UNIVERSITY OF PAVIA

FACULTY OF ENGINEERING
Department of Electrical, Computer
and Biomedical Engineering

Ph.D. in Electronics, Computer Science
And Electrical Engineering

Mining Forensic Data from File Fragments

Advisor
Prof. Antonio Barili

Candidate
Lanterna Dario
XXIX Cycle

ACADEMIC YEAR 2015/2016

Mining Forensic Data from File Fragments

Introduction

Digital forensics is an interesting field, like a hacker a digital forensic practitioner has to know how, the digital world around him, works. Digital forensics operates to support law enforcement; this is the key point.

Digital forensics is a field of work and research derived from computer forensics. The wide use of digital technology leads the consequence that data and information useful for investigation resides in digital devices. Modern devices use embedded computer to deliver smart functions, some of this devices are cloud connected to store and share data for further functions. The methodology used in computer forensics applied to these devices is called "digital forensics".

The digital devices are rarely the *corpus delicti*, they usually are analysed to define the digital crime scene and to compose the events timeline.

Fragments are common in digital environment analysis. Digital devices manage data splitting them into little parts called blocks, cluster, pages, chunks or packets. The use of limited or fixed quantity of data makes simple to store, elaborate or transmit data. When investigations need an in depth analysis, fragments are the primary source of information. The transmission of data over a network requires that data is splitted to fit in the payload area of a packet; packet is the basic element sent to or received from a network-connected device. Backups save data to avoid data loss, each backup is very similar to the previous, so recent backup systems use deduplication to save space, and this technique split data in fragments called chunks and save chunks in repository without duplicating them.

When an investigation meets ICT environment, it has to handle fragments. Deeper the analysis will be, greater the number of fragments to be analysed. Data mining techniques help to highlight information contained in data. Digital forensic analysis recover big quantity of data and most of this data is composed of fragments; automatic methods for information mining are welcome. The fragments data mining analysis is the most important topic of this work.

Scope of the Work

Forensic analysis requires technical, legal and psychological knowledge. This work starts with an introduction of legal aspect that affects forensic analysis, than it addresses technological aspect of digital devices analysis, and finally it focuses on file fragments analysis using data mining unsupervised methods.

The fragments analysis is the core of the work. I studied the fragments structure, and this work proposes two methods to perform their classification, the first method uses grammar analysis to extract feature from the fragments, the second method uses grammar induction and string distance metrics.

The evolution of storage technologies, changes the fragment generation process, the knowledge of new generation process enables effective recovering algorithms. The storage deduplication uses a fragments generation based on Rabin algorithm; I studied this storage technology in order to understand real implementations and to define how handle fragments coming from these storage devices.

The evolution of technology allows delivering of virtual desktops using cloud services. The virtual desktop infrastructure centralizes storage and computing power, users can connect from anywhere using their own network-connected devices. This technology changes the procedure for digital forensic investigations. Reaching fragments requires the analysis of a whole infrastructure. The analysis requires identifying the user disks, starting from the study of virtual infrastructure. The whole investigation must create a virtual crime scene using all traces left from virtual desktop usage.

The digital forensic analysis of storage devices outputs many fragments that come from unallocated or slack spaces. The number of this file fragments can be very high, to accomplish their analysis automatic procedure are required. The work starts from the replication of the results of previous researches. Then it proposes the study of lexical and grammar elements of the file fragments. The first method developed extracts tokens and analyses their distributions. The second method uses grammar induction and string similarity measures.

The last part of this work focuses particular cases where is important to know, how fragments are generated and used: deduplication and virtual desktop infrastructures. Deduplication techniques need "thorough study using experimental data, and physical acquisition" in order to make an "identification of markers that help to recognize storage technology" (Carlton & Matsumoto, A survey of contemporary enterprise storage technologies from a digital forensics perspective., 2011). In this work, I propose a detailed analysis of two deduplication engines, and I demonstrate that combining fragments present in chunkstore is possible to generate file never stored in the file system. This problem was marginal in fragmentation due to fixed size block allocation, but the algorithms used to split files in order to identify common chunks between similar file, amplify this problem. For this reason, the most important element needed to recover files from deduplicated system is the hash sequence, without the hash sequence is impossible to demonstrate that a file was really existed.

The work ends with the analysis of a virtual desktop infrastructure. This infrastructure uses fragments, to store user activities in an element called differential disk. The analysis of these infrastructures is very different from analysis of physical desktop computers. The work shows the phases of a virtual desktop delivery. This activity leaves many traces in the single elements that compose the infrastructure, and the analysis of the disk requires the knowledge of how it is composed. The primary element of analysis in a traditional digital forensic investigation was the hard disk; the virtualization of this element is made of multiple parts. This organisation of the storage, is functional to delivery requirements, but implies specific expertise to conduct forensic analysis.

Structure of the Work

The first chapter introduces the Italian regulations about proceedings; the focus is on the role of scientific evidences in investigations and in proceedings. The Italian criminal procedure code[1] I introduced it to explain the big picture, where forensic investigations act. I describe digital crime scene paying attention on the models of digital investigation.

[1] In Italian is called Codice di Procedura Penale or with the acronym CPP.

The second chapter briefly introduces digital devices, they are analysed according to the possible role in a crime and the focus is on the traces that we can find on them. The analysis involves storage devices, from little SD[2] cards to cloud storage.
The third chapter is dedicated to the fragments analysis. It starts from the study of simple attributes like BFD and n-grams entropy, and arrives to the grammar analysis and grammar induction of fragments. I used the attributes extracted from fragments to apply unsupervised learning techniques to cluster fragments.
The fourth chapter evaluates results obtained using techniques explained in the third chapter.
The fifth chapter makes an in depth study of two deduplication systems, OpenDedup and Microsoft Windows deduplicated file system. These systems handle fragments to deduplicate data and to save space in storage devices. The analysis of devices that store deduplicated data, require the knowledge of how this technology works, and which traces it leaves.
The sixth chapter introduces virtual desktop infrastructures. It starts analysing traces of user activity in a virtual desktop infrastructure. Then the analysis focuses on virtual disk assigned to the users, and its elements: base disk, personal virtual disk (PvD) and differential disk.

[2] SD stands for Secure Digital and is a type of flash memory.

1. Digital Forensics

Digital forensics is a field of work and research derived from computer forensics, the wide use of digital technology has the consequence that data and information useful for investigation have to be extracted from digital devices. Any forensic analysis is intended to give results valid for use in proceeding, it is fundamental the knowledge and understanding of civil and criminal law basic principles, it creates the right mindset. The chapter briefly introduce to Italian criminal procedure code; the focus is on the role of the expert (or expert witness) in a proceeding, and on the means of evidences research. The chapter close introducing the crime scene analysis and the chain of custody procedures.

The fundamental goal of this science is to support crime investigations. The needs in investigation is a fast and accurate identification of evidences and elements essential to prevent a crime or to write a right sentence for a crime. To obtain this goal, digital forensic researchers revisit many computer science topics, according to a forensic point of view. The most important topics in digital forensic research are:

- extraction of data from devices;
- reconstruction of data structure;
- automatic composition of timeline;
- data mining to deliver automatic extraction of information;
- cloud services (log/data) analysis.

The use of digital technology to commit crimes is clear; the support to investigation requires a big effort in this research field. The increased knowledge and the availability of standardized methodology permit to qualified practitioners to conduct digital investigations (Casey, 2000).

1.1 Digital Devices and Crime

When people talk about digital devices and crime, they think to cybercrime, but there are different levels of interaction between digital devices and crime:

- Cybercrime: is an activity that involves digital devices has the target of the attack, cybercrime is a pure digital crime, the goal of cybercrime is to take control of the target, to inhibit proper operation of the target, to steal data or money handled by the target, to attack new targets from this controlled platform;
- Digital enabled crime: the use of digital devices simplify communication and many “traditional” crimes use these devices as a means, the criminal activities use digital devices as a tool that simplify their activity. Digital tools make simple to reach and to hit the target. Example of this crime are: on-line activity against children, financial crime, cyberstalking;
- Digital related crime: this crime does not require a digital device to be committed, but outlaws use digital devices in their daily life and they leave traces or evidences on them, these elements complete the crime scene, adding a digital crime scene.

Media sometime use the term cyberterrorism, this criminal activity uses the same tools and technology as cybercrime, but “cyberterrorism is politically, socially, or religiously

motivated, aimed at generating fear and panic among civilians, or at disrupting military and civilian assets. Two different components of cyberterrorism can be singled out: terrorist that use of computers as a facilitator of their activities[3]; and terrorism involving computer technology as a weapon or target[4].” (Michael MATES NATO Parliamentary Assembly, April 2001)
The border between the categories is not well defined and sometime, the same investigation falls in more than one. For example, some years ago, one of my colleagues was investigating for a case of online fraud. The trickster sold pets online, the pets were sick, and when people realized the problem, there was no element to identify the crook. However, when traditional investigations locate a criminal for other type of crimes, the analysis of its digital devices detected hidden data, there was traces of TOR[5] (Tor, s.d.) usage (a tool used to secure/hide communication), and artifact of deleted images and thumbnails. These images were the same used on the online pets shop. Analysing the images, the EXIF indicated an originating digital device in use to a suspect, this added elements to solve the case. This case was a digital enabled crime solved thanks to a digital related device. The tool used to hide the communication was instead a cyber tool.
Investigation about every type of crime can be supported by an analysis of a digital devices related to the crime, a criminal usually searches information on internet, uses or carries a smartphone and sometimes uses a GPS[6]. Each action on digital device leaves a trace; the traces can be located on the digital devices, or in the infrastructure used by the device to deliver a service. Digital investigation can use all these breadcrumbs to compose a well-defined crime scene with an accurate timeline of the digital events occurred.

1.2 Scientific Evidence

In a proceeding, digital forensics produces scientific evidences, and the law rules the way they can be used in a crime investigation.
“Evidence refers to information or objects that may be admitted into court for judges and juries to consider when hearing a case” (NIST, 2016). Scientific evidence is an evidence that comes from scientific field. The most common scientific fields is biology with the DNA analysis, chemistry is fundamental in drugs identification, and each science can give added value to evidences identification, handling and preservation. The widespread availability of digital devices requires that also computer science tools are used to search evidences, called “Digital Evidences”. According to NIST definition, “digital evidence is information stored or transmitted in binary form that may be relied on in court” (NIST, 2016).
All scientific evidences require a scientist and special instrumentation to be handled, and often can be carried out only in a specific laboratory. A scientific evidence requires standard procedures and calibrated instrumentation to give a result with a computable error rate.

[3] This can be considered digital enabled terrorism.
[4] This can be considered cyberterrorism.
[5] TOR stands for The Onion Routing, and is a network projected to hide communication source.
[6] GPS stands for Global Positioning System, and this is the name commonly used for devices that are used as a support for positioning tracking and navigation.

Scientific evidence use in court was analysed first time in 1923 in a Federal court of appeal (Frye v. United States, 293 F. 1013, 1014, D.C. Cir.1923), and after this decision the acceptance of scientific evidence followed the so-called Frye rule (or Frye test). Frye rule requires the "general acceptance" from the scientific field community from which they come from. The Frye rule was accepted in court until 1990s when a judge in "Daubert vs Merrell Dow Pharmaceuticals (1993)" decided for a more accurate criterion. After this decision the Dauber rules (also called Daubert standard) (David E. Bernstein & D. Jackson, 2004) was the new acceptance standard for evidence presentation in court. The Daubert rules requires that:

- scientific knowledge produce a scientific method/methodology and the proponent has to follow a "scientific methodology";
- scientific knowledge and methodology need to be subjected to peer review and publication;
- scientific knowledge has to be explained to the court with sufficient clarity;
- potential error rate need to be estimated;
- methodology need to be documented and standard must be fixed for each procedure;
- scientific theory and methodology need to be accepted from a relevant community.

After Daubert standard for each scientific methodology, limits of application must be fixed, and this grants that methodological error can be identified. The error rate estimation is computed using "type I" error rate also called *false positive* (a true null hypothesis can incorrectly be rejected) and type II error called *false negative* (a false null hypothesis can fail to be rejected). Using these values also accuracy, recall and precision can be computed for a full characterization of the methodology and its results.

1.3 Regulation and Lawsuit

Italian criminal procedure code (Codice di procedura penale cpp) (C.C.P., 2016) has a clear separation between investigation phase and prosecuting phase. There are three phase in Italian criminal proceeding: preliminary investigation[7], preliminary hearing[8] and the trial[9].

The responsibilities of investigative phase is in the hand of a public prosecutor[10] and a preliminary investigation judge[11] (GIP) is assigned to the case. When a crime is discovered, law enforcement agencies are required to report events to the public prosecutor. The phase of preliminary investigations starts from this moment and can last up to six month, during this phase evidences are gathered. If there are witnesses or evidence that will not be available afterwards at trial, public prosecutor may require an "incidente probatorio", these procedure results in elements that are valid for future trial proceeding. The Italian regulation requires mandatory prosecution, but if the case is weak a dismissal applies, and the case stops after investigative phase.

[7] English translation of "Indagini preliminari"
[8] English translation of "udienza preliminare"
[9] English translation of "dibattimento".
[10] English translation of "pubblico ministero".
[11] English translation of "giudice delle indagini preliminari" or GIP.

