



UNIVERSITÀ
DEGLI STUDI
FIRENZE

SCUOLA DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

FORENSIC ANALYSIS OF VIDEO FILE CONTAINERS

Candidato

Saverio Meucci

Relatori

Prof. Alessandro Piva

Prof. Fabrizio Argenti

Correlatori

Dott. Massimo Iuliani

Dott. Marco Fontani

ANNO ACCADEMICO 2015/2016

To myself.

Contents

Contents	ii
Abstract	1
Introduction	2
1 State of the Art	6
1.1 Introduction to Multimedia Forensics	6
1.1.1 Applications	9
1.1.2 Tools	9
1.2 Forensic Analysis of Video File Container	10
1.2.1 Video File Containers	10
1.2.2 Applications for Video File Container	13
2 Video File Containers	14
3 Proposed Approach	15
3.1 Overview	15
3.2 Video Format Tool	16
3.2.1 Requisiti Funzionali	16
3.2.2 Implementazione	17
3.2.3 Command Line Tool	20
3.3 File Origin Analysis Tool	21
3.3.1 Requisiti Funzionali	21
3.3.2 Implementation	21
3.3.3 Command Line Tool	24

CONTENTS	iii
3.4 Web Application	24
3.4.1 Features	25
4 Experiments and results	28
4.1 Integrity Analysis based on File Container	28
4.2 Model Identification based on File Container	28
4.3 Application on Social Network	28
Conclusions	29
Bibliography	30
Credit	31

Abstract

Introduction

In an increasingly digital world, there are more and more applications where digital contents play an important role.

Thanks to the spread of smartphones and digital cameras, the use of social networks that allow the sharing of digital images and videos is becoming more widespread. These resources, in general, contain information of personal nature; however, since these images and videos are more and more permeating our lives, their content may include information about an event that has occurred and therefore represents a significant source of evidence about a crime that can be used during an investigation.

In this context, many techniques for the analysis of multimedia content have been developed; these techniques pose the goal of providing aid in making decisions about a crime so that a digital resource can be legitimately used as evidence in a courtroom.

Such is the task of the Forensic Analysis and in particular of Multimedia Forensics, i.e. to develop and apply techniques that allow, with a certain degree of accuracy, to determine whether the content of a digital resource is authentic or if it has been manipulated.

The questions that the Forensic Analyst must answer about these digital contents are:

- authenticity, or whether what they represent is true and corresponds to reality.
- integrity, or if they have been altered in such a way that does not compromise their authenticity.

- the determination of the acquisition device, a problem that in Forensic Analysis takes the name of Source Classification or Source Identification.

To give a better understanding of these problems, we provide some application scenarios.

Regarding the Source Identification, imagine a situation in which the very act of creation of an image or video constitutes a crime, due to the content of the digital resource. In this scenario, it is crucial to establish the source device that generated the offending resource, so that this resource can be used as an evidence. Depending on the context, determine the source of a digital content could mean to find the specific acquisition device or the typology of the acquisition device, or establish the last processing stage, such as compression algorithm.

In this context, it is also important an integrity check of a digital resource. In fact, the owner of the device that created the offending digital content could change the resource in such a way that it's not possible to trace the source device.

A more particular scenario, regarding the authenticity of a multimedia resource, is when the digital content is changed so as to deceive those who are watching it. The reasons for this manipulation can be different, such as the exaggeration or limitation of the severity of an accident or disaster, as well to change the context of the situation represented.

The tools available to the Multimedia Forensics are only those that can be extracted from the digital content itself. In fact, the fundamental idea of forensic analysis of digital content is based on the observation that both the acquisition process and the post-processing step leave traces, called digital fingerprints. It is the task of the Forensic Analyst to analyze these fingerprints to determine the history and the authenticity of a digital content so that it can be used as evidence in a digital investigation.

In this regard, the analysis techniques of digital content mainly focused on

two approaches: the analysis of the data stream, i.e. the audio-visual signal, which is based on the research of artefacts and inconsistency in the digital content; the analysis of the metadata, i.e. the determination of their compatibility, completeness and consistency with regards to the context in which it is assumed the resource has been created.

The presented issues have been extensively treated by Digital Image Forensics; instead, regarding the Digital Video Forensics, the research for these problems is still under study and improvement. The motivation for this discrepancy lies in much easier way that an image can be falsify than a digital video and the numerous set of video formats (MPEG, MPEG2, H26x, VP8, etc) in constrast to the few main formats for digital images (mostly JPEG, PNG, TIFF).

This thesis proposes a technique for forensic analysis of digital video based on *Gloe et al.* [1], whose work focuses on the study of some video formats standard and on the use of the internal structure of the video file container as a tool for the forensic analysis. This choice is justified by the fact that the video file container is very fragile so that it can contain a lot of information about the history of a digital video; in fact, both the acquisition phase and the subsequent post-processing steps or the editing of the file using various tools, modify the content and the structure of the container. This can be exploited both in terms of Source Identification and of integrity verification.

The thesis is divided in chapters as follows. Chapter 1 provides an introduction to Multimedia Forensics, discussing both the state of the art techniques and the standards to be followed in all aspects of a digital investigation; it also discussed in greater detail the work done by *Gloe et al.* [1], giving more information on video file containers. Chapter 2 will explain in great detail the structure of the video file containers for MP4-like formats. In Chapter 3, it is presented the proposed approach along with all the choices made and the details of the implemented tool. In Chapter 4, it is presented the dataset used, the motivation and the organization of the experiments along with a discussion of the results

obtained.

