte	32 bits	32 bits	Lista de x entradas de 32 bits(primer campo,muestra,ID)

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *sample size*.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *stsz* (0x7374737A).
- **Version**: Una especificación de 1 byte de la versión de este átomo *sample sizes*; establecer aquí a "0x00".
- **Flags**: Un espacio de 3 bytes para las banderas de *sample sizes*; establecer aquí para "0x000000".
- **Sample Size**: Un entero de 32 bits que especifica el tamaño de la muestra: si todas las muestras son del mismo tamaño, este campo contiene ese valor de tamaño. Si este campo se establece a "0", a continuación, las muestras tienen diferentes tamaños, y los tamaños se almacenan en la tabla de tamaños de muestra.
- **Number of Entries**: Un entero de 32 bits que contiene el recuento de la tabla de entradas *time-to-chunk* que seguir.
- **Sample-to-Chunk Table**: La tabla que asigna muestras a trozos. Cada átomo de muestra a pedazo contiene una tabla de este tipo, que identifica el trozo para cada muestra en un medio de comunicación. Cada entrada de la tabla contiene un primer campo trozo, un trozo muestras por campo, y un campo ID de descripción de la muestra. A partir de esta información, se puede determinar que las muestras residen en los datos de los medios

La Figura A.7 muestra la estructura de los átomos *stsc* y *stsz* de los vídeos MP4:

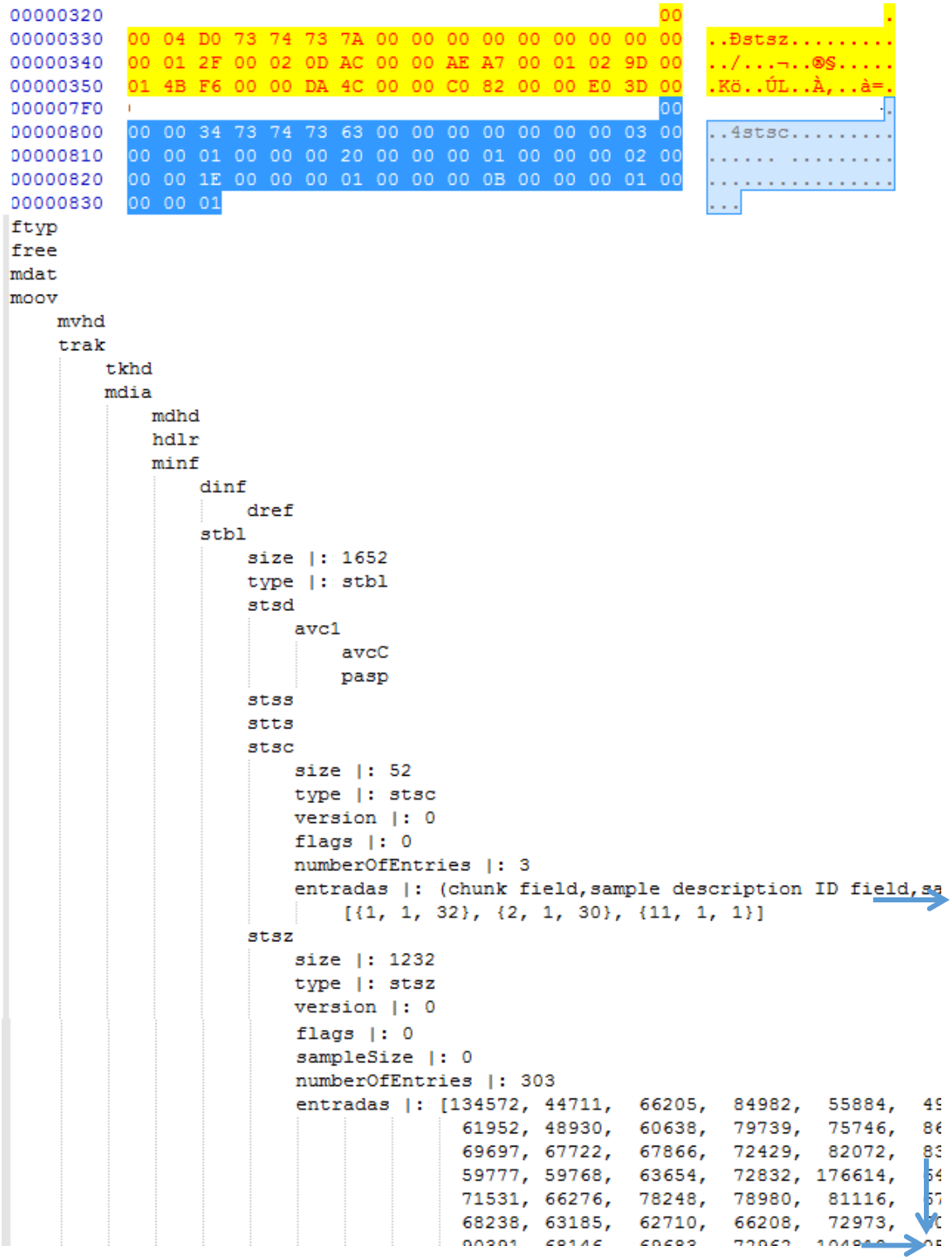


Figura A.7: Estructura de los átomos *stsc* y *stsz*

A.19. Átomo Chunk Offset

Este átomo puede aparecer como *stco* o *co64* y se encuentra en ambos átomos *track*. Este átomo identifica cada trozo de dato en el flujo de datos del medio. La tabla de desplazamiento de trozos da el índice de cada trozo contenido en el archivo. Hay dos variantes permitas, desplazamientos de 32 o 64 bits, siendo este último útil para películas muy grandes. Contiene los campos que se indican en la Tabla A.27.