The phase of preliminary investigation gather inculpatory and exculpatory evidence necessary for preliminary hearing phase, to decide if the case will continue to trial. Before the case continues to the preliminary hearing phase, the GIP notifies the defendant of the pending charge against him. The defendant can require further thirty days of investigation based on new evidences produced by the defendant. GIP responsibility requires that it establish if preliminary measure[12] are required, this are jail detention or restriction of personal freedoms.
After preliminary investigation, starts the phase of preliminary hearing, and a new judge take in charge the case, this judge is called GUP[13] that stands for judge for preliminary hearing. The GUP is required to decide if the case continues to trial[14] or if the case is archived.
When the trial starts, the judge can access the files of the investigation. During trial phase, the parties present evidence and witnesses to the judge.

In Italian regulations, admissibility of evidence is defined in 189 e 190 c.p.p.[15] Italian law agree that search of evidences can be conferred to the tree part involved: the prosecution, the defence and the judge. They have equal rights in this task.
All the analysis in digital forensic are based upon four basic operation: identification of digital devices and seizing; preservation and archiving of artifact; analysis of artifacts and writing of a report about results. The identification and seizing is done in the respect of rules about "sequestro probatorio"[16] and according to art. 253 of cpp.
The cpp rules also the use of scientific experts in each phase of a proceeding. The art. 220 of cpp asserts that a scientific opinion is admitted when investigations need to acquire data or evaluation that require specific knowledge (scientific, technical, or artistic). The art. 221 rules the designation of an expert, the judge has to appoint the expert from those enrolled in the appropriate register, or from people that has a certified experience in the field of investigation, an expertise can be considered void, and this event requires a new designation. If the elements of investigation are complex, the number of experts can be increased. In addition, if more expertise are required, the judge can designate a pool of experts. When an expert is designated, it must accept the responsibility, except in case of valid reason for the abstention (cpp art.222 and art 223).
Each part in the proceeding can require support from an expert. The judge expert is called expert[17], while the private parties expert is called expert witness[18], the number of expert witnesses cannot be greater the number of judge experts.
After experts designation the judge expresses the question, at the presence of experts, expert witnesses, defendant and public prosecutor.
The experts, immediately after designation and question communication, start their investigations and answer the question producing a report. If complexity is high, the

[12] English translation of "misure preliminary".
[13] GUP stands for "Giudice per l'udienza preliminare".
[14] The Italian expression is "rinvio a giudizio"
[15] C.p.p stands Italian "codice di procedura penale"
[16] English translation is " freezing of assets or evidence"
[17] English translation of "perito".
[18] The Italian expression is "consulente tecnico di parte".

expert can ask a due date extension, the maximum period after all possible extensions is six month (art. 227). The expert witnesses can ask the judge for remarks or reserves (art. 230), they can be present during expert operation, and can ask for custom tests. They can ask the judge the reports, or if they can repeat examination by themselves. The operations of expert witnesses must not delay the proceedings activities.
The art 234 of cpp rules the documentary evidence[19], the documentary evidence are "documents that represent facts", and they can be text, picture, video or audio recording and any other means. This is the digital forensics area of interest. The rules said that if a document can be altered or destroyed, a copy have to be used, this authorize the use of forensics copy for digital forensics analysis.

During investigation an important means of the evidences research[20] are inspections, searches and seizure. The judge can issue a decree with motivations, when it need to verify traces or effect of a crime. The art 244 and 252 of CPP stands that judicial authority can require survey using technical or scientific means, particular attention have to be taken in case of computer or network systems, to prevent modifying or deleting, experts must act in order to protect data and avoid any alteration. In this case, the expert needs to choose if analysis can be done on-site and how to conduct the analysis. If analysis require a lot of time it is possible to act a seizure.
The code has two different type of seizure. A seizure to prevent a crime called preventive seizure or "sequestro preventivo" (art. 321) and a seizure as means of evidence research called freezing of evidence or "sequestro probatorio" (art. 253-254 bis e 354). The freezing of evidence acts against *corpus delicti* or object related to the crime. Nowadays all electronic devices as smartphone, storage devices and personal computer are objects related to the crime. The art.254-bis rules specifically the seizure of data from online services providers. Data stored in online services assume a growing importance nowadays, online services has data related to different aspects of our life, they store data about our work and social interaction, they keep data about our sport activities and performance, keep trace of our positions, and recently personal medical devices store health data in the cloud.
The CCP art 269-271 rules the wiretapping or interception measures, the wiretapping is the listening of network communications to gain data or information. The articles define the limits of admissibility related to the type of crime. There is also a set of steps, they start from a request of the public prosecutor, then follow authorization from GIP that fix duration and modality, at the end of the interception operation the PM (public prosecutor) decides if keep or destroy data acquired.
Many means of the evidences research require an expert opinion. For example during a seizure in a big IT infrastructure is required the evaluation of right elements and the procedures, and sometimes is needed the live analysis of the data to avoid serious unjustified damage to infrastructure under investigation. The seizure must not exceed the purpose for which it was authorized and when digital media are involved, a brief analysis is required to detect which of them seize.

[19] English translation of "prove documentali".
[20] English translation of "mezzo di ricerca della prova".

1.4 Digital Investigation

(Mark Reith, Clint Carr, & Gregg Gunsch, 2002) The activity done in a digital investigation are collection, examination, analysis, report. The investigation requires proper management of the elements under analysis in order to respect the principles of a correct chain of custody. A good model has to integrate these activity and requirements. The model has to consider that some parts of the proceeding involve non-technical people. The final report has to explain the findings to a judge and a public prosecutor, this step requires the use of terms and concepts that can be simply transferred, the report have to describe the methodology used for the analysis and the level of confidence of the results.
The stakeholders of this process are from different field: law enforcement, computer science and justice. They use different and specific terminology; they have different expertise and background. The use of a model give to the process a formal description and shared knowledge of the steps required. Inside each step of the model, there are detailed procedures use by the person in charge.
The paper (Mark Reith, Clint Carr, & Gregg Gunsch, 2002) proposes a model composed of the following components:

1. Identification: analyse the crime scene to estimate the investigative value of digital evidence, identify the type of expertise needed;
2. Preparation that consists of: tools preparation, environment study, authorization request, experts identification;
3. Approach strategy that: formulates strategies and shows impact, quantifies the number and importance of evidences that can be collected;
4. Preservation: isolate and secure the devices and data collected;
5. Collection: is the use of standard methodology to collect, label, package and transport evidences;
6. Examination: this task require the knowledge of the crime and the criminal activity, to identify each possible trace, of the illegal activity. This phase require a detailed documentation to explain the use of technology especially if unconventional;
7. Analysis: to draw conclusions we have to analyse and correlate all artifacts, all traces and evidences, this phase uses technical and non-technical expertise;
8. Presentation: a good presentation is fundamental to enable judicial system to understand the significance of digital evidence;
9. Returning evidence: all material analysed must returns to judicial system, and sometimes to the original owner, the law codes properly this procedure.

The (Center, September 2013) (Mukasey, Sedgwick, & Hagy, Apr. 2008) (EEG-COE, 18 March 2013) gives an example of model for law enforcement operators, from this model can be created specific guidelines for crime scene analysis and for acting in an accurate way for the collecting phase. The first responder usually starts the collecting phase; the scientific knowledge of a first responder is often limited. The first responder has to preserve the crime scene, and possibly also the digital crime scene. Each phase must be fully documented, the actions done during seizure are fundamental; they are responsible of initial state of the digital evidence, during the examination phase. “The purpose of models is to provide support and guidance in the identification and handling of electronic evidence using methods that will ensure that the authenticity of evidence will be maintained throughout the process" (EEG-COE, 18 March 2013).

The traditional crime scene in the last 20 year has included digital devices. The composition of a crime scene requires attention to all the traces correlated to the criminal events. The digital devices store information, images, and logs of digital events. The traces require a careful analysis; according to their nature require different handling procedures. For example, most of our home and office have a wifi-network, each wifi capable device in the range try to automatically connect to the network, and leave a trace in the log, these traces are usually volatile, and need to be saved as soon as possible, before the switch off of the apparatus.

The traces according to their nature can require a live forensic analysis or can survive up to a post crime forensic analysis. The live forensic analysis acts during inspection, in the crime scene. The post-crime forensics analysis works in a laboratory, on devices subject to seizure.

The analysis of the digital crime scene requires technical skill, and a knowledge of the most recent trend in electronic devices technologies. The number of elements that carry information about activity of the victim or of the criminal are growing each day. For example recently introduced technology is represented by wearable electronic devices, this devices give information about the activity of the wearer, this devices are sometimes cloud connected, in this case the analysis require to access cloud services, this type of events have a good time stamp and can be used as elements of a timeline.

The presence of many digital devices that log events creates a problem of time reference. The time for smartphone and personal computer may be very accurate if they use a server NTP[21] and are network connected, otherwise is required the search of a common event to correlate the time reported in log events coming from different devices. The same considerations applies to the other digital devices as: burglar alarm, IP Camera, game console, router, access point, GPS system, car on-board computer, Smart TV, and so on.

In a workplace, inside a public or private company, the digital crime scene is more complex, and to understand all the elements we need the expertise of technical staff inside the structure. The digital scene in a factory integrate production environment, accounting, human resources and warehouse. The seizure of an element can create a damage greater than the crime we are investigating. Advanced ICT environment instead use cloud resources and seizure of physical elements give no relevant information, because data reside elsewhere. Access to data in cloud is always from remote, and to avoid alteration of this data, we need to identify as soon as possible the cloud service provider and require a freezing of the data.

As stated in (Mukasey, Sedgwick, & Hagy, Apr. 2008) "digital evidence: is latent, like fingerprints or DNA evidence; crosses jurisdictional borders quickly and easily; is easily altered, damaged, or destroyed; or be time sensitive". The proper training and skills are the basic requirement to operate in a digital crime scene.

[21] NTP stand for Network Time Protocol.

The *live forensics* is the set of actions that an expert does in the digital crime scene. The live analysis is intrinsically unrepeatable, and requires evaluation. The elements to take into consideration are:

- Volatility of traces;
- Complexity of the infrastructure;
- Size of the infrastructure.

The unrepeatability of a live analysis depend on some factors: if we analyse a running system, we alter the status of this system; the status of the system in each moments cannot be exactly reproduced, due to its complexity.

In the past, there was only personal computer and some external disks, the standard procedure for electronic devices was “switch off and seize”. This procedure is not always the best solution. Personal computers when powered off loose the RAM content, but some applications keep their data only in RAM, another problem is the presence of encrypted partitions that are accessible only when the operating system is running. To avoid problems a brief analysis is required in each situation.

When the crime scene is a workplace, the complexity of the scene suggests a live analysis to prevent damage to an infrastructure, or to prevent a shutdown. If a workplace is well designed, the IT staff centralizes all the data and computing power, and personal computing devices are only empty box, in this case the solution is to require a copy of the backups of central servers, or if possible a copy of the backups of the period under investigation. Instead, if the workplace has data spread over all PC and they are essential to keep the company operational, the only solution is an onsite live analysis. When the live analysis is required the time constraint impose fast and accurate analysis.

Live analysis is essential when network data are involved, this type of data is extremely volatile. Network traffic if stored without filter, may require high volume of storage, and very fast devices. With a live analysis the tool used to capture the network traffic, can be accurately tuned, this may avoid the use of expensive devices, and the acquisition of data that exceed the purpose of investigation.

The live forensics alter the digital crime scene, so all the action must be accurately documented, and the actions must be less invasive as possible.

The acquisition of evidence during a live analysis, requires the use of known good binaries, it should be avoided the use of binaries present on the system. The acquisition has to follow the order of volatility. To document all the actions, a capture of the screen is recommended.

Recovering of data or log from cloud services, can be considered a form of live forensics, in fact the system we analyse is running, the actions we accomplish alter the original situation, and if data is left in the cloud, we cannot ensure the integrity of data.

1.5 The Chain of Custody

The chain of custody is the set of operations done to preserve the integrity of evidences. The evidences need to remain intact from the time they are collected in the crime scene, until they are presented in court. The chain of custody need to ensure that the evidence collected are the same presented in court.

Standard procedure for a well-done chain of custody requires:

- to properly date and label evidences in the crime scene;
- to store and to handle evidences correctly;
- to document the names of operators that handle evidences different phase;
- to document all the action taken for their analysis;
- to document place and environmental condition where evidences are stored.

Maintain the chain of custody for digital evidences require some extra steps. Digital evidences may reside inside a digital device or in a cloud service, the first step is the extraction of data from the source, the file produced from this step is called the "best evidence". The chain of custody is relative to the best evidence. A digital evidence (or best evidence) is a data file and to detect tampering of data a common method is the use of hash functions. A hash function give an output of a fixed length and any alteration of input data produce a different hash output. The hash value of each evidence does not prevent alteration, but it allows detections of alterations.

Digital evidence can be copied and all the analysis can be done on a certified copy, there is no need to operate on the original evidence. To prove that the working copy is identical to the best evidence we can compare the hash values computed on the best evidence and on the working copy.

When an expert witness ends the evidences analysis, they are returned to the judge with an accurate documentation of the chain of custody. The working copy and any artifact created during the analysis must be removed from supports in analysis laboratory. The experts witness must store no evidence or its copy after its work is finished.

The documents reporting the chain of custody need to be updated on each single event about the best evidence.