Chapter 1

State of the Art

1.1 Introduction to Multimedia Forensics

With the increasing spread of digital audio and video content, the analysis of these multimedia objects is rapidly assuming importance in the context of digital investigations, which consider both digital data and digital devices.

In digital investigations, multimedia content such as images, audio, and video are more and more being used as forensic evidence. It is therefore very important to be able to extract information from such content in a reliable manner.

Multimedia Forensics has the aim to gain knowledge on a multimedia content life cycle exploiting the traces that the various processing steps leave on the data. In fact, the idea behind Multimedia Forensics is that each acquisition device and each processing operation on a digital resource leave on the media content data some traces, often called fingerprints, which characterize its history.

Many algorithms and techniques have been developed by the scientific community based on the extraction of features from the stream of audio-visual data of the multimedia content. By taking advantage of these features, such techniques try to infer information about the source/acquisition device and about which encoding and editing processes the digital resource was subject to during its life cycle. Specifically for multimedia content such as images and videos, the main approaches set themselves the goal to identify the source of digital resource and/or to determine if the content is authentic or has been modified from

its original without any a priori information about the content under analysis. This is possible using just these features and tools that allow to check for the presence or absence of such features or fingerprints that are intrinsically linked to the multimedia data by the acquisition device, the encoding step and any post-processing or editing software tools. In fact, we can distinguish three types of traces left on a multimedia content: acquisition traces, encoding traces, and editing traces.

As explained, from a scientific point of view, research has produced a large number of techniques for the analysis of multimedia content. From the point of view of the application of such techniques in the courtroom, however, there is an important gap that it is probably due to a lack of communication between the legal part and the scientific part, as well as a not complete maturity of the techniques that are often based on results obtained in laboratory contexts a not in real-world scenarios.

In addition, there is the need for greater sharing of standard in the field of digital multimedia forensics investigations that aid Multimedia Forensics to grow and reach maturity.

Several communities and groups have worked to put together guidelines and standard on important aspects of digital investigations, such as the chain of custody, data authentication, application of the scientific method, documentation, and reporting. The ISO/IEC JTC1 Working Group 4 is one of those groups that seek to give international standards LISTA whose main purpose is to promote the best procedures and methods for the investigation of digital evidence. It also encourage the adoption of approaches for the forensic analysis on multimedia content that are shared at an international level, in order to ease the comparison and the combination of results from different entities and organizations and also through various jurisdictions, so as to increase the reliability of such methods and of the results.

Another group that aims at giving standards and guidelines for the digital

investigations is the Scientific Working Group on Digital Evidence (SWGE) that deals with get in contact different organizations that work in the field of Multimedia Forensics in order to promote communication and cooperation and to ensure higher quality and consistency within the forensic community.

The Scientific Working Group on Imaging Technologies (SWGIT), instead, focuses his work on image analysis technology and has the aim to facilitate the integration of such methods of analysis of images in the context of the judicial system by providing best practices and guidelines for the acquisition, storage, processing, analysis, transmission, output image and archive of digital evidence.

Regarding the forensic analysis of images and/or videos, the process is defined to be composed of three main tasks: technical preparation, examination, interpretation. The technical preparation is concerned with all those operations necessary to prepare videos and images to the other tasks. The examination is the main part of the forensic analysis and deals with the application of techniques that aim to extract information from images/videos. The interpretation concerns the analysis of digital content from experts in order to provide conclusions on the features extracted from the images/videos under examination.

In this context, it becomes essential the figure of the Forensic Analyst, i.e. one who is able to develop and apply these methods for the analysis of digital content, interpret the results and make a summary of the results from different techniques in order to increase the reliability of the conclusions. It is also able to perform all of the analysis tasks in accordance with the standards shared by the forensic community. The Forensic Analyst is able to find the traces left on the multimedia data and to acquire information on the object under examination such as who is the source/acquisition device, whether the content is authentic and if the resource is intact, and so on.

1.1.1 Applications

Le principali applicazioni per l'analisi forense sono: la source identification, l'autenticazione e l'integrità della risorsa multimediale.

Il processo di identificazione della sorgente ha come obiettivo quello di recuperare informazioni riguardo il dispositivo sorgente del contenuto multimediali in esame. È possibile identificare la sorgente a vari livelli di dettaglio. Ad esempio distinguere fra tipi di sorgenti, distinguere fra differenti modelli di sorgente dello stesso tipo e distinguere fra dispositivi diverse ma dello stesso tipo e modello. Informazioni possono essere ottenuti dai metadati.

Il problema dell'autenticazione riguarda il compito di stabilire se il contenuto multimediale è un accurato rendiconto di un evento originale. Il processo di analisi si basa sulla ricerca di inconsistenze nelle features del segnale audiovisivo.

Il problema dell'integrità riguarda il compito di stabilire se un contenuto multimediale è stato modificato o meno da quando è stato acquisito dal dispositivo sorgente. L'analisi si basa sulla ricerca di tracce lasciate dai tools di editing e processing durante il ciclo di vita che non sono compatibili con i contenuti provenienti dal dispositivo sorgente che è noto.

1.1.2 Tools

Un file multimediale può essere visto come un pacco composto da due parti principali: l'header, che contiene i metadati ovvero informazioni sul contenuto del file; il contenuto stesso, ovvero il flusso di dati che forma il segnale audiovisivo. In generale, l'estrazione di features si basa sull'analisi delle tracce lasciate sia sui dati veri e proprio che sui metadati del file multimediale.