Tabla A.27: Estructura del átomo: *stco/co64*

Size	Type	Version	Flags	Number of Entries	Datos
	0x7374636F (stco) 0x636F3634 (co64)	0x00	0		
32 bits	32 bits	1 byte	3 byte	32 bits	X entradas de: Si type="co64" 64 bits Si type="stco" 32 bits

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes de este átomo *sample offset table*.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *stco* o *co64* (0x7374636F o 0x636F3634).
- **Version**: Una especificación de 1 byte de la versión de este átomo *sample offset table*; establecer aquí a "0x00".
- **Flags**: Un espacio de 3 bytes para las banderas de átomos *sample offset table*; establecer aquí a "0x000000".
- **Number of Entries**: Un entero de 32 bits que contiene el recuento de las entradas de *sample offset table* a seguir.

La Figura A.8 muestra la estructura del átomo *stco* de un vídeo MP4:

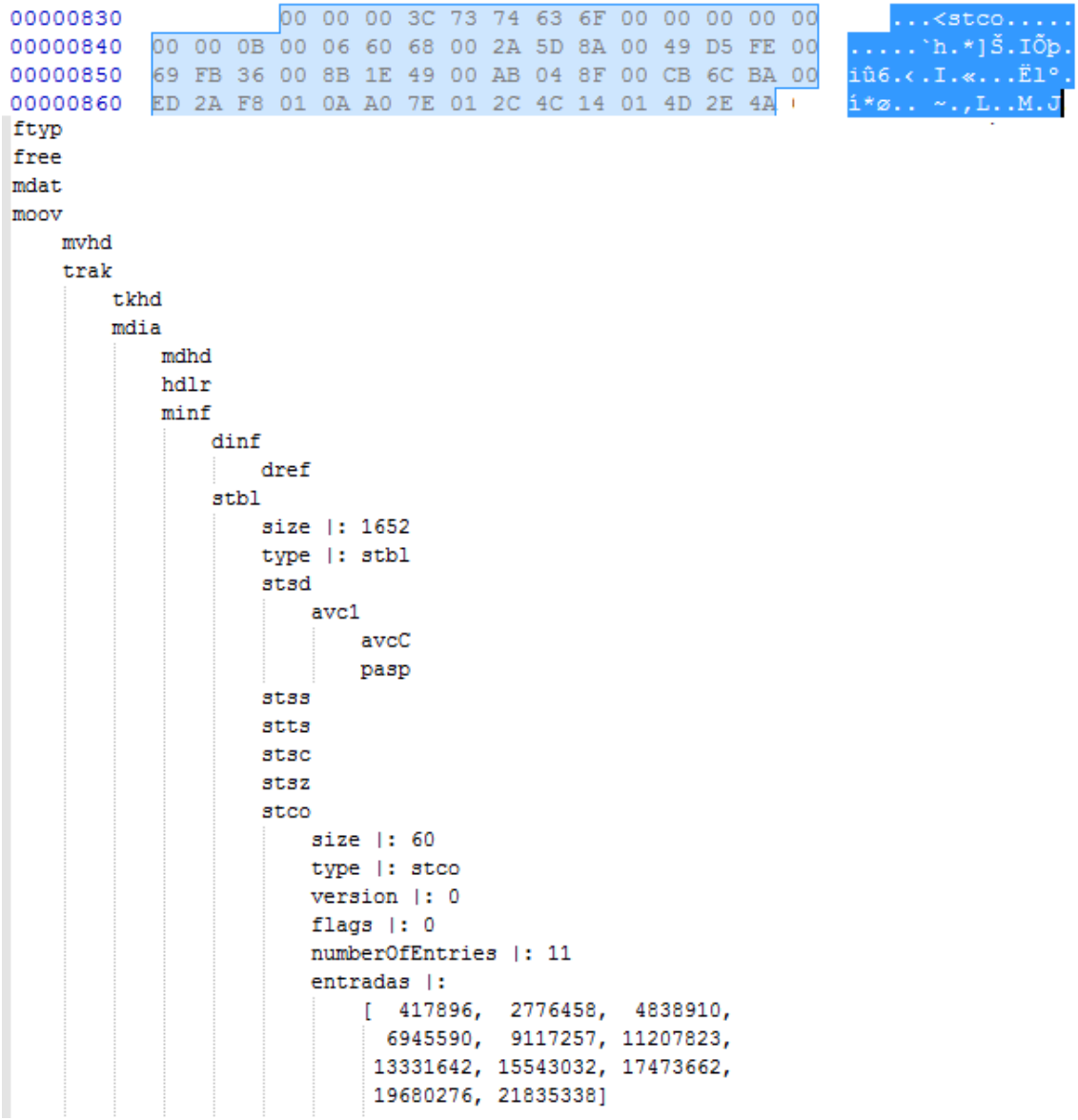


Figura A.8: Estructura del átomo *stco*

A.20. Átomo User Data - udta

Este átomo permite definir y almacenar datos asociados a un vídeo. Si este átomo tiene como padre al átomo *movie* contiene datos relevantes para la película en su conjunto, si es un átomo *track* sólo contiene datos específicos para esa pista. Contiene los campos que se indican en la Tabla A.28:

Tabla A.28: Estructura del átomo: *udta*

Size	Type	User Data List
	0x75647461 (udta)	
32 bits	32 bits	Lista de átomos

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este átomo *user data*.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *udta* (0x75647461).
- **User Data List**: Un lista de datos de usuario que tiene formato de una serie de átomos. Cada elemento de datos en la lista de datos de usuario contiene tamaño y el tipo de información junto con sus datos. Por razones históricas, la lista de datos se termina opcionalmente por un conjunto entero de 32 bits a "0". Si está escribiendo un programa para leer átomos de datos de usuario, deben permitir la terminación en "0". Sin embargo, si usted está escribiendo un programa para crear átomos de datos de usuario, puede dejar con seguridad fuera del campo el "0".