After the seizure, the digital evidences reside on digital devices, until the first analysis starts. The chain of custody must preserve data on these objects. Packaging, transportation and physical storage require some considerations:

- Before package evidence, they must be labelled, marked, and a schema of cable connections must be done;
- If a digital device contains latent data, we have to protect this data from loss;
- Digital devices are also physical devices, and we need to preserve also biological traces;
- Packs and bags need to be antistatic and placed in card box, attention must be taken if plastic materials are used to pack because they convey static electricity and develop condensation of humidity;
- Removable media need to be protected from scratch and dust;
- Handheld devices must be preserved in the power state they are found (on or off), and they must be isolated from external radio frequency signals (NIST procedures suggest to use a faraday isolation bag or an aluminium foils);
- During transportation they must be kept away from magnetic fields (example: car audio speakers);
- The period of time they are exposed to non-optimal environmental condition (temperature, humidity, sunlight, vibrations, and intense magnetic fields) must be as short as possible.

The chain of custody for a digital evidence involve the same elements of a chain of custody for physical elements: storage, documentation, access control. However, their digital nature requires for storage and access control some piece of advice. Storage for digital evidences must ensure to be fault resistant. The storage for little quantity of data may be CD, DVD or similar support. However, when the data reach many TB, the only solution are disks array storage. Disks array storage need a climate-controlled environment, because of they are temperature and humidity sensitive, they suffer also if exposed to magnetic fields and vibration. The requirements for this type of infrastructure are the same used for server farm devices. Access control need to be completed by adding prescription about network access to this data. This data requires a trusted platform, network and physical access control, and secure access from local and remote logins.

2. Digital Environment

This chapter describes digital environment, it introduce the classification of the digital evidence category and illustrate some details about disk images and memory dumps. It then analyse a set of digital devices focusing attention on traces they leaves or store inside. The analysis pay attention to the memories used by digital devices, considering that the primary task for a technician is to understand how they operate, both at hardware level that at software level. The last section gives an introduction to storage technologies and a brief description of operations inside the most common file systems, and closes talking about storage virtualization and cloud storage. The last topic are carver functions, and their classification using Garfinkel-Metz taxonomy.

2.1 Digital Evidence Category

The digital evidences classify under different category.

File

A computer file is a container that store data in a structure called file system. Files are organized in sequence of bytes. The way information are organized defines the file format. To store a text in a file characters are coded according to a standard like ASCII or UTF-8. Some formats use metadata to store information about their content and their characteristics.

Document

A document is a special type of file that carry information. To handle a document an application is used (Ex: word processor or image editor), these applications transform data present in a file in a human readable information. Typical type of document are text document, presentation, spreadsheet, image, video, audio.

Log Files

A Log File is a text file with a fixed syntax; it is used to trace events in a digital device. A log file entry contains a timestamp, an event, and some details about environment. The following example shows a typical log line of a web server:

```
[Wed Oct 15 14:22:01 2016] [error] [127.0.0.1] client denied: /var/www/htdocs/test
```

Temporary files

A temporary file caches information that applications use during their running period, it stores information that can speed up applications. A temporary file has a short life, but it is created, allocated, saved, modified and deleted, all these operations leave recoverable traces on the file system.

Forensic Disk Image

A disk image is a file or a set of files that contains the full structure of a whole disk or single partitions. A disk image is done by dumping all the blocks of a disk in a sequential order. Forensic disk analyses are done on disk images rather than original disks. The acquisition of a forensic disk image requires that no write operation be done on the original disk, to comply this requirement practitioner uses software or hardware write-blocker.

When there is a Full Disk Encryption (FDE) or on critical system, we need a live analysis, in this case we install a tool on the target system and we get the image of the un-encrypted file system. FDE offline acquisition results in an encrypted image, that requires further tools for decryption and analysis, usually the image is converted in a virtual machine image, the system is booted with the owner password and then the drives are analysed (Eoghan Casey & Gerasimos J. Stellatos, 2008).

There are different tools to generate a disk image. Linux dd command (disk dump[22]) to generate a disk image the output is a RAW Image Format, this format is a bit-to-bit copy of the original device, the file contains no metadata section. EnCase tool uses Expert Witness compression Format (EWF), this is a proprietary format and specifications are partially open. This format split the file image in multiple sections, and each section is limited to a max size of 2GiB. The image can be compressed reducing the space needed to store images of large disk. The format ewf use metadata to document content of section, and to save examiner name, acquisition date, notes and hash of the whole image. The format store data used for error detection that are computed using a granularity of 32 KiB. The ewf version 2 used in Encase 7 and further versions has the ability to encrypt the data section inside ewf file. Open source forensic softwares usually support Advanced Forensics Format (AFF), this is an open source format, it supports compression and encryption, max image size is unlimited, images can be optionally splitted in many AFD[23] files. Metadata can be stored inside the image or in a separate xml file called AFM[24], an unlimited number of metadata attributes can be added.

To validate a forensic disk image hash function are used. The hash computed on the original disk must be identical to the hash computed on the forensic copy. To avoid collision problem two techniques are used: the hash is computed on multiple chunks; the hash is computed using many different algorithms.

Memory Dump

The memory dump is a file that contains all the content of the memory of a device. The format for memory dumps can be plain or headed. A plain dump is a sequential dump of all the memory allocation units. A headed dump take care of the addressing of the memory device, and a header metadata section precedes each data section, to describe addressing and format. Memory dumps analysis requires the knowledge of the system and the applications it was running.

To obtain a memory dump we have different techniques depending on the situation.

- On a personal computing device, we can install a program and extract the content of the RAM[25] memory; this procedure alters the content of the memory, but in a known way.
- On a memory soldered on circuits, there are two options: JTAG[26] and Chip-off. JTAG is an interface for debugging and support and give access to chip memory content. Chip-off or desoldering requires special desoldering

[22] The Unix command dd make a raw dump of device data.
[23] AFD stands for Advanced Forensics Data.
[24] AFM stands for Advanced Forensics Metadata.
[25] RAM stands for Random Access Memory
[26] JTAG stands for Joint Test Action Group

station to remove the chip from the circuit without damage, and then using a memory reader, we read the chip. The chip-off requires special instrumentation and high skill, because chip are sensitive to high temperature needed for the desoldering process.

The chip-off is an unrepeatable assessment[27] for Italian regulations; this requires a special procedure coded in the art. 360 of C.P.P..

2.2 Digital Device

The (Mukasey, Sedgwick, & Hagy, Apr. 2008) and (EEG-COE, 18 March 2013) are a complete guides to digital crime scene investigation, and this paragraph is based on this special reports.

Computer System

A computer system can vary in size, from a netbook to mainframe. They have different use, computing power and storage capacity.

During investigation, in a private residence we usually found personal computing devices as netbook, notebook, and desktop. The potential evidences in personal computing devices are located in the files stored in the disk; if the device is powered on, also the content of the RAM memory is important.

The analysis of personal digital devices may be done in a laboratory, the disks are not analysed directly, but a forensic copy is done, and this file is analysed. The content of the memories is latent, an order of volatility can be evaluated, if required a live analysis can be performed.

In a corporate headquarters we can find personal computers and servers, the servers centralize applications, data and computing power. In modern company all data and applications reside on servers, this gives a high level in data security, permits centralized backup, controls data redundancy and acts data access control. Servers reside in special spaces called server farm, the server farm can be located inside or outside of the company building. Data in server are organized in file-server and databases, while applications are located in application servers. The seizing in a server farm needs a live forensic analysis, it is fundamental to identify the data to which we are interested. The quantity of data in a company server farm may be enormous, and probably the majority is useless for our inquiry. The analysis of digital devices in a company server farm requires a high technical skill, and the collaboration of internal staff. We have to consider that a live analysis is a time consuming activity, and require more time than seizure, and the presence of an expert.

Storage Devices

Storage devices contain data; their capacity can vary from few Kbyte to some Pbyte. A storage device has a limited computing power, which is used to manage data and communication interface. These devices can be inside a computer system or external. The most common external storage devices are:

- External hard drives: These disks use the same technology as internal disks, but they are connect to the computing system using different interface: USB,

[27] English translation of Italian sentence “accertamento irripetibile”.

FireWire or network. They are used for backup purpose, to share data on the network, or to extend the data storage capacity of a system;

- Removable media: This are optical support and require a specific device to read, their use is limited. In the past were also magnetic support as floppy disk or ZIP and JAZ were used; now they are considered obsolete.
- Flash drives (or thumb drives): they are very small; they use a flash memory technology, and can be connected directly to computers. This type of devices are very common for any storage purpose;
- Memory cards: these types of device requires a simple and fast protocol to be read or written, and are used in many type of electronic device as principal memory support. They are used in digital cameras, mobile phone, music player, game consoles, GPS, embedded systems;
- Disk array: they are rack-mounted system of disks, they can be connected to servers by means of optic fibre or by dedicated network, they give a secure and shared storage space for computer farm infrastructure, this type of storage can be configured to operate business continuity in case of fault. These storage devices can reach a capacity of some Petabytes; they are the common storage solution for server farms.

Except for disk array, storage devices seizure is simple and the analysis does not require special peripheral.

Handheld Devices

The use of handheld devices is strictly personal; they contain many potential valuable evidences. Typical devices in this category are smartphone, phone, tablet, GPS, camera, video camera, digital audio recorder.

The smartphone is a device that has many functions in it. Thanks to its complexity can be used as a phone, a tablet, a digital camera, a GPS navigator, a wearable devices interface, an embedded device controller. It can be connected to mobile networks, to wifi networks and to Bluetooth personal networks. Due to this flexibility, it preserve many traces, and leave traces. Analysing a smartphone we can find documents and trace of web surfing, list of incoming and outgoing calls, and many other information like the SSID of wifi network used for connection, or the details of Bluetooth pairing. Smartphone leave many traces, an example are data exchanged with the mobile network infrastructure. Mobile network, thanks to knowledge of signal strength, knows the approximate the position of each smartphone connected to it. When a smartphone try to connect a wifi network it leave its MAC address and device name.

To extract information from smartphones there are some possible techniques. The simple is to do a dump of memory content using program developed for backup the device. Some devices use memory card to save data and this can be read directly on a PC. Other model can be connected to a computer, using USB cable and they act as an external disk. In some case, a JTAG or a chip-off analysis is required. A smartphone analysis can identify social network or email profile and other traces according to the usage, these elements enable further investigation. The smartphone apps are another source of information, they tell information about hobby, work, favourite books, eating habits, wearable devices, home automation devices.

Peripheral Devices

These devices can be connected to computer to expand functions. Typical devices are keyboard and mouse, microphone and speaker, usb hub, memory card reader and VOIP device, printer and scanner, microcontroller programmer, DVD duplicator, odb2 interface. These devices contain little quantity of data, but tell something about the owner, its skill and activity. For example a memory card reader suggest the existence of memory cards, an odb2 interface indicate the ability to manipulate car control units.

Network Devices

Network devices are used to connect other digital devices to the internal network, and to internet networks. These devices keep logs about connected devices, the simplest and cheaper use permanent memory only for configuration, and log are saved in a volatile memory. Analysing these objects we can discover which devices use the network and when, the most sophisticated (firewall, IDS, IDP, VPN controller) can tell something about less recent use of the network.

Wearable Devices

A wearable device is an electronic gadget that can be wore. These devices have an embedded computing power, and can store information. They are used to capture activity of the wearer using sensor, or they act as a proxy (ex: Apple watch) to another device like a smartphone. Their functions can support sport activity (fitness tracker) or health treatment (wearable glucose monitor for diabetics). This is a new field of electronic devices, and there are new devices and functions each week. These devices tell information about the wearer, its health status, its recent activity, its movements and sometime its habits.

2.3 How Data is Stored on Device

Digital devices store information using different technology: magnetic, optic, electronic. Each technology has a peculiar way of work. The following paragraph will explains how these devices work, and introduces to their forensic analysis. Most devices are defined block devices, this is an abstraction of their real physical hardware space organization, a block is a group of bytes that are handled together, each operation on these devices use blocks.

Magnetic disk

The hard disk drive is a magnetic storage device; it is a permanent memory device. Inside a disk, there is a rotating magnetic platter and two magnetic heads fixed on a moving arm. The bits are stored using different orientation of magnetization. The head read the bits detecting the magnetic orientation transitions. To reach the position where data is stored two movements are needed: rotation of the platter and linear motion of the heads. To connect disk to a computer system, a standard interface is used, the most common interfaces to computer systems are PATA[28],

[28] PATA stands for Parallel ATA (AT Attachment - IBM personal computer AT)

SATA[29], SCSI[30], SAS[31]. The interfaces differ in maximum speed for transfer operations and for command used in the communication protocol. Hard disk hardware is presented to computer system as set of contiguous logical blocks of the same size, and the indication of cylinder, heads and sectors is used only for didactic purpose, the scheme of modern hard disk is based on Logical Block Addressing (LBA). LBA describes disk as a linear sequence of blocks, starting from block 0 or LBA0. In LBA schema there are also interblock areas used for ECC[32] and spare space, this areas are not counted in disk size.

Two important processes prepare a disk for its use in a system, the low-level format and the high-level format. Low level formatting defines logical block on physical disk platters. High level formatting defines partition, file systems, and prepare structures for the operating system. An example of file system data structure are File Allocation Table (FAT), Master File Table (MFT) and "inodes". In digital forensics, file system allocation structures and how this structure are used by operating system is fundamental, knowledge of technological details allows the development of tools used to recover data and to extract log information from devices. The most common class of tool that recover data from disks are carvers. Almost all file systems have been designed to operate on hard disk.

Electronic Memory

The electronic data storage devices store data using only semiconductor integrated circuits. This type of memories have a fast random access, they may be volatile as RAM or permanent as Flash, EEProm[33] or ROM[34]. Each group of memory cells (typically 8) can addressed, so each byte can be read directly.