Per quanto riguarda l'ispezione del flusso di dati, l'esaminazione consiste di due aspetti principali: l'interpretazione del contenuto, che riguarda l'analisi del contesto come comprendere cosa sta succedendo, chi sono i soggetti e gli

oggetti coinvolti, informazioni sull'ambiente e tutte le informazioni che derivano dall'osservazione umana; identificazione di dettagli rivelanti nella scena rappresentata, come anomalie audio-visive, features interessanti come direzione della luce, ombre, prospettiva incoerenze, segni di smudge ecc. Sempre nel contesto dell'ispezione del segnale audio-visivo, è l'analisi e l'enhancement dei contenuti, come migliorare il segnale per rilevare dettagli o oggetti rilevanti, estrazione di relazioni dimensionali come dimensioni di oggetti o soggetti, comparazione fotografica fra oggetti noti e oggetti rappresentati nella scena.

L'altro strumento riguarda l'estrazione e l'analisi di metadati. I metadati possono essere estratti facilmente e possono contenere molte informazioni riguardo il segnale audio-visivo come ad esempio (video) il dispositivo sorgente, lo spazio colore, la risoluzione, i parametri di compressione, la data, dati gps, frame rate, (audio) format tags, bit rate, sample rate, numero di canali. Ovviamente il tipo e il numero di metadati dipende dal tipo del file in esame e quali processi ha subito durante il suo ciclo di vita. Ad esempio, per quando riguarda le immagini i metadati sono estratti dall'header Image file format (Exif). Una volta ottenuti i metadati è compito dell'analista forense verificare la compatibilità, la completezza e la coerenza delle informazioni estratte rispetto allo scenario da cui il contenuto si suppone provenire: metadati saranno differenti se una risorsa proviene da un social network piuttosto che direttamente dal dispositivo di acquisizione.

1.2 Forensic Analysis of Video File Container

1.2.1 *Video File Containers*

Gli standard dei formati video prescrivono un numero limitato di features, lasciando quindi molta libertà alla casa produttrice. Ciò può essere sfruttato come risorsa per identificare e autenticare, dato un video, la sorgente di provenienza. In particolare lo studio si concentra su MP4-like video format (mp4 e mov).

In un mondo sempre più digitale c'è sempre più la necessità di metodi per l'autenticazione dei dati multimediali, del riconoscimento della loro sorgente di provenienza e della loro storia.

L'analisi forense si è principalmente concentrata su l'analisi di immagini sviluppando tecniche basate su due principali approcci: analisi del rumore (PRNU); analisi dei metadati JPEG e EXIF.

Anche l'analisi forense di video digitale è sempre più rilevante. I primi lavori hanno seguito un procedimento simile a quello delle immagini, basandosi rumore e artefatti presenti nel video ed anche l'utilizzo di metadati JPEG e EXIF. Tuttavia, riguardo all'utilizzo di metadati per l'autenticazione di sorgenti digitali, sono state espresse preoccupazioni rispetto falsificabilità. Infatti, è solo questione di trovare il giusto strumento, spesso disponibile pubblicamente, per modificare facilmente informazioni di alto livello come appunto i metadati EXIF.

Contrariamente, gli strumenti noti non permettono di modificare strutture core dei file multimediali, come appunto i video file containers, ed altri strumenti pubblicamente disponibili che permettano di falsificare sistematicamente i containers non sono noti. Quindi queste caratteristiche di basso livello riducono le preoccupazioni riguardo la falsificabilità di tali metadati in quanto solo in possesso di skills avanzate di programmazione sarebbe possibile modificare la struttura interna dei file video rendendo la falsificabilità un'operazione non triviale ed altamente tecnica.

Il formato container MOV è stato introdotto da Apple nel 1991. Utilizzando questo formato come base, sono stati introdotti anche i formati MP4 e 3GP.

I file containers di questo formato sono composti da atomi (o boxes) che sono identificati caratteri unici di 4 byte, preceduti dalla dimensione dell'atomo. Tali atomi possono avere o meno dei campi e possono essere o meno nidificati, ovvero possono contenere altri atomi; quindi si incontrano 4 tipi di atomi: - atomi senza campi né che contengono altri atomi; - atomi senza campi che contengono altri

atomi; - atomi con campi ma che non contengono altri atomi; - atomi con campi che contengono altri atomi;

La struttura più comune dei MP4-like containers è la seguente: - atomo **ftyp**: è semi-obbligatorio, ovvero il più recente standard ISO prevede che sia presente e che venga esplicitato il prima possibile nel file container. Fa riferimento alla specifica del tipo di file con la quale il file video, a cui il container appartiene, è compatibile. I campi principali sono **majorbrand**, che indica quale specifica sia la migliore, e vari campi numerati chiamati **compatiblebrand**, che indica una specifica che è compatibile con il file video. Non contiene altri atomi. - atomo **mdat**: contiene lo stream dei dati ed è accompagnato dal campo **size** che ne indica la dimensione. Non contiene altri atomi. - atomo **moov**: è l'atomo con la struttura più complessa del container. È un atomo significati, ovvero contiene numerosi altri atomi. In questo atomo, e nella sua sotto-struttura, sono contenuti i metadati necessari per la decodifica del flusso dati contenuto nell'atomo **mdat**.