En la Tabla A.29 se observa los tipos de entradas posibles de la lista de datos de usuario.

Tabla A.29: Tipos de entradas de la lista de datos de usuario y su descripción

Tipo de las entrada	Descripción
'©com'	Nombre del compositor
'©cpy'	Copyright
'©day'	Fecha de creación de los contenidos de la película
'©dir'	Nombre del director de la película
'©ed1' to '©ed9'	Fechas editadas y descripciones
'©fmt'	Formato de la película (generado por ordenador, digitalizado…)
'©inf'	Información sobre la película
'©isr'	Código ISRC
'©lab'	Nombre del sello discográfico
'©lal'	URL del sello discográfico
'©mak'	Nombre del creador o marca
'©mal'	URL del creador o marca
'©nak'	Título de las palabras clave del contenido X
'©nam'	Título del contenido
'©pdk'	Palabras clave para el productor X
'©phg'	Declaración del copyright, normalmente precedido por el símbolo P
'©prd'	Nombre del productor
'©prf'	Nombre de los artistas
'©prk'	Palabra clave del artista principal o artista X
'©prl'	URL del artista principal o artista X
'©req'	Requisitos especiales de hardware y software
'©snk'	Palabras clave del subtítulo del contenido X
'©snm'	Subtítulo del contenido
'©src'	Créditos
'©swf'	Nombre del compositor musical
'©swk'	Palabras clave del compositor musical X
'©swr'	Nombre y versión del software (o hardware) que generó la película
'©wrt'	Nombre del escritor de la película
'©xyz'	Coordenadas GPS del vídeo
'AllF'	Play All Frames-Indica que todos los frames deben ser mostrados, independientemente del tiempo
'hinf'	Hint Track Information- Estadísticas de datos para la reproducción de una pista en particular
'hnti'	Hint Info Atom-Datos usados para la reproducción de la película o pista
'name'	Nombre del objeto
'tnam'	Localiza el nombre de la pista opcionalmente presente en los datos de usuario. La carga útil se describe en el nombre de la pista.
'tagc'	Características del medio opcionalmente presente en los datos de usuario. Texto especial que describe datos de interés para la pista
'LOOP'	Un long integer que indica el estilo del bucle. Éste átomo no está presente a menos que la película esté en un bucle. 0 para el bucle normal y 1 para el bucle palindrómico
'ptv '	Print to vídeo—Película a pantalla completa. Contiene 16 bytes de estructura
'WLOC'	Localización por defecto de la ventana para la película. Dos valores de 16 bits

La Figura A.9 muestra la estructura del átomo *udta* de un vídeo MP4:

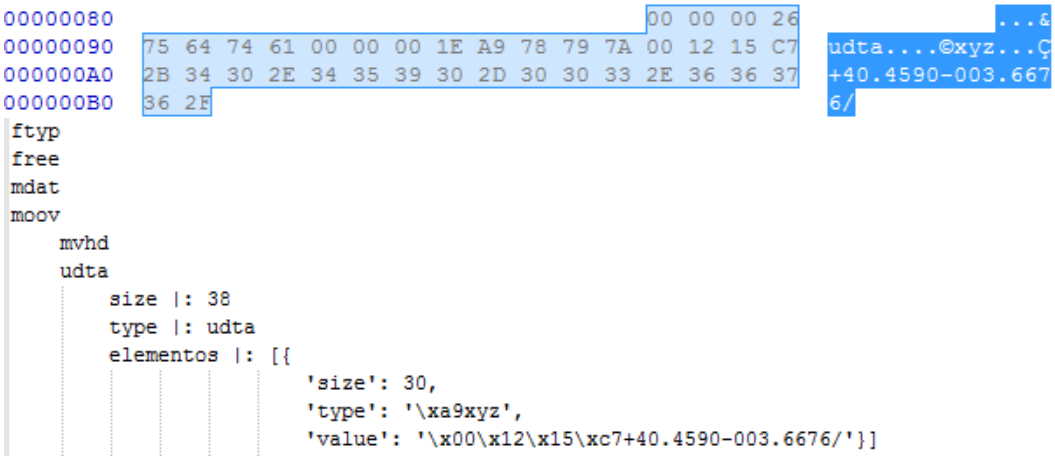


Figura A.9: Estructura del átomos *udta*

A.21. Átomo Metadata - meta

Este átomo contiene metadatos del archivo. Contiene los campos que se muestran en la Tabla A.30.

Tabla A.30: Estructura del átomo: *tkhd*

Size	Type	hdlr	keys	ilst
	0x6D657461 (meta)			
32 bits	32 bits	átomo	átomo	átomo

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este primer átomo *meta data structure*.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *meta* (0x6D657461).