RAM Memory

RAM is a fast but volatile memory support, it is used in all computing environment from smartphone to mainframe. To extract its content is required a live analysis. The RAM is used as temporary storage for operating system and running applications. The allocation of pages in RAM depend on the operating system, while the organization of memory cells inside each page depend on the application which is using that page. The applications save on the RAM their data structures as variable, arrays, and complex objects. The simplest analysis conducted on a memory dump is the strings search, this analysis give data as mail-addresses, site URLs, passwords, text strings and sometimes IP addresses. Modern tools for memory analysis (Volatility, Rekall) can recognized well-known data structures, and thus simplify the investigation about process running on the system.

Flash Memory

Flash memory are considered non-volatile or solid-state electronic computer storage. There are two types of flash memory NOR[35] and NAND[36], the most used

[29] SATA stands for Serial ATA

[30] SCSI stands for Small Computer System Interface

[31] SAS stands for Serial Attached SCSI

[32] ECC stands for Error Correction Code.

[33] EEProm stands for Electrically Erasable Programmable Read Only Memory.

[34] ROM stands for Read Only Memory.

[35] They are called NOR flash memory, because the cells are like logical NOR gates.

[36] They are called NAND flash memory, because the cells are like logical NAND gates.

type is the NAND. The NAND type flash memories are faster in write erase operations, they allow a greater density and are cheaper than the NOR type, the NAND type has an endurance 10 times the NOR type. The granularity of erase operations in a NAND type is the page (a page is a block of some KBytes), the granularity in NOR type is the byte. NAND flash are accessed as an hard disk using blocks and a block consists of a fixed number of pages, the number of pages creates block size of capacity equal to 16 KB to 512 KB or more. This type of memory suffer at high temperatures and that causes data retention to decrease at a poor level. The flash memory has a limited number of write-erase cycles and a controller manage the wear levelling, this controller check also bad blocks. The endurance of a NAND flash can reach 100.000 write/erase cycles. The NAND flash has also an ECC on a page base, this compensate the moderate error rate in this type of memory, if the ECC fails in correcting a page the block is marked as bad block.

The command set for low-level reading, writing and erasing NAND flash chips are standardized from ONFI (Open NAND Flash Interface), and each chip has its own data sheet that details any operation of the chip. This command set are used in case of chip-off to read the content of these memories.

Different type of devices are based on flash memories:

Thumb disk

This device is also called USB drive or USB key, this is a little printed circuit that integrates a flash chip and a USB controller. Data inside these devices can be organized using any file system. They are used to transfer data from a computer to another, or to do backup copies of documents and folders. They are also used to hide program and data or whole system, computer systems support booting process from these devices. This device type is very important in investigation, because people save any type of data on it.

Memory card

Memory card are called SD[37] card or MMC[38], the present product are called SDHC[39] or SDXC[40]. They are called "secure digital" because of a protected area used for security function of SD standard. Thanks to their small form factor, they are used in portable devices (handheld, wearable). These cards use a serial connection to transfer data, the most common mode of operation are: SPI bus mode to interfaces with microcontroller circuits; One-Bit SD bus mode that support different channel for data and command. To support these communication functions the memory card has a flash chip and a microcontroller. First SD cards supported only FAT12 file system; modern memory cards support any type of file system.

Solid State Drive

[37] SD stands for Secure Digital.
[38] MMC stands for Multi Media Card.
[39] SDHC stands for Secure Digital High Capacity
[40] SDXC stands for Secure Digital eXtended Capacity

The Solid State Drive (or solid-state disk - SSD) has external format factor and connector suitable for replace magnetic disk drive in computer systems. The SSD are very fast if compared to magnetic disks, require less power, and are lighter. They are composed of a set of flash memory chips and a controller section. The SSD support the same communication protocols of magnetic hard disks. The data recover of SSD drive is difficult than magnetics one, because they allocate data in a nonlinear way. The TRIM command fills of zero the deleted areas, the handling of the correct wearing level change the mapping between physical address in the flash chip and address on the operating system. Some controllers scramble data on the flash memory for data security reason. Many newer SSDs use embedded encryption on data stored on them.

Storage Virtualization

Storage space in farm environment or simply on server is virtualized, the need of storage space and data security, requires overcoming the features of the single drive.

The RAID[41] uses a set of disks to improve reliability and/or space; the disks are connected to the system by means of an interface that combines disks to make available more space, or to replicate data on different disk (or slice of a disk) to secure data against single disk fault.

The RAID are named with a number:

- RAID 0 or striping
  This RAID level has no redundancy, is used to increase the total capacity. Data is striped over the disk in order to maximize throughput of read write operation. Recovering data from this architecture require all disk and the knowledge of the striping policy used from RAID hardware;
- RAID 1 or mirror
  This RAID level write at the same time on two or more disks, this configuration improve fault tolerance, and may improve performance in reading operations. Any single disk has the same content, so recovering data requires the analysis of a single disk in the array;
- RAID 2 or bit level striping
  Data is striped distributing sequential bit on different disks of the array. The parity computed using Hamming-code is stored on a further dedicated drive. No commercial computer system use this RAID level anymore;
- RAID 3 or byte level striping
  Data is striped distributing sequential byte on different disks of the array.
- RAID 4 or block level striping with dedicated parity
  This RAID level is now replaced by RAID-DP that use two dedicated parity disk;
- RAID 5 or block level striping with distributed parity
  This level is used frequently; the parity information is distributed using all drives in the array. This RAID level resists to the failure of any single disk

[41] RAID stands for Redundant Array of Inexpensive Disk or Redundant Array of Independent disks

in the array, when a disk fails, a spare disk is used to reconstruct all the data present in the failed disk, this process is done using the parity information.

- RAID 6 or block level striping with double distributed parity
  This level is based on RAID 5 but the double parity give an improved fault tolerance;
- RAID 1+0 or RAID 10
  This level stripes data (RAID 0) using in a set of mirrored disks (RAID1);

The use of RAID techniques in a system requires attention during forensic analysis, the volumes may be allocated on more than a single disk and if the original controller cannot be used, the information about RAID configuration must be detected. Stripe size and stripe map must be determined to recover data (Zoubek, Seufert, & Dewald, March 2016).

The RAID solution can be used directly by a single computer system or in enterprise disk array storage. An enterprise disk array adds a further level of virtualization, this type of devices is common in computer farms. These storages are connected to multiple servers by means of a fibre channel network called SAN (Storage Area Network), each server can access only the part of the storage (datastore) configured for its access. The analysis of data in a similar device is impossible (very hard) starting from the analysis of each single disk, but this storage devices can support analysis with their proper functions. The clone function allows to create a perfect copy of a data store, without halting the system under analysis. We can transfer the clone on a different storage device for laboratory analysis. When the exported data is a virtual machine, it is possible to manipulate it to be powered on, with the support of a virtualization software. In this type of infrastructure sometimes, we can find the so called "volume consolidated backup", in this case we can restore the datastore at different date according to the backup policy.

CLOUD Storage

Cloud storage is another type of a virtual device; the connection to this type of storage is done using a network connection. The physical supports are located in remote sites and the infrastructure provider ensure that data is available, accessible and protected. Cloud data can be used by application running on digital devices, but also from a web based management console. Some cloud providers are specialized provider, and accept only a fixed type of data such as images, video or virtual machine. Cloud storage services, to provide a secure service, log many information about user and their activity. This information are usually stored in log repository and are available using API to access all information about an account (Chung H. , Park, Lee, & Kang, November 2012). Smartphones can be configured to automatically transfer data on cloud storage and release space on the device, this is important in forensic analysis. A specific attention have to be paid considering that, these storages can be accessed and modified simultaneously from different channel. During investigation, it is important to identify if this type of storages are used, and require a freezing of data. In a server farm, system administrator uses cloud storage resource to allocate operating data, but also for disaster recovery purpose. Disaster recovery sites usually contains the same data of principal sites and this give the opportunity to access data destroyed on purpose or accidentally on the primary site, the systems in data recovery site are not in a production environment, and this simplify seizing in some situation.

2.4 File Systems

This section is inspired by the most important book dedicated to file system in the field of forensics, "File System Forensic Analysis" written by Brian Carrier (Carrier, 2005). The file systems analysed are FAT, NTFS and ext3, presentation of their structure base concepts is finalized to introduce the file system forensics.
File system are the structures used to organize data in digital storage devices. The knowledge of key concepts allows to Analysing a device, recovering files and artifacts leaved from deleted files. The file system allocates files in block-based device and manages the structure that permits files allocation. The blocks that contain no data at a particular time are called "unallocated blocks", an unallocated block contains no allocated valid data, but can have contained data in the past, the content of an unallocated block can be read and its content analysed.
The files allocated in a file system may occupy more than one block, and rarely their size is an exact multiple of the block size, this creates an area that contains no data pertaining to the file allocated in that block, this unused space is called "slack area" (Figure 1 - Slack Space). The slack area after the allocation of a file, can be filled with zeros as it happens in exts file systems, or can be left untouched as it happens in FAT and NTFS.
The analysis of the slack area is important as the analysis of unallocated block. In slack areas can be found only little fragments of previously allocated files. The analysis of a fragment can give information about previous content and is illustrated afterwards.
All file systems define two fundamental elements, files and folder. A file is a data container and a folder is a files and folders container. These elements have descriptive entity called metadata. Metadata contains timestamps for files and folder operations as: modify, access, create and delete. Metadata gives necessary information regarding security and ownership.
File system uses structures that allow locating file system objects, creating access and modifying the objects, these structures contains all metadata. These structures are the Superblock for Unix-like file systems, the Master File Table (MFT, 2016) for NTFS and the File Allocation Table (FAT) for FAT file system.

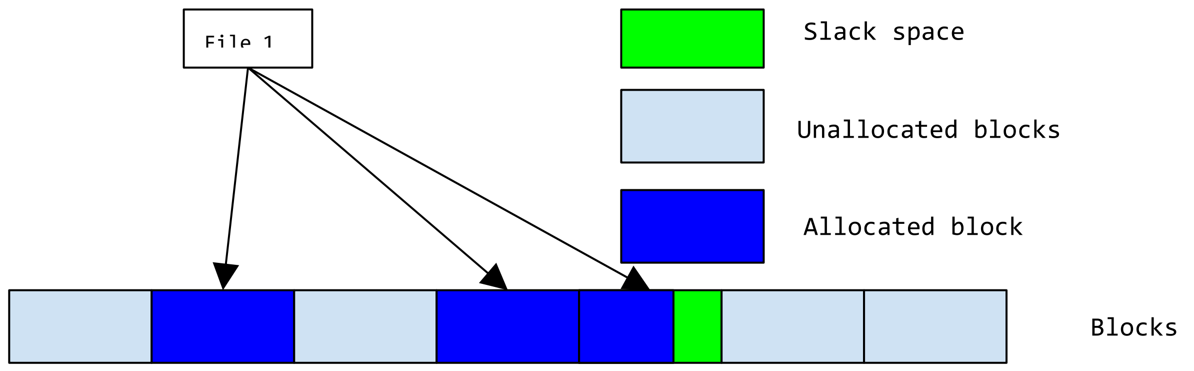


Figure 1 - Slack Space

In the following description is used the term “cluster”, it represents the allocation unit in FAT file system, the concept is similar to the “block” used in description of storage device and is used in the same context of the “allocation unit”, all these terms describe the logical entity that can be addressed in a block device.

2.4.1 FAT

The name FAT stands for File Allocation Table; this is also the name of the most important structure of this file system, the file allocation table.
The FAT file system according to the addressable number of clusters is called: FAT12, FAT16 or FAT32. The numeric part specify the number of bits used to address clusters. The partition identifier for FAT12 is 0x01, FAT16 is 0x04, 0x06 0x0e and for FAT32 is 0x0b and 0x0c, the dual identifier for FAT16 and FAT32 indicates if the devices use CHS or LBA architecture. The FAT is supported from all modern operating systems; the FAT12 is commonly used in embedded devices. FAT12 and FAT16 support only 8.3 file names format while FAT32 support long names. The 8.3 file names format reserve 8 characters for filename and 3 characters for the extension. The extension is one of the methods used to identify the file format. The long file names format uses up to 255 characters for filename, but the full path cannot be greater than 260 characters. This file system family has no security constraint and it is not possible define the owner of an object.
The layout (Figure 2 - Fat 32 layout) consists of five areas: boot sector, reserved area, FAT area, directory entries area, data area.

Boot	Reserv	FAT Area	Root Directory	Data Area

Figure 2 - Fat 32 layout

The *boot sector* contains:

- the dimension of a data unit or cluster, calculated multiplying “bytes per sector” by “sector per cluster”;
- the size of the reserved area;
- the number of allocation tables, that are typically 2, the main FAT and a backup FAT;
- the maximum number of files in the root directory.

The *FAT* is the main structure of this file system, it specify the allocation status of a cluster, and contains the chain of clusters pertaining to the same file. The last cluster of a file contains the EOF indication, while if a cluster is not allocated its value is 0. The EOF is a number greater than maximum size addressable from FAT structure, so it is different for FAT12, FAT16 and FAT32, if a cluster points to the last cluster in the FAT, this cluster is damaged and unusable.