Tale struttura non è sempre rispettata. I file container dei formati MP4-like sono molto complessi e presentano numerosi elementi e differenze dipendenti dalla sorgente e dalla casa manifatturiera; quindi costituiscono uno strumento prezioso per un'analista forense. Le differenze dei containers di diversi file video si possono presentare nei seguenti modi: - posizione relativi degli atomi: nonostante lo standard dia specifiche riguardo la posizione di determinati atomi (in particolare quelli principali), ciò non è sempre rispettato. Soprattutto, si ha variazione di posizione per quegli atomi la cui posizione non è specificata dallo standard. Questo discorso è valido sia al primo livello, ovvero quello contenente gli atomi **ftyp**, **mdat**, **moov**, sia per quanto riguarda gli atomi della sotto-struttura contenuta nell'atomo **moov**. - presenza di atomi aggiuntivi non standard ai vari livelli del container. - differenze nei valori dei campi contenuti negli atomi.

Tutte queste possibili differenze e tipi di differenze danno vita ad un nu-

meroso insieme di combinazioni che l'analista forense può sfruttare per autenticare una risorsa video.

PARLARE ANCHE DEL FORMATO AVI.

1.2.2 Applications for Video File Container

Il problema dell'autenticazione si basa sull'analisi del segnale audio-visivo, quindi non è nel nostro dominio visto che usiamo i file container che sono metadati. Con i metadati non si può stabilire se il contenuto multimediale è una rappresentazione autentica di un evento originale.

Il problema dell'integrativa invece viene affrontato. Infatti, come descritto prima, i file container di un video è il file container creato dall'ultimo strumento del ciclo di vita. Se un video è modificato anche leggermente da un software avrà un suo particolare container. Questo container conterrà atomi aggiuntivi o in posizione diversa e valori degli attributi differenti rispetto ad un video proveniente direttamente dalla sorgente. È quindi possibile confrontare un container reference, proveniente da un dispositivo noto, ed un container query, che si suppone provenire da quel dispositivo noto, per verificare se il video è stato o meno modificato.

Il problema della source classification ha senso perchè siccome gli standard definiscono solo alcune features obbligatorie, lasciano molta libertà alle case manifatturiere. Significa che ogni classe di dispositivi avrà il suo container diverso. Ciò può essere usato per distinguere fra dispositivi, nel caso di video provenienti da smarthphone, dello stesso brand, dallo stesso brand e model, dallo stesso brand, model e sistema operativo. Per fare ciò occorre rappresentare in qualche modo le varie classi presenti in un training set. Inoltre occorre una misura di compatibilità che stimi la probabilità di un container query di appartenere ad una determinata classe di dispositivi.

Chapter 2

Video File Containers

Chapter 3

Proposed Approach

3.1 Overview

L'applicazione è la source identification ovvero dato un video vogliamo determinarne la sua origine. Da che dispositivi, servizio, piattaforma, programma viene? Siccome il file container è molto arbitraria, lascia tracce che possono essere usata per identificarne l'origine. Per ciascuno delle origini, o classi, di interesse viene posta una domanda binaria, ovvero: tale video proviene dall'origine X? La risposta dovrà essere una likelihood, che mi esprimerà quanto il video appartiene a quella classe; sogliando poi opportunamente la likelihood sarà possibile rispondere alla domanda binaria. Per ogni domanda, vengono fatte due query nel dataset di training: vengono presi un insieme di video che sappiamo rispondere true alla domanda, ovvero appartengono alla classe in oggetto, e un insieme di video complementare che corrisponde ad una classe avversaria, composta cioè da video che rispondono false alla domanda binaria.

Il groundtruth viene quindi diviso in questi due set. Viene poi fatto un merge di tutti i file container (come spiegato in precedenza) in un unico file xml in modo che contenga tutti gli atomi e tutti i possibili valori degli attributi dei due set. Durante il merge, viene inizializzato il vettore dei pesi nella stringa degli attributi; ogni volta che un valore viene aggiunto si inizializza a zero il suo peso, quando troviamo il solito peso più volte il suo peso incrementa, modulato dal numero totale di video del set. Ciò serve a calcolare il potere discriminante

di ciascun attributo rispetto alle due classi.

A questo punto, dato in ingresso un video query di cui vogliamo determinare la classe, guardiamo i valori dei suoi attributi e per ciascuno di questi viene calcolato il rapporto fra i pesi associati a quel valore nei due set. Il numeratore è il set che risponde true e il denominatore il set che risponde false. Tale rapporto ha un massimo di 1 e un minimo di zero escluso (vedi dettagli). I rapporti di tutti gli attributi vengono poi moltiplicati (ecco perchè non può essere zero); più tale prodo è maggiore di 1, più il video è sicuro appartenere al set che risponde true; più è minore di 1 e più il video è sicuro appartenere al set che risponde false, ovvero appartiene al set che risponde false.

3.2 Video Format Tool

3.2.1 *Requisiti Funzionali*

Per poter utilizzare i container dei video come strumenti di analisi, abbiamo implementato un programma che permetta di estrarli da in input un video. Tale programma presenta inoltre varia altre funzionalità che sono state rese disponibili sia come libreria Java che come programma da terminale, implementando sia funzioni di basso livello che di alto livello che fungono da interfaccia per il programma da terminale. Le features implementate sono: - Parse: dato in input un file video di formato mp4-like (mp4 o mov), estrae il file container sfruttando la libreria Mp4Parser e lo salva in un file xml. - Batch parse: dato in input una folder, fa il parse di tutti i video presenti nella folder e nelle sottofolder, ricreato la stessa struttura delle cartelle. - Draw: permette di visualizzare tramite interfaccia un albero, dato in input un file xml. - Merge: dati in input due file xml, permette di unire tali file xml in un unico xml. Prendendo uno come base, vengono aggiunti atomi gli atomi che l'altro ha in più; inoltre per gli atomi in comune, vengono considerati gli attributi, ovvero vengono aggiunti i valori nuovi. - Update: come merge, ma prende in input una cartella che contiene file

xml e fa il merge di tutti i file; considera anche le sottocartelle. - Compare: dati due file xml, li compara ovvero dà una misura di quanto sono simili.