A.22. Átomo Metadata Handler - hdlr

Este átomo es la cabecera del átomo *metadata*. La estructura es igual que al *media data handler*, pero los describimos por separado porque sus campo *component name* tienen tamaños distintos. Como se puede observar en la Tabla A.31 sus

campos son los mismos.

Tabla A.31: Estructura del átomo: *tkhd*

Size	Type	Version	Flags	Predefined	Component Type	Component Subtype	Component Name
	68646C72 (hdlr)	0x00	0	0	mdta		
32 bits	32 bits	1 byte	3 byte	32 bits	32 bits	32 bits	

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este primer átomo *meta data handler*.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *hdlr* (0x6D657461).
- **Version**: Una especificación de 1 byte de la versión de este átomo *meta data handler*; establecer aquí a "0x00".
- **Flags**: Tres bytes que especifican la versión de este átomo *meta data handler*; establecer aquí a "0x000000".
- **Predefined**: Un entero sin signo de 32 bits con valor "0x00000000".
- **Component Type**: Un código de cuatro caracteres que identifica el tipo del manipulador (*handler*). Este campo siempre vale "0x6d647461" (*mdta*).
- **Component Subtype**: Un entero sin signo de 32 bits puestos a "0".
- **Component Name**: Una cadena terminada en cero con caracteres UTF-8 que le da un nombre legible para al tipo de metadatos, para fines de depuración e inspección. La cadena puede estar vacía o de un solo byte puesto a "0".

A.23. Átomo Metadata Keys -keys

Metadata Keys, abreviatura *keys*. Este átomo contiene una lista de las claves de los metadatos que pueden estar presentes en el átomo *metadata*. Esta lista se indexa a partir del valor 1, el 0 está reservado. Contienen los campos de la Tabla A.32.

Tabla A.32: Estructura del átomo: *keys*

Size	Type	Version	Flags	Number of Entries	Entries
	0x6B657973 (keys)				
32 bits	32 bits	1 byte	3 bytes	32 bits	átomos

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este primer átomo *metadata item keys*.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *keys* (0x6B657973).
- **Version**: Un entero de 1 bytes que indica la versión del átomo *keys*.
- **Flags**: Un entero sin signo de 3 bytes, que normalmente valen "0"
- **Number of Entries**: Un entero sin signo de 32 bits, que indica cuántas entradas (*keys*) hay.
- **Entries**: Cada entrada tiene el formato de un átomo, cuyos campos son *size*, *type* y *value*.

A.24. Átomo Metadata List - ilst

Este átomo contiene una lista de los valores de los metadatos actuales que están presentes en el átomo *metadata*. Contiene los campos que se indican en la Tabla A.33:

Tabla A.33: Estructura del átomo: *ilst*

Size	Type	Size Key	Number of Entries	Entries
	0x696C7374 (ilst)			
32 bits	32 bits	32 bits	32 bits	átomos

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes en este primer átomo *metadata item list*.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *ilst* (0x696C7374).
- **Size Key**: Un entero sin signo de 32 bits que indica el tamaño de las *keys*.
- **Number of Entries**: Un entero sin signo de 32 bits que indica el número de entradas que hay en el átomo.

- **Entries**: Cada entrada tiene el formato de un átomo *data* y cada una completa la información de una *key* específica.
- **data**: Este átomo contiene los valores de la lista de claves de metadatos. Su estructura se puede observar en la Tabla A.34.

Tabla A.34: Estructura del átomo: *data*

Size	Type	Key Index	Locale	Value
	0x64617461 (data)			
32 bits	32 bits	32 bits	32bits	32 bits

- **Size**: Un entero sin signo de 32 bits que especifica el número de bytes los átomos *data*.
- **Type**: Un entero sin signo de 32 bits que identifica el tipo, representado como un código de cuatro caracteres; este campo debe estar a *data*.
- **Key Index**: Un entero sin signo de 32 bits que identifica la *key* a la que está complementando.
- **Locale**: Un entero sin signo de 32 bits que identifica, 16bits para indicar el país y 16 bits para indicar el idioma. Si no se especifica este campo vale "0". Los valores permitidos para este campo son del 1 al 255.
- **Value**: Valor de la *key* a la que corresponde esta entrada.

La Figura A.10 muestra el esquema del funcionamiento y la correlación de los átomos *keys* e *ilst*.