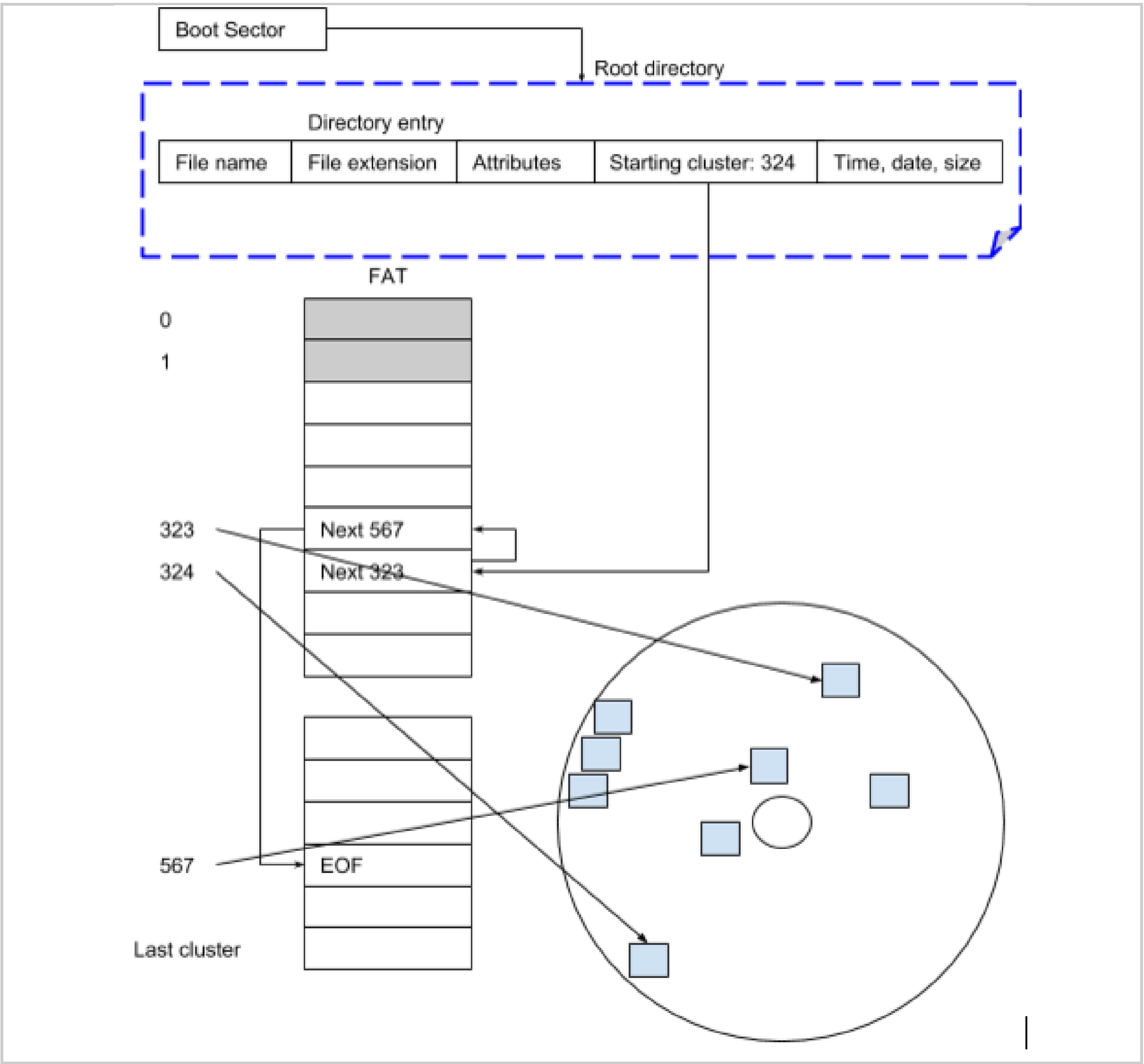


Figure 3 - File Allocation Table structure

The FAT structure (Figure 3 - File Allocation Table structure) starts immediately after the reserved area, there is no header or footer, the length of this structure can be calculated using information in boot sector. Cluster 0 and 1 store information about the system and the dirty status of the file system.

The root *directory* structure contains the metadata for files and directories. Each entry is called directory entry; the first byte of each entry contains the allocation status where 0xe5 or 0x00 stands for unallocated, while a valid character is for allocated entry. The bytes from 2 to 10 are for the rest of the file name. The 11 bytes report the flag values; that are: read-only, hidden, system, volume label, long-filename, directory, archive. The bytes from 13 to 25 report time and data for creation, access and modify operations. Time indication in FAT use local time of the computer, when a FAT device is analysed the local time of the storing device must be recorded. The last two attributes indicate where the allocation starts, the address is expressed in term of clusters, the second attribute is the total size of the file or directory, the size of a directory is 0.

The main problem of FAT is fragmentation; a file system is defined fragmented if the allocation of a file is done using non-contiguous clusters. After long usage of file systems, the file allocation require an effort to find free cluster because the unallocated cluster are not contiguous. When the storage device was only magnetic disk, this was the main reason for system slowness. The recovery of disk with a fragmented allocation, and without the allocation tables, requires a big effort. The content of the recovered clusters have to be parsed to detect which fragment are part of the same file.

2.4.2 NTFS

The acronyms NTFS stands for New Technology File System, this is a standard file system for Microsoft Windows operating systems[42], its maximum number of storage allocation units($2^{64} - 1$). There is no official low-level documentation, but there is some good unofficial source of information (Carrier, 2005) (Linux-NTFS, 2016). The NTFS is designed to be secure, and to support large disks or storage devices, compression is an internal feature of the file system and transaction are logged.
The boot sector contains information that describe the volume, and the boot code that run OS. The boot sector has a backup located at the end of the file system, if the Disk Administrator flags a volume as “unknown”, the boot sector backup can be used to restore the volume Figure 4 -Volume with NTFS File system).

The files that contain metadata, logs and information about NTFS have a name that starts with the character $:

- $MFT Master file table;
- $MFTMirr backup copy of $MFT;
- $LogFile journal that store metadata and transaction;
- $Volume contains volume information;
- $AttrDef defines attribute;
- $Bitmap store allocation status of each allocation unit in the file system;
- $Boot contains Boot Sector and Boot Code;
- $BadClus maps cluster that have bad sectors;
- $Extend folder for file system extension
    - $Reparse
    - $Quota
- $I30 NTFS allocation Index

The fundamental concept for NTFS is “*everything is a file*”. The data and the metadata are stored in a file. The most important file for NTFS structure is the $MFT file, the acronym stands for Master File Table. The pointer to the MFT starting point is inside the boot sector. The MFT is organized in entity called records, each entry in the MFT is stored

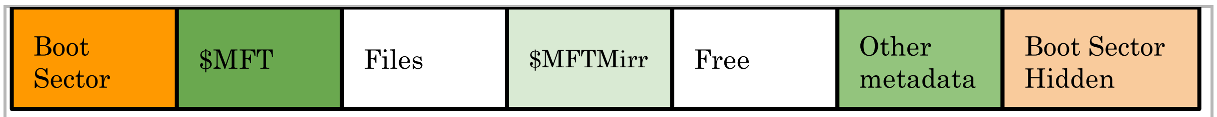


Figure 4 -Volume with NTFS File system

[42] Source https://technet.microsoft.com/en-us/library/cc938432.aspx: 2^64 allocation unit on disk format, 2^32 allocation unit on real implementation, the practical limit for physical and logical units that use NTFS is 2 Terabytes.

in the record structure, the MFT file starts with a header of 16 reserved file records, the first record is for $MFT itself.
Each record has 42 bytes of header that has a fixed format. The other attributes in the record are composed of two parts, the header and the content (data stream). The header is standard for all attributes and identify the type of attribute, the size, the name and a set of flags that tells if it is compressed or encrypted and if an attribute is resident or non-resident. The content part of an entry attribute can have custom format and size, if it requires a high quantity of space can be non-resident. If the attribute content is resident, it starts immediately after the header. If an attribute is non-resident it is stored in a "*cluster run*", the clusters run are consecutive clusters defined by a starting address and the length.
The standard length of a record is about 1 KB, the size of $MFT records is declared in the boot sector, a record contains the attributes relative to a file as owner, rights, MAC[43] dates, filename, encryption keys, compression flag and unused space. The total size of the $MFT grows according to files allocation. If a file entry has a high number of attributes, it can use multiple entry. The very first entries in $MFT contains metadata of "file system structure files" these files have names that start with the char "$". The File System Data is an area that contains all the files, except the little ones stored directly inside the $MFT. The $MFT has a copy that is used to recover file system in case the main $MFT is damaged. The cluster size is fixed according to the size of the volume; greater volumes imply greater cluster size. The NTFS is a journaling file system, all unsuccessful operation can be rolled back, and the file changes are logged in the journal files: $LogFile and $USNJrnl. The time stamps in NTFS are updated for Modify, Access (MAC) and Creation events, the $MFT also tracks and keeps the last update to an $MFT record. The "file time" is represented using a 64-bit value that counts the number of intervals of 100 nanoseconds that have elapsed from the 1st January of 1601 (UTC), the dates in $MFT are stored using UTC reference and do not use the time zone indication, they are localized when displayed to the user.
The file slack space in NTFS is not wiped, and little fragments of previous file can be retrieved.
When a file is deleted in NTFS, its entry in SMFT is simply marked as deleted, and the clusters in $BitMap are marked as free. The content of cluster occupied by deleted file remain untouched in unallocated areas.

2.4.3 EXT

Linux distributions use ext3 and ext4 file systems; the ext3 is a direct upgrade of ext2. The ext3 is a journaled file system and allows online system growth. The allocation unit used in this file system is called block, ext3 can allocate up to 2^{32}blocks, using 4 KB block size the maximum file system size is 16 TB. The layout of a partition is a starting Boot Block followed by n Block Groups. Each Block Group is composed of: block group descriptor, data block bitmap, inode bitmap, inode table, data blocks.

[43] MAC stands for Modify Access Creation.

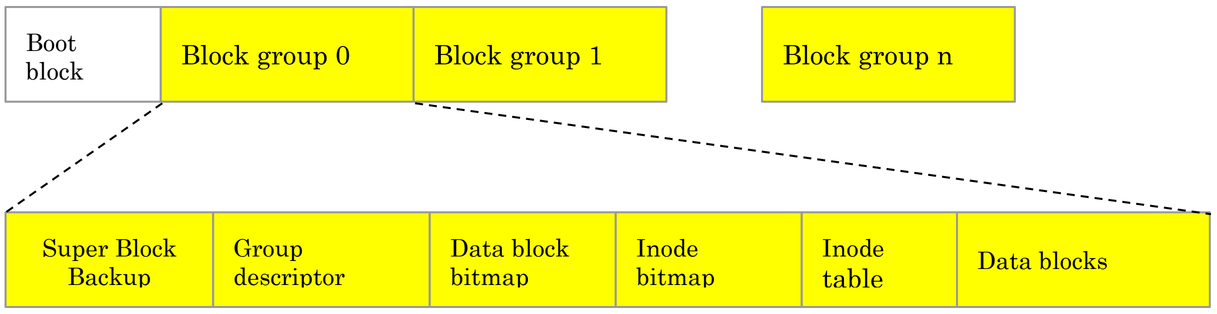


Figure 5 - EXT file system

The superblock (Figure 5 - EXT file system) is 1024 bytes after the start of the file system, and its size is 1024 bytes, the first block group contains a backup copy of the superblock, the superblock contains metadata of the filesystem. The metadata stored inside the superblock are the number of its copy in the block groups, block size, number of blocks, and number of free blocks number of inodes, number of free inodes, and many others. The super block contains the first inode which is the root "/" where all the file system structure starts.

The group descriptor is in each block group, and there is a backup inside the same block group. The content of a group descriptor is blocks bitmap, inode bitmap, inode table, count of free block, count of free nodes, and count of inodes allocated in each directory.

The objects in the file system are defined by an inode, the inode is a structure that contains pointers to the blocks where data is stored, the inode contains also all the metadata except the filename. The metadata stored in inode are file type, owner and group, security permissions, flag and size, number of blocks, pointers to data blocks, access/change/modify/delete date, links and some extended attributes. The ext3 wipe inodes when an object is deleted, the only way to recover deleted files is to analyse journal file.

The directories are regular files, and its type has a special value, the directory contains a set of directory entry, each directory entry contains the description of files or subdirectories, the max length of a directory entry is 255 bytes. The fields of directory entries are inode of the object (file or directory), file type, name length and record length, name of the object. The object type in a directory entry are not only files and subdirectories, but there are also special type as character device, block device, named pipe, socket and symbolic link.

To delete a directory entry the length of the previous record is incremented. When a new allocation is needed in the directory, the list of directory entry is controlled, and if some record is longer than needed for its data structure, a new directory entry is allocated in this slack space, otherwise is allocated at the end of the directory structure.

Time values are defined using UTC time, using time zone setting is possible to identify the right data for events. Using simple operating system tool the time/date attributes of files and directories can be modified, to have a good level of confidence on these values; they must be correlated to data stored in journal and with other file system elements.

The Ext file systems write zero in the slack space, traces of old files remains only in unallocated blocks.