3.2.2 *Implementazione*

Durante lo sviluppo dell'applicazione sono state fatte varie scelte implementative. Di seguito verranno spiegati tali scelte in aggiunta ad ulteriori dettagli. Le feature spiegate saranno solo quelle principali, ovvero parse, merge e compare:

- Parse: la feature parse sfrutta la libreria Mp4Parser, a Java API to read, write and create MP4-like file. La funzione di alto livello è `parse()`, che viene utilizzata da interfaccia. Prende in ingresso due stringhe, l'url del file video da parsare e il percorso della cartella di output dove salvare il file xml. Si occupa di fare il setup per l'operazione, di chiamare la funzione di basso livello `parser()` e di salvare il container estratto in un file xml.

La funzione `parser()` è ciò che estrae effettivamente il file container. Prende in ingresso un file iso del video da parsare, estratto dalla libreria Mp4Parser. Questa funzione crea la base (root) del file xml (`org.dom`) e istanzia un oggetto di classe `BoxParser`. Tale classe ha il compito di leggere il file iso ricorsivamente per estrarre tutti gli atomi e i relativi attributi. Infatti, come spiegato in precedenza, gli atomi hanno una struttura nidificata. `BoxParser` implementa una funzione ricorsiva chiamata `getBoxes()`. Tale funzione prende in ingresso un `AbstractContainerBox` e un `Element`. Il primo parametro è passato come null nella prima chiamata da `parser()`; il secondo è il root element creato in `parser()`.

La funzione `getBoxes()` legge il file iso (passato nel costruttore di `BoxParser`) ed estrae ricorsivamente gli atomi. Inizialmente il `AbstractContainerBox` è inizializzato a null, quindi la prima volta vengono estratti gli atomi direttamente dal file iso; questi sono gli atomi di primo livello (`ftyp`, `moov`, `mdat`, ecc). In generale, per ogni atomo vengono estratti i figli. Viene ciclato sui figli e, in base al loro codice di 4byte rappresentato da una stringa di 4 caratteri, viene chiamato il giusto parser per quell'atomo che è in grado di estrarre i

suoi attributi. Tali "parser" implementano l'interfaccia Wrapper che ha un solo metodo toString(). I wrapper implementati chiamano infatti le funzioni get degli atomi specifici e costruiscono una stringa che costituirà la lista degli attributi e dei loro valori dell'atomo. Un attributo che è stato inserito artificialmente nell'implementazione a tutti gli atomi è l'attributo *count* con valore fisso a 0, che servirà successivamente solo per verificare se un atomo è presente o meno.

Gli atomi non riconosciuti, ovvero per il quale non è riconosciuto il nome di 4 caratteri, vengono gestiti dal un wrapper di default; tale wrapper si limita a inserire solo l'attributo *count*. È sempre possibile estendere il numero di atomi supportati, semplicemente implementando un nuovo wrapper dedicato.

Una volta estratti gli attributi viene creato l'elemento xml col nome uguale al codice e vengono settati gli attributi dell'atomo come attributi dell'elemento xml corrispondente. A questo punto, se l'atomo ha figli viene passato alla funzione parser(), continuando la ricorsione. Quando la ricorsione per un certo atomo arriva alla fine, l'elemento xml viene aggiunto al root, che ogni volta sarà l'elemento xml che era stato passato nella ricorsione, fino ad arrivare al root vero e proprio. A questo punto, il parsing del file iso del file video è terminato e la funzione parser() ha un Element root che contiene il file container rappresentato come file xml. La funzione parse() si occuperà di salvarlo su disco.

Alcuni dettagli implementativi: - per considerare la posizione degli atomi nel container, viene utilizzata una variabile che da assegna un numero ad ogni atomo figlio, in modo da assegnare un ordinamento a tali figli. Ciò fa in modo che possa essere riconosciuta la differenza di posizione fra atomi di due video, utile successivamente. L'atomo *moov* nell'xml non corrisponderà più al tag xml *<moov>* ma ad esempio *<moov-2>*. Due atomi *moov* in posizione diversa nel container sono considerati completamente diversi e aiutano notevolmente nella discriminazione dei file video usando i container. - la posizione degli attributi degli atomi non è considerata. Sia che venga sfruttata la funzione toString della libreria Mp4Parser che quella implementata nei wrapper,

l'ordine degli attributi è comunque arbitrariamente decisi da uno di questi due proxy e non ha niente a che vedere con la scelta della casa produttrice come invece avviene per l'ordine degli atomi.

- Merge: tale feature, unisce due file xml che rappresentano file container. Verrà utilizzata in seguito nella fase di training.

Dato in input due file xml, viene preso il primo come base. Viene esplorato ricorsivamente il secondo e per ogni atomo viene verificato se è presente o meno nel file base: - se è presente in entrambi, si procede a confrontare i valori degli attributi. Se nel secondo file, dato un attributo, viene trovato un valore diverso, questo viene aggiunto al file base, creando quindi un vettore di valori per tale attributo. Al "valore" dell'attributo, che è una stringa, viene anche inserito un vettore di pesi, inizializzati a zero, di dimensione pari al primo vettore. - se non è presente, l'atomo del secondo file, viene aggiunto al file base insieme ai suoi attributi.