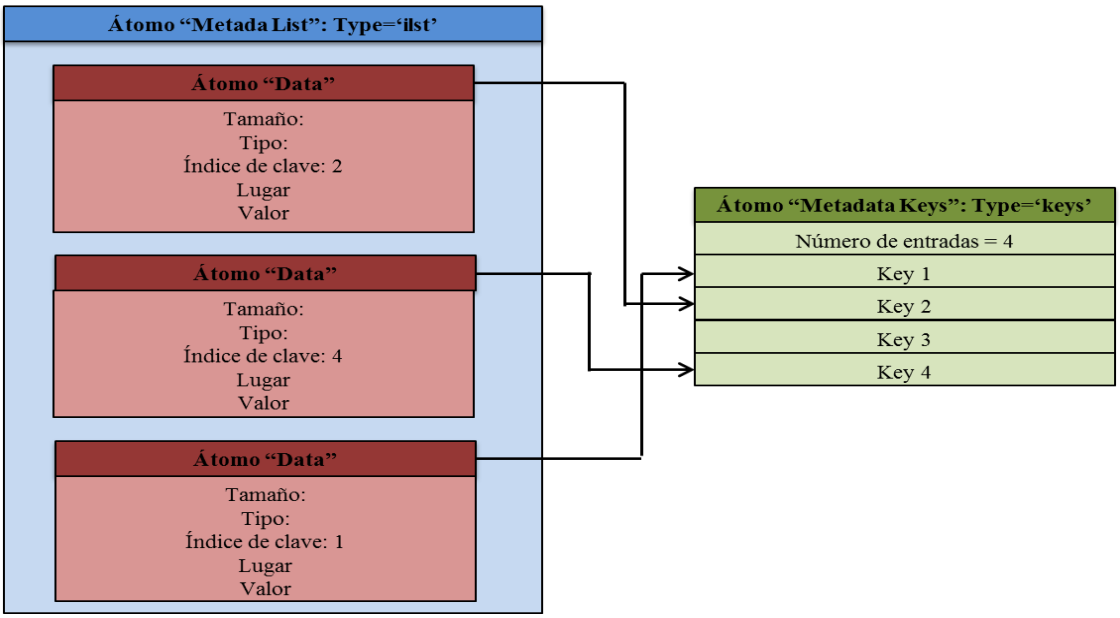


Figura A.10: Correlación entre los átomos *keys* e *ilst*

La estructura del átomo *meta*, *hdlr*, *keys* e *ilst* de un vídeo MP4 se encuentra en la Figura A.11:

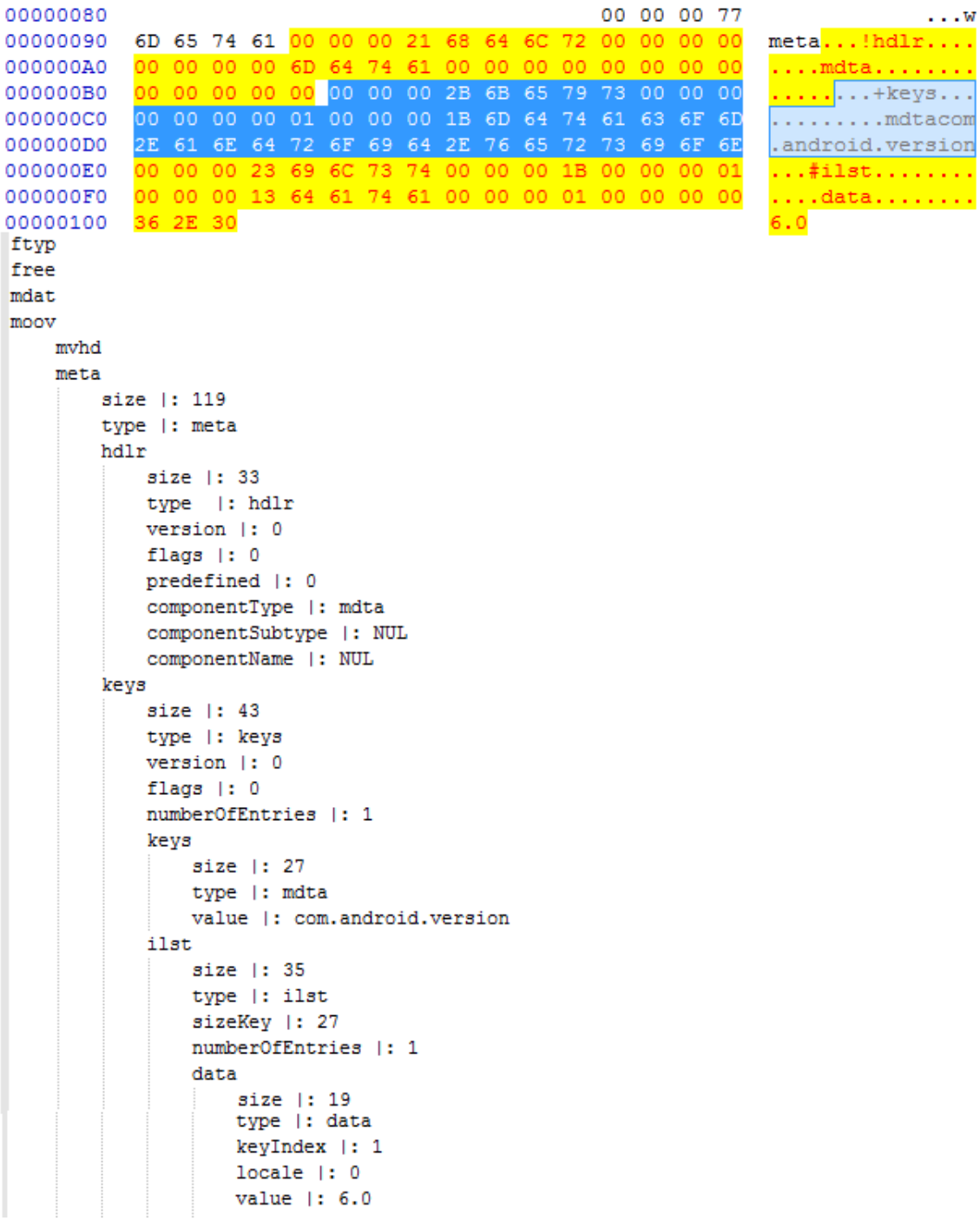


Figura A.11: Estructura de los átomos *meta*, *hdlr*, *keys* e *ilst*

REFERENCIAS

[APS98] J. Adams, K. Parulski, and K. Spaulding. Color Processing in Digital Cameras. *Micro, IEEE*, 18(6):20–30, December 1998.