2.4.4 Linux LVM

LVM stands for Logical Volume Management; this is a support in management of logical volume and filesystems. LVM creates an abstract layer from storage hardware, and creates a simple form of storage virtualization; the virtual partition extends or shrinks without worrying about contiguous allocation of physical space. The elements of LVM are volume group (VG), logical volume (LV), physical volume (PV).
In LVM structure, the element called physical volume is a physical hard disk or a hard disk partition, or a LUN defined on an external storage device. The PVs are splitted in chunks of the same size called physical extent (PE). The volume group is the structure that contains the physical volumes; PVs can be added or removed from a VG, the PEs of a VG are the sum of the PEs of all the PVs that belong to the VG. The logical volume (LV) are a concatenation of PEs inside a VG, the physical extent may belong to different physical volumes. File systems are created inside the logical volumes.
The analysis of a system where file systems are allocated in an LVM requires the knowledge of the configuration, or the use of functions of LVM system. LVM can create snapshots of the logical volume (read-only or read-write), this function freezes the situation of the LV, and using read-only image the content can be analysed safely. Another function supported from LVM is the *vgexport* that creates an export of the volume group in a file; this export can be mounted on another system for the analysis. The analysis of the disks without be aware of the LVM gives a sequence of big chunks that are equivalent to the physical extents, and the sequence depend on the configuration history of the disks.

2.5 Carvers

A carving process helps to recover files from a disk when file system information are not available. The file system information are not available when file have been removed from file system, or file system is heavily corrupted. Carving tools play an important role in digital forensic investigations. They help to recover deleted files that may contain valuable information. Another category of file that can be recovered are temporary files that are automatically deleted by system procedures after creation and use. The original information contained in a deleted file undergoes a progressive degradation while the free space is reclaimed and reallocated. Carving algorithms search for known characteristics of common file formats. The most important are: header and footer signatures, recurring markers, syntax, keywords.
In digital forensic investigations, an important role is played by carving tools. These programs help to recover file from magnetic memory support. These tools recover, when right condition are met also temporary files, these files are automatically deleted by system procedures after use or expiration, typical temporary files are internet cache, and files before encryption or compression process. Deletion process in most file systems do not wipe file areas, but unlink them removing their metadata from allocation structures. File content is preserved until other files are allocated in the same areas. Sometimes information are partially overwritten from new allocated documents, and only fragments of original files are recoverable. File carvers can analyse space without the support of file system information, they can extract files parsing raw data.

Simson Garfinkel and Joachim Metz have proposed the following file carving taxonomy (GARFINKEL-METS, 2016):

- Carving
  General term for extracting data (files) out of undifferentiated blocks (raw data), like "carving" a sculpture out of soapstone.
- Block-Based Carving
  Any carving method (algorithm) that analyses the input on block-by-block basis to determine if a block is part of a possible output file. This method assumes that each block can only be part of a single file (or embedded file).
- Statistical Carving
  Any carving method (algorithm) that analyses the input on characteristic or statistic for example, entropy) to determine if the input is part of a possible output file.
- Header/Footer Carving
  A method for carving files out of raw data using a distinct header (start of file marker) and footer (end of file marker).
- Header/Maximum (file) size Carving
  A method for carving files out of raw data using a distinct header (start of file marker) and a maximum (file) size. This approach works because many file formats (e.g. JPEG, MP3) do not care if additional junk is appended to the end of a valid file.
- Header/Embedded Length Carving
  A method for carving files out of raw data using a distinct header and a file length (size) which is embedded in the file format
- File structure based Carving
  A method for carving files out of raw data using a certain level of knowledge of the internal structure of file types. Garfinkel called this approach "Semantic Carving" in his DFRWS 2006 carving challenge submission, while Metz and Mora called the approach "Deep Carving."
- Semantic Carving
  A method for carving files based on a linguistic analysis of the file's content. For example, a semantic carver might conclude that six blocks of French in the middle of a long HTML file written in English is a fragment left from a previous allocated file, and not from the English-language HTML file.
- Carving with Validation
  A method for carving files out of raw data where the carved files are validated using a file type specific validator.
- Fragment Recovery Carving
  A carving method in which two or more fragments are reassembled to form the original file or object. Garfinkel previously called this approach "Split Carving."
- Repackaging Carving
  A carving method that modifies the extracted data by adding new headers, footers, or other information so that it can be viewed with standard utilities. For example, Garfinkel's ZIP Carver looks for individual components of a ZIP file and repackages them with a new Central Directory so that they can be opened with a standard unzip utility.

There are some open source carver like Photorec (PHOTOREC, 2016) Foremost (FOREMOST, 2016) and Scalpel (SCALPEL, 2016) (the last is a complete rewrite of Foremost 0.69 made by G.G. Richard III). Analysing their source code is clear how many different techniques are used to recognize file system type and its metadata, the carver extract blocks, and try to identify file type contained. Scalpel and Foremost use a configuration file that describes formats, while Photorec uses a source C file for each format, in these files are defined characteristics to look for, to recognize a file format (best examples are: file_pdf.c, file_jpg.c/h).
When a carver cannot recover a whole file, it outputs fragments that need further analysis, much of these files are textual fragments. A text file has a simple container structure, it contains plain text. A text file ends with a typical char called end-of-file (EOF). Plain text is a sequence of printable character. Typical encoding used in text file are ASCII, UTF8, UTF16. Text file requires sometime manual procedure to be properly classified. Since file carvers recover a large number of text files, in order to obtain good efficacy in filtering results, before proceed to manual analysis, we need to identify a set of metrics to use in data mining techniques for automatic file classification.

3. Fragments Classification

This chapter is the core of the entire work, it concerns fragments analysis. I started with a revision of previous researches, which use simple attributes like BFD, n-grams entropy, RoC (rate of change) and statistical distributions to classify fragment type. The usage of simple attributes shows problems analysing fragments that contain different languages and mixed formats.

A fragment is an isolated part of a whole file. We can find fragments in disk analyses, RAM dumps or network-traffic captures. Digital forensic analysis of fragments starts from classification. To classify a fragment means try to determine which was it original file format, this step is important to extract useful information. The nature of fragments extracted from a memory dumps is related to the allocation of memory during execution of applications, so the structures of data in these fragments require an in depth knowledge of the system originating the dumps, and the accurate knowledge of which programs run in that system. The network transports data using packages, when there is a communication over a network; data are splitted in packages and fitted in payload for sending. In the payload of network packages, we can find fragment of files, communication between client and server, encrypted communications. The fragments that contains part of communication between applications has the same problem of memory dump analysis, they require the knowledge of the applications, and their communication protocol.
Analysing disks, fragments originate from unallocated space or slack space. Another source of fragments are non-contiguous blocks in failed fragmented disks. The analysis of deduplicated file systems, outputs fragments of different size, which requires a specific work to become useful data. The Virtual desktop Infrastructures (VDI) use storage to store changes made by users to template virtual desktop, analysing this fragments we must consider that fragments are meaningful if related to a specific template.

I have proposed a grammar analysis and a grammar induction on fragments. The result of these analyses is a set of complex attributes. I analysed this outputs using token frequency distribution, and clustering fragments using grammar attributes.
To extract attributes, I focused on the different strategies to implement lexical analyser and parser. I started using a detailed lexical analyser capable to recognize a high number of tokens, but it gave rise to many problem of lexical ambiguity. Then I approached the problem freezing the context, and focusing on grammar construct for specific formats, but this determinist approach requires the previous knowledge of the context or the iteration of the same process using different contexts. The last approach used a restricted set of tokens and grammar constructs (less than 30), these tokens are common to a large number of formats. The last approach gave satisfactory results in clustering analysis.
Using the results given by restricted tokenizer, I continued the analysis using grammar induction algorithms. I coded tokens using a single character for each token, then I extracted a set of rules from each fragments. The output rules were characters strings; to group this rules I used strings distances. The result was a good separation between fragments format. This method works rather well in case of unknown formats, or mixed formats.

To test previous works, and to verify results of the method proposed, I developed a set of procedure using Python language, Qt Library, and the Orange[44] Library.

3.1 Previous Research

Many authors in their work focus attention on the large amount of digital artifacts produced by modern carvers and their analysis and classification (Roussev & Quates, 2013) (Garfinkel S. , 2007) (Garfinkel, Nelson, White, & Roussev, 2010) (Karresand & Shahmehri, 2006) (Nguyen, K., Tran, D., Wanli Ma, & Sharma, D., 2014) (Sallis, Aakjaer, & MacDonell, 1996). (Penrose, Macfarlane, & Buchanan, 2013) work sets as a priority in the digital forensics research, the speed of fragments analysis and underlines the big problem of encryption and compression.
To perform a fast analysis we use features simple to extract through low complexity algorithms. In (Moody & Erbacher, 2008) the researchers assert that a metrics-based approach can detect only broad classes of fragments, such as textual, binary, and compressed or encrypted fragments, while more detailed classification results are inaccurate. The work (Poisel & Tjoa, 2013) presents the current state-of-the-art of file carving and they propose an ontology driven approach for data recovery and fragment type classification.
The article (Moody & Erbacher, 2008) identify two fundamental concepts: file type and data type. File type relates to the application used to access the file, while data type relates to the contents of a file. Some file types - like Adobe pdf or Microsoft doc - may contain more than one data type. (Moody & Erbacher, 2008) research uses statistical method as average, kurtosis, distribution of averages, standard deviation, distribution of standard deviation in byte distribution to classify fragment of different data type.
An important and frequently used parameter is Shannon entropy or information entropy. If we organize our data in byte, the entropy can range from 0 to 8. A fragment that shows low entropy (less than 3) contains only short sequences repeated and composed of a very limited number of symbols, an example are padding sequence. A high entropy fragment (greater than 6.5) identifies compressed or encrypted data.
(Penrose, Macfarlane, & Buchanan, 2013) addresses the problem of high entropy fragments classification using statistical analysis of randomness on 4KiB fragments. (Axelsson, 2010) uses entropy and compression ratio for file type classification, they compute the values of these parameters using sliding windows of fixed size. This approach can detect different entropy zone, creating a fragment fingerprint. They also proposed to weight entropy using the number of symbols in the window. They used the technique to extract feature from binary and multimedia files, conversely the procedure is useless for compressed or encrypted files where entropy is always (Axelsson, 2010).
Another attribute of a text fragment is BFD (Byte Frequency Distribution) the computation requires low computing power. It computes the frequency of each possible byte value in the range 0x00 to 0xFF. (McDaniel & Heydari, 2003) research used the byte frequency cross-correlation (BFC) distribution in the same file for their analyses. An example is the use of byte values 60 (<) and 62 (>) to identify HTML file format. The use BFD and BFC requires a previous knowledge of file formats fingerprints. (Veenman, 2007)

[44] http://docs.orange.biolab.si/2/reference/rst/index.html

presents an application of statistical learning, using as a feature the bytes frequency distribution, the entropy and the Kolmogorov complexity (or algorithmic complexity).
(Sportiello, L. & Zanero, S., 2011) analysed fragments using support vector machine SVM, the features were the complete set of byte distribution (mean byte value, entropy, complexity, BFDs, entropy BFDs, complexity BFDs, Rate Of Changes, Word Frequency Distributions).
Both entropy and byte frequency distribution are invariant with bytes order. To account bytes order, we can use n-grams[45] instead of single symbols, as in (Li , Wang, Stolfo, & Herzog, 2005).
The second step of our investigation concerns lexical and syntactical analysis. Using grammars (Underwood & Laib, 2012) investigated how to preserve and how to test for good preservation of digital media. In (Underwood & Laib, 2012) research is suggested the use of an "attribute grammar" to investigate the semantics and to represent the context sensitive grammars. The grammars are used for interpretation or translation in new languages/formats of the digital contents.
The third step in our procedure involves grammar induction. In this field, there are different strategies used to study natural language. (Spitkovsky, Alshawi, & Jurafsky, 2012) discussed the main characteristics of this method. A typical problem to avoid during induction phase is biasing, to achieve the goal researchers filtered the texts. Tokenization granularity have to be well administered as in (Spitkovsky, Alshawi, & Jurafsky, 2012) to obtain a good grammar induction.
The grammar induction method I was looking for is more similar to a motifs or recurring patterns identification as in (Kiran, Shang, Toyoda, & Kitsuregawa, 2015) (Senin, et al., 2014) (Li, Lin, & Oates, 2012) (Kamiya, Kusumoto, & Inoue, 2002) (Lakhotia, Dalla Preda, & Giacobazzi, 2013). The recurring pattern, identification uses two algorithms frequently: Re-Pair (Moffat & N. Jesper Larsson, 1999) and Sequitur (Nevill-Manning & Witten, 1997). To have a good control of the induction process, and to modify when required by the procedure, in our analysis we used Re-Pair.
The last step in my procedure requires strategies to analyse the inducted grammar rules. Since rules are essentially strings, the first methods explored was the computation of the edit distance (or Levenshtein distance (Levenshtein, 1996)) between the inducted rules. However, edit distance has a structure too simple to compare grammar rules, so I tried Jaro-Winkler distance (Winkler, 1990), and other strings distances or string similarity algorithms (Cohen, Ravikumar, & Fienberg, 2003). The algorithms I adopted were Jaro-Winkler and Jaccard distance (computed using n-grams).

3.2 Characterization of Text Fragments

Forensic analysis use data carving tools to recover file from unallocated space, these tools generate a large quantity of textual artifacts. Files that have header and footer are simple to recognize. However, there are a lot of fragment without a specific recognizable format, the only visible characteristic is the textual characters contained. A big percentage of these file fragments are useless, they contain only padding pattern or similar text. Nonetheless, a small fraction of them still contains valuable information. It is useful to

[45] N-gram is an ordered sequence of n symbols.

find a fast procedure to rank these textual artifacts, in order to find remarkable files, and analyse them first. This chapter presents how we use entropy, di-grams joint entropy, frequency distribution, and grammar based attributes as metric to identify interesting items.

This work presents a procedure that applies a sequence of filters based on entropy, di-grams entropy, frequency distribution, and grammar based attributes to identify interesting fragments.