La procedura procede poi ricorsivamente, in maniera simile alla feature parse. Dato che il merge produce un file xml come output, il merge può essere fatto tra un file xml di un video e il file xml dato in output ad un merge precedente. L'unica differenza è che quando verranno confrontati i valori degli attributi, non dovrò più verificare se il valore è diverso, ma se il valore dell'attributo del secondo file è presente nel vettore di valori per quell'attributo nel file base.

- Compare: la feature compare server per confrontare due file xml e dare una misura della differenza dei container che rappresentano. L'applicazione pratica è l'Integrity analysis, ovvero verificare che il video è integro, ovvero il suo file container non ha subito modifiche dopo essere uscito dalla camera del dispositivo.

Per confrontare i due file xml, viene scelto uno come reference e l'altro come query. In teoria, per il reference viene costruito un vettore di dimensioni pari a tutti gli elementi del tag che possono essere diversi, quindi per tutti gli attributi di tutti gli atomi (per gli atomi senza attributi abbiamo precedentemente inser-

ito l'attributo count). Gli elementi di tale vettore sono idealmente inizializzati a 1. Per il query viene inizializzato un vettore delle stesse dimensioni. Procedendo come in precedenza, il query viene esplorato ricorsivamente cercando gli atomi del reference nel query e si hanno due casi: - atomo presente: vengono confrontati gli attributi, ogni attributo diverso corrisponde ad uno zero nel vettore di query ad indicare una differenza. - atomo non presente: vengono messi tanti zero quanti sono gli attributi dell'atomo del reference.

Alla fine, otterremo due vettori della stessa dimensioni, uno di uni e l'altro di uni e zeri con gli zeri che indicano le differenze. Siccome fare la distanza euclidea fra due vettori di valori 0, 1 in cui uno è tutto inizializzato a uno corrisponde semplicemente a contare il numero di zeri presenti nel secondo vettore e poi fare la radice quadrata, invece di creare questi vettori, semplicemente viene incrementata una variable per ogni differenza trovata. Oltre al numero di differenze, viene calcolato anche il numero totale degli attributi, quindi delle potenziali differenze, del file xml reference.

La feature compare restituisce infine un json con i seguenti campi: reference, query, tot, diff. (spiegare nomi).

- Albero: serve per rappresentare i file xml come oggetti da manipolare con più facilità. Ogni tag del file xml è un Node o un Leaf, classi che implementano l'interfaccia Tree. Ogni Tree ha delle variabili per identificarla (code), un padre (per il root è null), una lista di altri tree (i figli) e una lista di fields (gli attributi dell'atomo corrispondente).

3.2.3 *Command Line Tool*

'usage: vft [-b | -c | -d | -h | -m | -p | -u] [-i <file|folder>] [-i2 <file>] [-o <folder>] [-wa] -b, -batch batch parse a directory of video files; it recreates the same folder structure -c, -compare compare two xml files -d, -draw draw a tree from an xml file -h, -help print help message -i, -input <file|folder> video file for -parse, xml file for -draw and -compare, a folder for -batch and -update-config

-i2,-input2 <file> second xml file for -merge and -compare -m,-merge merge two xml file into one -o,-output <folder> output folder for the xml file,for -parse, -merge and -update-config -p,-parse parse a video file container into a xml file -u,-update-config merge all files in the dataset folder into a config.xml file -wa,-with-attributes whether to consider attributes in -merge and -update-config or not. Default is false ‘

3.3 File Origin Analysis Tool

3.3.1 *Requisiti Funzionali*

L’implementazione della teoria è stata realizzata con un’applicazione java da utilizzare da terminale. Le funzionalità principale sono: - Training: si occupa prendere i due set considerati e di fare il merge di tutti i file xml. Inoltre calcola il potere discriminante di ciascun attributo relativamente ad entrambi i set. - Testing: è la fase che si occupa di rispondere alla domanda. Dato un video in ingresso, calcola la likelihood usando i config file costruiti dalla fase di training. La likelihood viene modulata usando il logaritmo in base 10 e tale score viene sogliaato sullo zero. >0 true, <0 false.

In foa sono state poi implementate una serie di operazione su database sqllite che serviranno successivamente per la web application che serve da interfaccia di foa e verranno quindi descritte in seguito.

3.3.2 *Implementation*

Sono state fatte alcune scelte implementative per migliorare la capacità del programma di identificare correttamente la classe: - quando cerco un atomo per calcolare i rapporti, prendo l’atomo corrispondente nel config di training cercando il nome intero es. $*\langle\text{moov-2}\rangle*$. Questo vale per tutti gli atomi tranne per $*\langle\text{trak}\rangle*$. Il codice trak è usato sia per l’atomo delle traccia audio che