[AZ06] M. Al-Zarouni. Mobile Handset Forensic Evidence: a Challenge for Law Enforcement. In *Proceedings of the 4th Australian Digital Forensics Conference*. School of Computer and Information Science, Edith Cowan University, December 2006.

[BFM+12] P. Bestagini, M. Fontani, S. Milani, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro. An Overview on Video Forensics. In *APSIPA Transactions on Signal and Information Processing*, volume 1, pages 1229–1233. APSIPA, August 2012.

[BL04] M. Boutell and J. Luo. Photo Classification by Integrating Image Content and Camera Metadata. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 4, pages 901–904. IEEE Computer Society, August 2004.

[BL05] M. Boutell and J. Luo. Beyond Pixels: Exploiting Camera Metadata for Photo Classification. *Pattern Recognition*, 38(6):935–946, June 2005.

[Bov05] A. C. Bovik. *Handbook of Image and Video Processing (Communications, Networking and Multimedia)*. Academic Press, Inc., Orlando, FL, USA, 2005.

[Bra99] Karlheinz Brandenburg. Mp3 and Acc Explained, 1999.

[BSM06] S. Bayram, H. T. Sencar, and N. Memon. Improvements on Source Camera-Model Identification Based on CFA Interpolation. In *Working Group 11.9 International Conference on Digital Forensics*, pages 24–27. Springer, February 2006.

[BSM08a] S. Bayram, H. T. Sencar, and N. Memon. Classification of Digital Camera-Models Based on Demosaicing Artifacts. *The International Journal of Digital Forensics & Incident Response*, 5(2):49–59, September 2008.

[BSM08b] Sevinc Bayram, Husrev Taha Sencar, and Nasir Memon. Video Copy Detection Based on Source Device Characteristics: A Complementary Approach to Content-based Methods. In *Proceedings of the 1st ACM International Conference on Multimedia Information Retrieval*, pages 435–442, New York, NY, USA, October 2008.

[CAS+06] O. Celiktutan, I. Avcibas, B. Sankur, N. P. Ayerden, and C. Capar. Source Cell-Phone Identification. In *Proceedings of the IEEE 14th Signal Processing and Communications Applications*, pages 1–3. IEEE, April 2006.

[CESR12] F. D. O. Costa, M. Eckmann, W. J. Scheirer, and A. Rocha. Open Set Source Camera Attribution. In *Proceedings of the 25th Conference on Graphics, Patterns and Images*, pages 71–78. IEEE, August 2012.

[CFGL08] M. Chen, J. Fridrich, M. Goljan, and J. Lukas. Determining Image Origin and Integrity Using Sensor Noise. *IEEE Transactions on Information Forensics and Security*, 3(1):74–90, March 2008.

[CIS16] CISCO. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020, February 2016.

[COF12] V. Conotter, J. F. Obrien, and H. Farid. Exposing Digital Forgeries in Ballistic Motion. In *IEEE Transactions Information Forensics and Security*, pages 283–295, California, Berkeley, USA, February 2012.

[Cor12] Panasonic Corporation. Lumix Digital Camera Know-Hows, 2012.

[CSA08] O. Celiktutan, B. Sankur, and I. Avcibas. Blind Identification of Source Cell-Phone Model. *IEEE Transactions on Information Forensics and Security*, 3(3):553–566, September 2008.

[CSS07] W. Chen, Y.Q. Shi, and W. Su. Image Splicing Detection Using 2-D Phase Congruency and Statistical Moments of Characteristic Function. In *Proceedings on Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, pages 65050R–65050R–8. SPIE-International Society for Optical Engine, January 2007.

[fS03a] The International Organization for Standardization. Information Technology Coding of Audio Visual Objects Part 14 MP4 File Format, 2003.

[fS03b] The International Organization for Standardization. Information Technology Coding of Audio Visual Objects Part 3 Audio, 2003.

[GBK+01] Z. J. Geradts, J. Bijhold, M. Kieft, K. Kurosawa, K. Kuroki, and N. Saitoh. Methods for Identification of Images Acquired with Digital Cameras. In *Proceedings on Enabling Technologies for Law Enforcement and Security*, volume 4232, pages 505–512. SPIE-International Society for Optical Engine, February 2001.

[GFC11] M. Goljan, J. Fridrich, and Mo Chen. Defending Against Fingerprint-Copy Attack in Sensor-Based Camera Identification. *IEEE Transactions on Information Forensics and Security*, 6(1):227–236, March 2011.

[Gil07] Paul L. Gilman. Digital Audio Encoding, 2007.

[GKWB07] T. Gloe, M. Kirchner, A. Winkler, and R. Bohme. Can We Trust Digital Image Forensics? In *Proceedings of the 15th International Conference on Multimedia*, pages 78–86. ACM Press, September 2007.

[GMT10] Valenzise G., M. Magni, S.and Tagliasacchi, and S. Tubaro. Estimating Channel Induced Distortion in H.264/AVC Video Without Bitstream Information. In *Workshop on Quality of Multimedia Experience QoME)*, pages 100–105, June 2010.

[HHLH08] C. C. Hsu, T. Y. Hung, C. W. Lin, and C.T. Hsu. Video Forgery Detection Using Correlation of Noise Residue. In *IEEE 10th Workshop on Multimedia Signal Processing*, pages 170–174, October 2008.