After a pre-selection step, file fragments are lexically analysed to extract a sequence of tokens to make it simpler to identify structured data. A typical way to recognize these structures is the identification of recurring patterns of tokens through grammar induction: each fragment is associated with a set of grammar rules that can be used to compare different fragments for similarities that can be used to rank the fragments.

Our proposed procedure has four steps, each of which investigates a different aspect of the fragments. The first step select the artifacts using parameters as entropy, byte frequency distribution, n-grams entropy and n-grams frequency distribution. Second step transforms selected fragments in a sequence of tokens. Third step induces a grammar from the tokenized fragments. The last step clusters fragments using similarity between the inducted grammars.

As stated in (Penrose, Macfarlane, & Buchanan, 2013), methods of classification fall into three broad categories: direct comparison of byte frequency distributions, statistical analysis of byte frequency distributions and specialised approaches, which rely on knowledge of particular file formats. The proposed method can be classifies as a specialised approaches, the knowledge of a file format is inducted from fragments.

(Moody & Erbacher, 2008) paper asserts that file formats has a specific set of statistics map over a file, each new version of the file format (for example Microsoft Word or Excel) change the internal structure, and statistic characteristics. The statistic distributions must be recomputed for any different file type. Grammar inducts the structure of new file formats, it is not needed a previous knowledge to work.

(Hall & Davis, 2006) proposed entropy and compression ratio use for file type classification. They computed the value of these parameters using sliding windows of fixed size. This approach can detect different entropy zone, creating a fragment fingerprint. They also proposed to weight entropy using the number of symbols in the window. This technique extracts feature from binary files and multimedia using, but is unsuitable for compressed or encrypted fragments.

The Byte Frequency Distribution (BFD) represents the distribution of each byte (from 0x00 to 0xff) in a file. In text fragments, bytes take values from 0x00 to 0x7f. Different type of file can have very similar BFD. If we compute very similar BFD, this information is not meaningful. Using BFD, we can discover different distribution of characters.

We can try to cluster elements using a distance Euclidean (or cosine similarity) as a metrics and the BFD of each file as a vector of attributes (space dimension 255). However, file of different type sometimes has very similar distribution, so clustering using only BFD will result in a poor efficiency in file type discrimination.

In (McDaniel & Heydari, 2003) Byte Frequency Cross-correlation (BFC) is used to analyse the relationship between byte distributions in the same file. While (Veenman, 2007) uses the frequency distribution of two particular byte values, 60 (<) and 62 (>) to identify HTML file format. The use of BFD and BFC need a previous knowledge of fingerprints to

identify the file format quickly. (Veenman, 2007) presents an application of statistical learning, using as features the byte frequency distribution, the entropy and the Kolmogorov complexity.
N-grams are groups of n consecutive characters or bytes, for example, di-grams (2-grams) are groups 2 consecutive elements, and so on. The main attribute used to study n-grams is the Shannon entropy or information entropy that is defined as the minimum number of bit needed to encode a message.
Equation 1 - Entropy

$$H(x) \ = \sum_{i} P(x_i) log_2 P(x_i)$$

A file is composed of a sequence of bytes and each byte is represented has a sequence of eight bit, therefore entropy can range from [0 to 8) ($log_2 256 = 8$). If a file as a low quantity of information, it shows a low entropy. Values of entropy less than 3 is *low*. If we have an entropy equal to 3 then the max number of different symbols in the text fragment is less than 8 (2^3). A low number of symbols indicate files containing only padding or similar sequence, a file like this in not interesting in forensic analyses and further investigations over these texts are useless. Medium entropy values are from 3 to 6 (from 8 to 6 symbols), these files fragments are part of document, mail, web pages, navigation caches. These files are very important. Data carving activities produce many files like these, their classification is important for investigation activities, and these files require further analysis.
High-entropy files have an entropy value greater than 6, it is a value typical of compressed or encrypted files, and they are important files like image, videos, and office documents.

1-gram entropy is invariant with respect to permutation of symbols in a file, a text has the same entropy of the ordered set of its characters, and the same limit is experienced using character frequency distribution. The following texts show the problem, the Text 1 and Text 2 has the same length (493 bytes), the same entropy (4.37017666646), the same BFD, and using a k-means clustering algorithm (calculated over BFD) will be clustered together.

> You are about to embark upon the Great Crusade, toward which we have striven these many months. The eyes of the world are upon you. The hopes and prayers of liberty-loving people everywhere march with you. In company with our brave Allies and brothers-in-arms on other Fronts, you will bring about the destruction of the German war machine, the elimination of Nazi tyranny over the oppressed peoples of Europe, and security for ourselves in a free world. —Eisenhower, Letter to Allied Forces

Text 1

> ,,,,,---
> ....AACEEFFGGILNTTYaaaaaaaaaaaaaaaaaaaaaaaabbbbbbcccccccddddddddddeeeeeeeeeeeeeeeeee
> eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeffffffgghhhhhhhhhhhhhhhhhhhhhiiiiiiiiiiiiiiiiiikllllllllllllllmm
> mmmmmmmnnnnnnnnnnnnnnnnnnnnnnnnnoooooooooooooooooooooooooooooooooooooopppppppppppprr
> rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrsssssssssssssssssssttttttttttttttttttttttttttttuuuuuuuuuuuuuvvvvvv

Text 2

Observing the two texts, we can detect how sequences of characters are distributed. Each language has its own distribution, the same happens for programming language and web pages.
2-grams (di-grams) entropy reveals text with a complexity in characters permutations. Natural language texts have a typical distribution of di-grams. Observing results in a little sample, we can make some hypothesis about the language and the file format. In uniform sequence of character, where repetitions are common, difference from 1-grams entropy and 2-grams entropy is low. While for text in natural language, the difference is in a range from 0.8 to 2.0 and for marked-up language can reach 5.0. For compressed or encrypted files is over 4.0.

3.3 Mining of Fragments

The data mining process (
Figure 6 - Classification) starts from the file fragments, it computes a set feature for each file fragment, and then it acts a classification.
Features or attributes

Equation 2 - Features/attributes

$$X = [x_1, x_2, \ldots, x_n]$$

Equation 3 - Classes

$$Y = \begin{Bmatrix} C_1 \\ \cdots \\ C_m \end{Bmatrix}$$

Equation 4 - Classification process

$$Y = F(X)$$

The data mining techniques define the $F(X)$ according to the data to be analysed. The classification is not an exact process, but it gives an estimation of the conditional probability of the class when an input X is given $P(Class \mid X)$.
The process of inferring the function $Y = F(X)$ can be supervised or unsupervised. The supervised learning requires a set of inputs X for which the class is known. While the

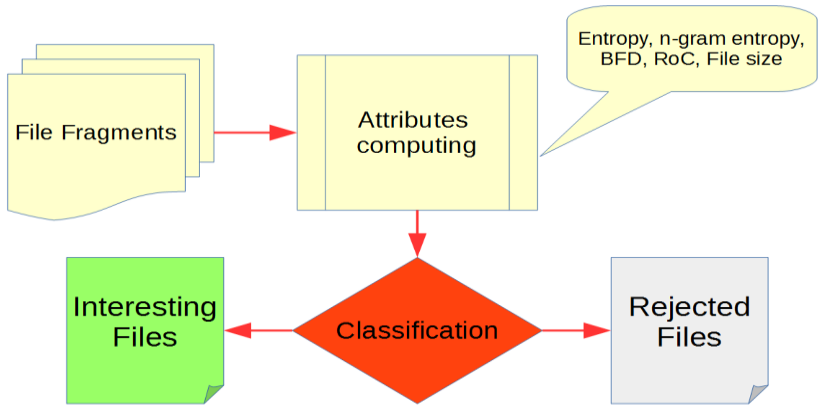


Figure 6 - Classification

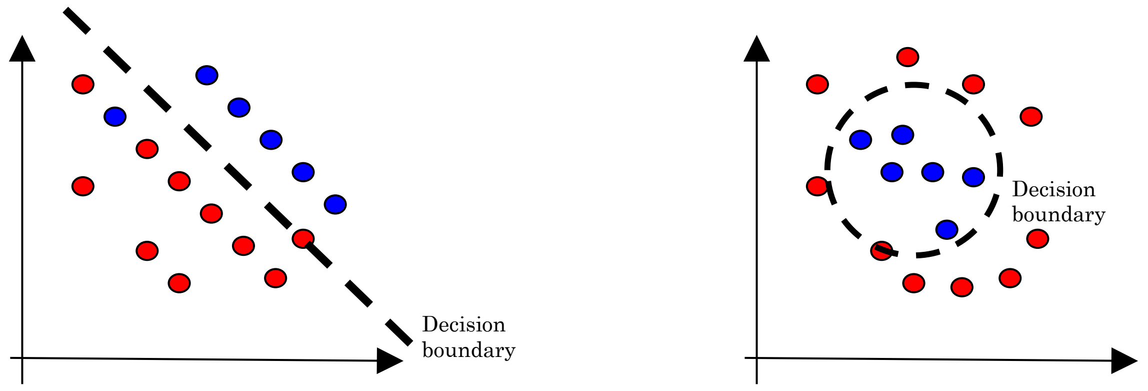


Figure 7 - Decision boundary

unsupervised learning process operates only on input data, discovering automatically the decision boundaries between classes (Figure 7 - Decision boundary).

Considering BFD analysis of textual fragments, its features are the frequencies of the different bytes, there are 256 features, and this implies a space dimension of 256. The data mining techniques suggest reducing the space dimensionality to the feature really correlated to the classes.

To reduce space dimensions we can use simple techniques as:

- eliminate columns relative to attributes with a relative frequency under a threshold;
- group columns with similar attributes for example letter, number, special character and punctuations;

The features selection acts a space dimensions reduction, it can be done in a supervised mode, this procedure requires the definition of a well-composed training set, using the training set the feature have to be automatically analysed in order to define the importance of each particular feature or their combination in the classification process. Only the features with the right level of importance must be considered in the analysis.

Unsupervised method can use correlation between variable, if two variables are highly correlated, they carry the same information, and we can discard one of them. To select between the features, the one with high variance is preferred. Another method to reduce space dimension is the Principal Component Analysis (PCA), this method acts a orthogonal space transformation, in order to detect the transformation that give a new space where dimensions are uncorrelated, the new dimension are called principal component. The method gives as results the new components in order of importance; the first component has the largest variance and is the most important.

3.4 Clustering

Clustering is a data mining unsupervised technique that groups input elements, according to an association rule. This technique divides input data in a number of homogeneous groups according to criteria of internal similarity or external separation. That means elements in the same group are very similar, and element pertaining to different groups are very different and there is separation between groups. This technique works efficiently

if the input data has a globular space distribution, with dense regions that correspond to the final groups. The main clustering technique are the density based.
The number of clusters is the most important parameter to define or to detect. This parameter can be set according to a design criteria, or can be detected trying different solutions and evaluating the resulting clusters.
The density based clustering methods use the distance concept, in order to compute parameters, and the distance used in clustering algorithms is required to be a valid metric.

K-means is an unsupervised clustering algorithm that operates well in convex and isotropic clusters. The algorithm takes as input a data set of N vectors. Each observation is an n-dimensional feature vector. The output result is a set of k groups, each group has a *centroid*, and each vector of the data set is assigned to a group minimizing the within-cluster sum of square error (distance from each vector to group centroid). The number of clusters k is required to operate. The K-means scales well in large data set.
K-means procedure

1. Fix centroids randomly, selecting within member of dataset;
2. Assign each data-sample to a cluster, using minimum Euclidian distance criterion;
3. Compute the centroid of each cluster using mean of element in each cluster;
4. Fix centroids computed in "step 3";
5. Repeat "step 2" to "step 4" until centroids converge.

Consideration:

- Centroids are not guaranteed to converge in a global minimum (local minimum convergence problem);
- Different initial centroids configuration can take a different results;
- How can identify if a data set is convex and isotropic?;
- Manage poorly noisy data and outliers.

To evaluate clustering performance we use "silhouette coefficient". In unsupervised clustering algorithms, this step is very important to understand if a result is better than another is. This coefficient operates without knowledge of true classification of examples. The silhouette coefficient evaluate mean cohesiveness inside the same cluster, and dissimilarity to other cluster.

Equation 5 - Silhouette coefficient

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \qquad s(i) \in [-1, +1]$$

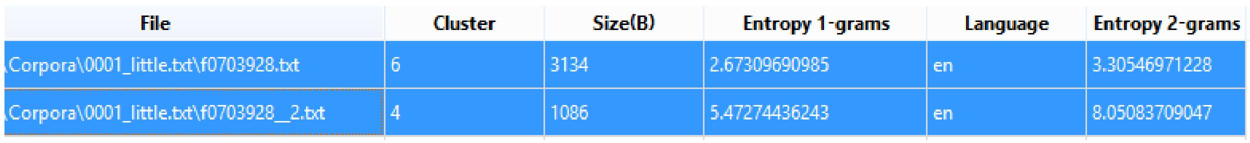
Where:
a(i) is the average dissimilarity of sample *i* with element of his cluster;
b(i) is the lowest average dissimilarity of sample i with element of all other clusters;
s(i) near to 1 identifies a good clustering result, while -1 indicates a bad result.
To estimate the right number of clusters we can compute Mean Silhouette Coefficient. That algorithm compares tightness and separation of points in clusters, tightness with points of the same cluster and separation with points of other clusters.