quello della traccia video. Cercare `*<trak-2>*` potrebbe restituire sì l'atomo con lo stesso nome ma che invece riguarda l'audio e non il video, dovuto al diverso posizionamento nel file container. - alcuni attributi che sono noti essere unici per il video e non per la classe a cui appartiene non vengono considerati nella fase di testing. Tali attributi sono ad esempio la dimensione del file video, o la data di creazione e modifica. Potremmo considerarli e poi sogliare la likelihood in accordo dopo aver fatto una statistica di quanto sposta tale modifica per tutte le classi. (come in compare). - alcuni atomi non vengono considerati, in particolare xyz e udta. L'atomo xyz è dove sono salvate le informazioni relative alla posizione del dispositivo nel momento dell'acquisizione. Sebbene molto utile per certi scopi, non è utile come discriminante per l'appartenza ad una classe di dispositivi. Infatti basta filmare due video dallo stesso dispositivo una volta col gps acceso e l'altra spento per ottenere due container diversi. Tale diversità è accentuata dal fatto che considerando la posizione degli atomi (numerazione dei figli), l'aggiunta di un tag fa incrementare il valore del numero a tutti gli atomi figli che vengono dopo. Questo significa che i container verranno visti come completamente diversi. Per porsi rimedio basta attivare una flag quando ho trovato ad un certo livello dell'albero creato dal file xml. Quando la flag è attiva, per quel livello del file xml, cerco i tag in base al codice vero senza considerare il numero della posizione. (da rivedere). - dato un atomo ho tanti attributi; nella fase di testing per ogni attributo di un atomo calcolo il rapporto e poi ne faccio il prodotto. Ciò rappresenta la likelihood di quell'atomo. Se però i rapporti degli attributi di un atomo sono tutti esattamente uguali, praticamente sto moltiplicando più volte la stessa cosa; è come se stessi contanto più volte la stessa uguaglianza. Per ovviare è ciò la likelihood di ciascuno atomo è modulata in base ad un'entropia data da una formula (da mettere). Quando l'entropia è massima, ovvero tutti i rapporti diversi fanno il prodotto normale; quando è minima ovvero zero praticamente considera solo un rapporto; nei casi medi fa una specie di media per quei gruppetti che hanno i rapporti uguali. Altrimenti alcuni

atomi poco discriminanti ma con tanti attributi pesano tantissimo. - rapporto: quando vado a calcolare il rapporto fra i pesi di un attributo di un atomo, vado a prendere i pesi nel set A e i pesi nel set B. Si possono verificare fra casi diversi: - numeratore e denominatore = 0: faccio rapporto normale. - numeratore = 0 e denominatore != 0: siccome non posso avere un rapporto pari a zero altrimenti col prodotto mi manda a zero tutta la likelihood, devo modificare il numeratore. Siccome tale caso in teoria è favorevole alla classe del denominatore (B), mi basta scegliere come numeratore un valore per il quale il rapporto è sempre minore di 1. Quindi scelgo $1 / \text{il numero di video presenti nella classe B} + 1$. In tal modo è sempre minore di 1, con il rapporto che però varia in base al valore del denominatore. - numeratore != 0 e denominatore = 0: il denominatore non può essere zero. Come prima: mi basta che il rapporto sia sempre in favore di numeratore, quindi che sia sempre maggiore di 1. Scelgo il denominatore pari a $1 / \text{il numero di video della classe A} + 1$. Anche in questo caso l'intensità di tale rapporto varia in base al valore del numeratore. - numeratore e denominatore != 0: questo caso si verifica quando il video di query presenta atomi o valori di attributi che non sono presenti nel config di training. Tale caso viene considerato a favore della classe B. Infatti tale problema, seppure binario, non è simmetrico: nel dubbio dico che non è della classe A (meglio lasciare libero un colpevole che mettere condannare un innocente). Questo caso viene considerato come un 0/1, in cui il peso per la classe A è zero e quello per la classe B è pari a 1 massimo. Modifichiamo il numeratore con i principi del secondo caso, ma invece di considerare il numero di video della classe del denominatore, lo scegliamo di quello del denominatore. Infatti, seppure deciso in favore di B, tale caso è comunque dubbio quindi moduliamo il rapporto in base al numero di video di A. (Spiegare meglio discorso dei rapporti e dei numA ecc)

3.3.3 *Command Line Tool*

‘usage: foa [-cA <xml file>] [-cB <xml file>] [-h | -init | -trn | -tst | -ute | -utr] [-i <xml/txt file or folder>] [-lA <txt/json file>] [-lB <txt/json file>] [-o <folder>] [-v] -cA,-configA <xml file> xml config file for class A, only for -test -cB,-configB <xml file> xml config file for class B, only for -test -h,-help print help message -i,-input <xml/txt file or folder> xml file or txt file with list of xml paths for which compute the likelihood for class A and B, only for -test, dataset folder path for -update -init,-initialize initialize database -lA,-listA <txt/json file> text/json file containing a list of xml file for class A, only for -train -lB,-listB <txt/json file> text/json file containing a list of xml file for class B, only for -train -o,-output <folder> output folder for the training config files, only for -train -trn,-train train a binary classification problem -tst,-test predict the class of a xml file -ute,-update-testing update testing database -utr,-update-training update trainingdatabase -v,-verbose whether or not display information, only for -test ‘

3.4 Web Application

Per utilizzare più semplicemente gli strumenti di analisi forense implementati e spiegati precedentemente, è stato deciso di implementare una web application che funga da interfaccia e permettere ad un utente di customizzare le query e ottenere un output dei risultati. La web app è sviluppata come node.js app ed utilizza un server express. Segue quindi la struttura: html + css per aspetto, js client-side per interattività e ajax call, js server-side per rispondere alle chiamate ajax e eseguire le features e poi rispondere al client (node+express).

Le features principali sono classify, per identificare la classe del dispositivo sorgente dato un video, compare, per integrity analysis, infine test per rendere più veloci i test con query.

Il dataset utilizzato è diviso in training e testing. Ciascun video possiede un

file xml di informazioni utilizzato per costruire il groundtruth. Tali informazioni, insieme ai path del video, dell'info.xml e del file xml del container, sono salvati in un database. Tale database viene interrogato per ottenere i video per il training e per ottenere i video da testare.

3.4.1 Features

È possibile selezionare le features dal nav in cima alla pagina, con in aggiunta una pagina dove spiega come usare tali features.