[HPN07] Barry G. Haskell, Atul Puri, and Arun N. Netravali. *Digital Video: An Introduction to MPEG-2 (Digital Multimedia Standards)*. Springer US, Orlando, FL, USA, 2007.

[HS09] Li. Huiying and Forchhammer. Soren. MPEG2 Video Parameter and No Reference PSNR Estimation. In *Picture Coding Symposium*, pages 1–4, Chicago, IL, May 2009.

[Iai10] E. R. Iain. *The h.264 Advanced Video Compression Standard*. Wiley, London, UK, 2010.

[Inc12] Texas Instruments Incorporated. Digital Still Camera, 2012.

[KS16] Jasdeep Kaur and Nipun Sharma. Survey on the General Concepts of MPEG Moving Picture Experts Group. *Paripex*, 5(2):252–255, February 2016.

[KTY10] M. Kobayashi, O. Takahiro, and S. Yoichi. Detecting Forgery From Static Scene Video Based on Inconsistency in Noise Level Functions. In *IEEE Transactions Information Forensics and Security*, pages 883–891, December 2010.

[LF03] J. Lukas and J. Fridrich. Estimation of Primary Quantization Matrix in Double Compressed JPEG Images. In *Digital Forensic Research Workshop*, pages 5–8, August 2003.

[LFG06] J. Lukas, J. Fridrich, and M. Goljan. Digital Camera Identification from Sensor Pattern Noise. *IEEE Transactions on Information Forensics and Security*, 1(2):205–214, June 2006.

[LH06] Y. Long and Y. Huang. Image Based Source Camera Identification using Demosaicking. In *Proceedings of the IEEE 8th Workshop on Multimedia Signal Processing*, pages 419–424. IEEE, October 2006.

[Li10] C.-T. Li. Source Camera Identification Using Enhanced Sensor Pattern Noise. *IEEE Transactions on Information Forensics and Security*, 5(2):280–287, June 2010.

[Liu15] C. Liu. Worldwide Internet and Mobile: Users EMarketer's Updated Estimates for 2015, 2015.

[LKL+08] Min-Jeong Lee, Kyung-Su Kim, Hae-Yeoun Lee, Tae-Woo Oh, Young-Ho Suh, and Heung-Kyu Lee. Robust Watermark Detection Against D-A/A-D Conversion for Digital Cinema Using Local Auto-correlation Function. In *ICIP*, pages 425–428, San Diego, USA, October 2008.

[LKL12] Min-Jeong Lee, Kyung-Su Kim, and Heung-Kyu Lee. Digital Cinema Watermarking for Estimating the Position of the Pirate. In *IEEE TRANSACTIONS ON MULTIMEDIA*, volume 10, pages 605–621, November 2012.

[LWH08] Weiqi Luo, Min Wu, and Jiwu Huang. Mpeg recompression detection based on block artifacts. In *Proc. SPIE*, pages 1–12, February 2008.

[M.11] Jokay M. The Design of a Steganographic System Based on the Internal MP4 File Structures. In *International Journal of Computers and Comunications*, Bratislava, Slovakia, June 2011.

[McK07] C. McKay. Forensic Analysis of Digital Imaging Devices. Technical report, University of Maryland, 2007.

[MCP+07] N. Mondaini, R. Caldelli, A. Piva, M. Barni, and V. Cappellini. Detection of Malevolent Changes in Digital Video for Forensic Applications. In *Proceedings of SPIE*, pages 1–12, February 2007.

[Moe12] T. B. Moeslund. *Introduction to Video and Image Processing, Building Real Systems and Applications*. Springer-Verlag, London, August 2012.

[MT15] A. Murat Tekalp. *Digital Video Processing*. Prentice Hall, Massachusetts, USA, June 2015.

[Nak05] J. Nakamura. *Image Sensors and Signal Processing for Digital Still Cameras*. CRC Press, Boca Raton, FL, USA, August 2005.

[Org04] National Information Standards Organization. Understanding Metadata. http://-www.niso.org/publications/press/UnderstandingMetadata.pdf, 2004.

[OVG+15] A. L. Sandoval Orozco, L. J. García Villalba, D. M. Arenas González, J. R. Corripio, J. Hernandez-Castro, and S. J. Gibson. Smartphone image acquisition forensics using sensor fingerprint. *IET Computer Vision*, 9(5):723–731, September 2015.

[PF05] A.C. Popescu and H. Farid. Exposing Digital Forgeries by Detecting Traces of Resampling. *IEEE Transactions on Signal Processing*, 53(2):758–767, February 2005.

[PO11] F. Pescador O. Contribución a las metodologías de optimización del tiempo de ejecución de algoritmos de descodificación de vídeo . Tesis doctoral, Universitat Politecnica de Madrid, Julio 2011.

[PPML15] C. S. Po, L. S. Pey, K. C. Min, and Jie L. Forensic and Anti Forensic Techniques for Video Shot Editing in H.264 AVC. In *CrossMark*, February 2015.

[RC13] J. Rosales Corripio. Algoritmo de Identificación de Fuente en Imágenes Digitales de Dispositivos Móviles. Tesis de máster, Universidad Complutense de Madrid, Julio 2013.

[RCAGSO+13] J. Rosales Corripio, D. M. Arenas González, A. L. Sandoval Orozco, L. J. García Villalba, J. C. Hernandez-Castro, and S. J. Gibson. Source Smartphone Identification Using Sensor Pattern Noise and Wavelet Transform. In *Proceedings of the 5th International Conference on Imaging for Crime Detection and Prevention*, pages 1–6, London, UK, December 2013.

[RCC+08] N. L. Romero, V. G. Chornet, J. S. Cobos, A. S. Carot, F. C. Centellas, and M. C. Mendez. Recovery of Descriptive Information in Images From Digital Libraries by Means of EXIF Metadata. *Library Hi Tech*, 26(2):302–315, 2008.

[RCWA15] M. Ramesh Chourasiya and P. Wadhe A. Video Anti Forensics - A Review. *IJRITCC*, 3(4):2110–2114, April 2015.

[RP07] Reibman R. and D. Poole. Characterizing Packet Loss Impairments in Compressed Video. In *IEEE International Conference on Image Processing*, pages 77–80, San Antonio, TX, October 2007.

[SLFK10] M. Steinebach, H. Liu, P. Fan, and S. Katzenbeisser. Cell Phone Camera Ballistics: Attacks and Countermeasures. In *Proceedings on Multimedia on Mobile Devices*, pages 75420B–75420B–9. SPIE-International Society for Optical Engine, January 2010.

[SMW07] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, September 2007.

[SOAGRC+13] A.L. Sandoval Orozco, D.M. Arenas González, J. Rosales Corripio, L.J. García Villalba, and J.C. Hernandez-Castro. Techniques for Source Camera Identification. In *Proceedings of the 6th International Conference on Information Technology*, pages 1–9, May 2013.

[Spe16] QuickTime File Format Specification. QuickTime File Format Specification, 2016.

[SRL11] M. C. Stamm and K. J. Ray Liu. Anti-Forensics For Frame Deletion Addition In MPEG Video. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1876–1879, Maryland, USA, May 2011.

[SXD09] Y. Su, J. Xu, and B. Dong. A Source Video Identification Algorithm Based on Motion Vectors. In *Proceedings of the Second International Workshop on Computer Science and Engineering*, volume 2, pages 312–316, Qingdao, China, October 2009.

[SZC+03] Tengku Mohd. Tengku Sembok, Halimah Badioze Zaman, Hsinchun Chen, Shalini R. Urs, and Sung Hyon Myaeng. Technology and Management of Indigenous Knowledge for Global Access. In *Proceedings of the 6th International Conference on Asian Digital Libraries*, pages 346–350, Kuala Lumpur, Malaysia, December 2003.

[Tes05] J. Tesic. Metadata Practices for Consumer Photos. *IEEE Multimedia*, 12(3):86–92, September 2005.

[THN+] M. Tennoe, E. Helgedagsrud, M. Naess, H. Kjus Alstad, H. Kvale Stensland, V. Gaddam, Reddy Gaddam, D. Johansen, C. Griwodz, and booktitle = Proceedings of the 2013 IEEE International Symposium on Multimedia title = Efficient Implementation and Processing of a Real-Time Panorama Video Pipeline year = 2013 month=December pages = 76–83 address = Washington, DC, USA Halvorsen, P.

[TLL07] M. J. Tsai, C. L. Lai, and J. Liu. Camera/Mobile Phone Source Identification for Digital Forensics. In *Proceedings of the International Conference on Acoustics Speech and Signal Processing*, pages II–221–II–224. IEEE, April 2007.

[VCEK07] T. Van Lanh, K. S. Chong, S. Emmanuel, and M. S. Kankanhalli. A Survey on Digital Camera Image Forensic Methods. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 16–19. IEEE, July 2007.

[VOC15] L. J.García Villalba, A. L. Sandoval Orozco, and J. Rosales Corripio. Smartphone Image Clustering. *Expert Systems with Applications*, 42(4):1927–1940, March 2015.

[WBSH09] A. W. A. Wahab, J.A. Briffa, H. G. Schaathun, and A. T. S. Ho. Conditional Probability Based Steganalysis for JPEG Steganography. In *Proceedings of the International Conference on Signal Processing Systems*, pages 205–209, Singapore, May 2009.

[WF06] W. Wang and H. Farid. Exposing Digital Forgeries in Video by Detecting Double MPEG Compression. In *Proceedings of the 8th Workshop on Multimedia and Security*, pages 37–47, New York, NY, USA, September 2006.

[WF07] W. Wang and H. Farid. Exposing Digital Forgeries in Video by Detecting Duplication. In *Proceedings of the 9th Workshop on Multimedia and Security*, pages 35–42, New York, NY, USA, September 2007.

[WF08] Weihong Wang and Hany Farid. Detecting Re-projected Video. In *Information Hiding*, pages 72–86, Berlin, Heidelberg, October 2008.

[WHL12] A. W. A. Wahab, A. T. S. Ho, and S. Li. Inter-Camera Model Image Source Identification with Conditional Probability Features. In *Proceedings of the IIEEJ 3rd Image Electronics and Visual Computing Workshop*, Kuching, Malaysia, November 2012.

[Woo05] C. Wootton. *A Practical Guide to Video and Audio Compression*. Elsevier, Burlington, USA, January 2005.

[YHW12] S. Yahaya, A. T. S. Ho, and A. A. Wahab. Advanced Video Camera Identification Using Conditional Probability Features. In *Proceedings of the IET Conference on Image Processing*, pages 1–5, London, UK, July 2012.