- Classify: questa feature funziona in due modalità per il training, ovvero manuale e automatica. La modalità manuale si ha quando l'utente è a conoscenza del dispositivo sorgente del video query ma non ha il dispositivo ed intende verificare tale classe. La modalità automatica invece viene utilizzata quando l'utente è totalmente all'oscuro del dispositivo sorgente del video query. Le due modalità funzionano così:

- manual: nel box class l'utente seleziona il brand, il model, e il sistema operativo. Questi elementi sono in sequenza; l'utente può scegliere solo il brand oppure il brand e il model oppure tutti e tre. In base a tale scelta cambierà come la classe avversaria (B) verrà selezionata, ed anche quella scelta (A). Per scegliere la classe A viene fatta un query al database sulla tabella dei video di training in base alla scelta della classe. Se scelgo solo il brand prendo tutti i video di quel brand; se scelgo brand e model prendo tutti i video di quel brand e specifico modello; infine se scelgo brand model e os prendo tutti i video di quel brand e modello con quel sistema operativo specificato. La classe B viene presa sulla base della scelta per la classe A. Se è stato scelto solo il brand prendo tutti i video i cui dispositivi non appartengono a quel brand; se è stato scelto solo brand e model, prendo tutti i video provenienti dai dispositivi di quel brand ma degli altri modelli disponibili per quel brand; infine se è stato scelto brand model e os, prendo tutti i video provenienti dai dispositivi di quel brand e model ma degli altri sistemi operativi disponibili. Nel caso di brand model e os, se non sono

disponibili altri sistemi operativi mi riporto al caso 2, ovvero brand e model. Nel caso di brand model, se non sono disponibili altri modelli per quel brand, mi riporto al caso 1, ovvero brand.

- automatic: nel box class l'utente non seleziona nulla, lascia tutto any. A questo punto verranno testate tutte le classi disponibili nel database. Verrà viene selezionata ogni possibile classe A e di conseguenza ogni possibile classe B.

Nel box upload, viene selezionata il video query che di cui si vuole predire la sorgente. È possibile caricare un video (mp4 o mov) o più video (max 5?), oppure direttamente i file xml dei rispettivi container. Nel caso in cui si caricano i file video, verrà fatto il parse del video prima di procedere.

I file di training sono creati ed aggiornati periodicamente indipendentemente dalla query. Infatti tali file dipendono dai video presenti nel dataset; la scelta della classe da parte dell'utente serve solo a decidere quali file di training utilizzare. Quindi è possibile ottimizzare creandoli prima e andando a selezionare quelli che mi servono di caso in caso.

Una volta che ho caricato il video, scelto la classe A e B, il server utilizza i corretti file di training. Dopo c'è la fase di testing (utilizza foa), che, dati i video caricati, calcola la likelihood rispetto alla classe selezionata. Per il caso automatic verrà calcolata la likelihood per tutte le classi; tali likelihood vengono ordinate dal migliore al peggiore.

Infine, finito tutto, nel box output viene mostrato il nome del video query e la classe testata insieme alla likelihood associata; per il caso automatic vengono mostrati, per ciascun video caricato, i migliori 5 risultati (classe + likelihood).

- Test: serve per fare il classify più velocemente, senza dover caricare ogni volta i video. C'è solo il box class dove selezionare la classe da testare (manual o auto) e il box output. I video da testare vengono presi dal database dalla tabella dei video di testing, dove sono già parsati in file xml. Oltre all'output come prima, viene mostrata una statistica sui risultati della classificazione per tutti i video considerati (true positivi ecc, correct classification rate); infatti in questo

caso è nota la classe vera dei video testati, classe (label) che viene mostrata accanto al nome del file video testato.

- Compare: il compare sfrutta la feature compare di vft. È possibile caricare due video, uno reference l'altro query dal box upload. Una volta finito, nel box output verranno mostrati i risultati, ovvero il numero totale di attributi di reference e le differenze rispetto a query; inoltre lo stesso calcolo è fatto invertendo ref e query, per dare una visione migliore delle differenze ed eventuali somiglianze/uguaglianze.

- vft-parse: se l'utente non è in possesso dei file container, può caricare un video; se però il file video è grande e non si può/vuole aspettare il caricamento sul server, dal box upload può essere scaricato un programmino jar chiamato vft-parse. Tale programma, con una semplice interfaccia in java swing, è una costola di vft completo e permetto di fare il parsing di un video selezionato o di un intera directory. In questo modo è possibile poi caricare i file xml invece dei file video, velocizzando notevolmente il processo per tutte le features.

- problema del parlante: il motivo per cui vengono scelte le classi avversarie nel modo spiegato precedentemente è che si va incontro al problema del parlante. Il problema del parlante consiste nel fatto che ad esempio nel riconoscimento del parlato, se uso un dataset molto vario in cui includo tutte le lingue e dialetti del mondo per fare il training, poi non sono in grado di classificare accuratamente un parlato query; mentre se invece per fare il training seleziono dialetti simili allora sono in grado di classificare correttamente. Ciò sembra contro-intuitivo per la realtà, dove accade il contrario, ma in questi contesti usare cose simili significa dare molto più peso a piccole differenze che nel caso globale andrebbero perse. Questa è l'idea usata alla base della scelta delle classi avversarie, che cercano di essere diverse ma simili alla classe selezionata dall'utente.

Chapter 4

Experiments and results

4.1 Integrity Analysis based on File Container

4.2 Model Identification based on File Container

4.3 Application on Social Network

Conclusions

Bibliography

- [1] T. Gloe, A. Fischer, and M. Kirchner. Forensic analysis of video file formats. *Digital Investigation*, 11, Supplement 1:S68 – S76, 2014. Proceedings of the First Annual {DFRWS} Europe.

Credit