The procedure to identify the right k for a particular analysis is:

1. Extract a subset;
2. Fix starting number of clusters n-start and the maximum n-stop,
   Set k = n-start;
3. Compute K-means and Silhouette Coefficient;
4. Increase k = k++;
5. if k<=k-stop then step_3
6. Find better Silhouette Coefficient and fix k
7. Load full dataset
8. Compute K-means using the best k;
9. Verify result using Silhouette Coefficient;

Clustering algorithms apply to 1-grams (BFD) and 2-grams frequency distribution. In the case of 1-grams, there are 256 maximum element instances, text files usually contain less than 128 different 1-grams that have frequency greater than zero, and clusters analysis can be restricted to these elements. Clustering of 2-grams has a maximum number of 65536 elements, but in real case the attributes greater than zero are less than 10.000. The data need to be pre-elaborated in order to act a features selection.
Analysing small file size the 2-grams table is a very sparse matrix, 2-grams can be up to 65536, but for little files (size of around 1 kB), and the number of di-grams is much smaller. The same happens analysing natural language text, where di-grams are language dependent, and only a limited number of combination are possible.
In a single device, the number of text fragment files, that can be recovered, can be greater than 10.000, and for very big devices, the number of text fragment files can be over 100.000. Clustering a number of files so big requires a lot of time and computing power. If standard matrix structures (dense matrix) are used, a huge quantity of memory is required. We need to use sparse matrix to develop a program capable to compute this task in a limited time and using a limited quantity of RAM memory.
Python has a set of libraries called SciKit (scikit.learn) that implements learning algorithms using sparse matrices. To represent a sparse matrix structure, it uses list of list (scipy.sparse.lil_matrix). The *sklearn.Cluster.MiniBatchKMeans* computes K-means operating on sparse matrix.
The file fragments need to pre-elaboration before using data mining techniques. The analysis of fragments without selection and filtering, gives a clusterization result predictable. The clusterization groups points around centroids, minimizing the total Euclidean distance. The fragments with low entropy (ex: padding sequence) have low number of di-grams with high relative frequencies. This implicates that centroids drift towards zone where different type of padding sequences are located. The other files groups around a single centroid, and this happens because the number of di-grams is high in files with a medium level of entropy, this implies that the relative frequency of di-grams is low. This proves the importance of files filtering before clusterization, using clusterization to separate low energy files is an expensive procedure in term of time and computing power. Interesting files are all in the same cluster, and in this cluster contains very different file formats (html, word, rtf, temporary cache) but with a large number of different di-grams. Without reducing the number of files, clustering requires a computing power so high that only computing model like MapReduce [Appendix-MapReduce] can carry out this task.

Fragments classification based on fragment entropy differs if we take into account "separators" characters like the blank space or the newline, or if we ignore them. If we take into account a large number of consecutive separators, entropy and BFD are altered, for example if we have two similar file, but at the end of the first there is a long padding sequence made of blank characters, then clustering and comparison will likely put the files in different classes. As the number of separators in a file fragment is not usually an important attribute, we suggest reducing separators to single instance.
An example can make it clear:
Table 1 - Entropy and file cleaning



File	Cluster	Size(B)	Entropy 1-grams	Language	Entropy 2-grams
\Corpora\0001_little.txt\f0703928.txt	6	3134	2.67309690985	en	3.30546971228
\Corpora\0001_little.txt\f0703928__2.txt	4	1086	5.47274436243	en	8.05083709047

The two files f0703928.txt and f0703928_2.txt differ only for 2048 spaces added at the end of the first file (f0703928.txt). If this character is a separator, we divert from correct computation of file attributes. A big number of instance of the same character cut the entropy, and this mislead file filtering when we cut low entropy files, before clustering.

This procedure requires file previously cleaned, to avoid errors. The sequence of action as showed in the table (Table 2 - Step for file filtering) is: first clean files from repetitive sequence of the same character, then remove low 1-gram entropy files, and last remove 2-grams entropy files.

Table 2 - Step for file filtering

Order	Description	Action
1	Multiple consecutive separators	Remove from file
2	Low entropy calculated 1-gram	Discard file
2	Low entropy calculated 2-gram	Discard file

N-gram analysis is common in language detection (Fitzgerald, Mathews, Morris, & Zhulyn, 2012) (Brown, 2012). Each natural language has its characteristic n-gram distribution, programming language and web pages has a distinctive n-gram fingerprint. If a text is long enough, we can infer some other parameters. Using table of "letter" frequency we can classify the language. If we calculate di-grams frequency, identification is more accurate. The Python language has a library (langdetect 1.0.1) based on analysis of frequency of single letter, 2-grams, 3-grams. We can observe that it detects correctly simple plain, but files formatted as rtf or html/xml are not accurately identified. To operate correctly a language detector needs to discard all non-alphabetical character and replace them with a standard separator and to remove all keywords of the mark-up language. If we consider a JavaScript or an xml file, the distribution of n-grams mostly depends on the names of the variable or attributes. For these reasons, it is impossible to compute a typical distribution of n-grams related to a programming language. To detect programming language, the best way is to verify if the text fit the syntax of the programming language itself.

3.5 The problem

Entropy, frequencies distributions and n-grams analysis can help to recognize interesting fragments, but many fragments that show the same values of those parameters have very different content. For example, a JavaScript fragment that use long function names and a fragment of English text show similar parameters. Another problem arises when different formats are present in the same fragment, this happens when a fragment coming from unallocated space of a disk is composed by different original files, or the fragment is part of a pdf file or an e-mail. When this happens, a human being can quickly distinguish this peculiarity.
The problem of recognize file formats is addressed by Roussev (Roussev & Quates, 2013), he underlines that "specialized methods for each file type is needed, to make progress in field of file type recognition. Most forensic practitioners can identify a file type looking at a fragment; we need to transfer this skill to automatic tools". However, how a practitioner can identify a file by simply looking at it? How we can replicate this ability?
What a human does is to observe a scene and split it into its elements, and then he tries to interrelate the observed elements and look for something he knows from previous his experiences.
In computer science, grammar analysis use a similar process (splitting the scene into its elements), the tokenization or lexical analysis phase of a grammar analysis splits a text in its lexical elements (or tokens). The process that "interrelate the observed elements" is similar to semantic analysis. The study of a fragment using a syntax analyser requires a specialized tokenizer and parser for each language or format known, and this is a time consuming process. In my research I identified a small subset of tokens common to many formats, and after the tokenization, I have analysed the resulting tokens sequence using grammar inductions obtaining a set of rules. I analysed the rules using clustering techniques, the metric used, was a strings distance.

Each file has a characteristic format, a set of rules that define its organization of bit/bytes in a computer file. Specifications of a file format, define the encoding of a file, and a set of rules used to define organization. Specifications also define application of a format and if it can act as a container for different type of contents. To identify a file format there are different convention: filename extension, internal metadata, file header, magic numbers (signatures). However, in our forensic analysis we have only a fragment, we do not dispose of sufficient elements to detect format using conventional methods. The internal structure of a file can be described using a grammar (Ullman, Aho, Sethi, & Lam, 2006), and so we have to use lexical and syntactical analysis to extract element useful for the analysis.
It is important to extract the rules pertaining to the grammar of a file, also for little fragment, to be able to separate format from the contents. We can discover element of information in little fragment, for example coordinates left in a temporary file looking at an on-line map.

A fragment is a piece of text that contains elements coming from one or more vocabulary. For example, an html text is a sum of the two vocabulary, "natural language" plus "hypertext mark-up language".

$V_{NL} = \{w_1, w_2, \ldots, w_n\}$	Vocabulary of natural language
$V_{ML} = \{t_1, t_2, \ldots, t_m\}$	Vocabulary of mark-up language
$V_{PL} = \{p_1, p_2, \ldots, p_o\}$	Vocabulary of PDF language

n,m,o number of terms in the vocabulary

The three vocabulary have a non-null intersection, as some html or PDF tags are also English words.

Equation 6 - Not empty intersection Natural language Markup language

$$V_{NL} \cap V_{ML} \neq \emptyset$$

Equation 7 - Not empty intersection PDF language Natural language

$$V_{NL} \cap V_{PL} \neq \emptyset$$

In the example (Figure 8 - Html fragment example) we can identify the tags "head", "title", "body", that are also common English words. The same characteristic is present in many text fragment type (Figure 9 - PDF fragment example). The difference between the occurrence of these words in a natural language and in a mark-up language is the semantic form.

```
<!DOCTYPE HTML>
<html>
        <head>
                <title> …. </title>
        </head>

        <body>
                ….
        </body>
</html>
```

Figure 8 - Html fragment example

```
21 0 obj
<< /Type /Page
   /Parent 1 0 R
   /MediaBox [ 0 0 611.999983 791.999983 ]
   /Contents 3 0 R
   /Group <<
      /Type /Group
      /S /Transparency
      /I true
      /CS /DeviceRGB
   >>
   /Resources 2 0 R
>>
endobj
```

Figure 9 - PDF fragment example

The analysis of a fragment requires the identification of the format. An important task is to find the file type, to which the fragment originally belonged.

3.6 Grammar

A grammar is a set of rules that used to generate or verify strings of a formal language. The alphabet of a language is composed of tokens. The rules define how to use tokens to create or validate a statement. The rules define the syntax of the language. The rules are composed of one or more productions. A grammar is not related to the meaning of a string. The definition of a context-free grammar G requires the following elements:

Equation 8 - Definition of a Grammar

$$G = (N, \Sigma, P, S)$$

S is the start symbol ($S \in N$);
N is a finite set of non-terminal symbol;
Σ is a finite set of terminal symbol;
P is a finite set of production rules;

The set of production rules specify how terminal (Σ) and non terminal (N) symbols can be combined. The terminal symbols (that form Σ) are commonly called tokens.
A language $L(G)$is composed of all the sentences derivable from the grammar G.
We also define $V = \Sigma \cup N$ and $\Sigma \cap N = \emptyset$
The grammar in a fragment can be classified according to "Chomsky Hierarchy"

Type 0 (free or unrestricted grammar)

Production are of the form $u \rightarrow v$, u, v are strings of symbols in $V$and the only restriction is that the left-hand side is not empty (ε), these language are recognized by Turing Machines;

Type 1 (context-sensitive grammar)

Production are of the form $uXw \rightarrow uvw$, u, v, w are strings of symbols in V

$v \neq \varepsilon, X \in N$
These languages are recognized by linearly-bounded automata (subclass of Turing Machines);

Type 2 (context-free grammar, CFG)
Production are of the form $X \rightarrow v$, is a string of symbols in V, $X \in N$
Pushdown automata (PDA are finite-state automata that can push and pop symbols in a stack) recognize these languages;

Type 3 (regular grammar)
Production are of the form $X \rightarrow a$, $X \rightarrow aY$, $X \rightarrow \varepsilon$, $X, Y \in N$, $a \in \Sigma$
Deterministic finite-state automata (PDA are finite-state automata that can push and pop symbols in a stack) recognize these languages;
Type 2 and 3 grammar has very efficient algorithms for their handling. While is more difficult to handle efficiently Type 1 grammar.

Analysing a fragment there is no information about the context and many different format can be mixed together. This type of grammars are hard to handle efficiently. Full grammar induction for each fragment is impossible, we will use grammar tools to extract a set of rules from each fragment, and these rules will explains how data is organized in the files. The number of rules, their structure and their similarity to already known rules will be used for file analysis.

To do a grammar analysis of an input sequence two steps are needed a lexical analyser (lexer or tokenizer) and a syntax analyser (parser) (Figure 10 - Lexer and Parser). Lexical analyser extracts from an input stream a sequence of tokens. Tokens are words in natural language, tags in a mark-up language, keywords in a programming language, but also punctuations and special characters.

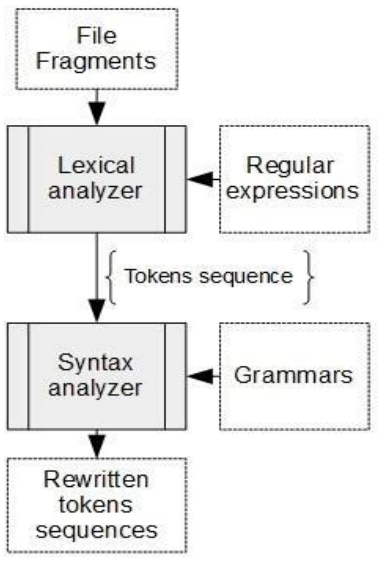


Figure 10 - Lexer and Parser

The syntax analyser (parser) takes the tokens stream as input, it applies grammar productions and it produces an output sequence containing the syntactical constructs recognized by the syntax rules. This process is commonly used to transform a programming language in an executable or in a byte code. The purpose is, to use the same tools to transform a text fragment in a sequence of functional tokens, and then use grammar tools to identify complex sequence according to well-known file format types, or to group (cluster) fragments using the distance between their grammars
To accomplish this task we are going to use a LALR(1) grammar parser, as implemented in LEX and YACC, these tools are well implemented in Python library called PLY(Python Lex and Yacc) (Beazley, 2016).

The lexical analyser transforms an input sequence of symbols in a set of tokens. We used two lexical analyser techniques, first was the "big tokenizer", the second was the "reduced tokenizer".
The "big tokenizer" was capable to identify more than 350 tokens; these tokens came from a large number of text formats. The concept of this lexer was that, more tokens we identify, more different file formats we can recognize.

```
kw = ('DOCTYPE', 'EOF', 'PDF', 'abbr', 'abstract', 'acronym', 'add',
'address',
…
'h1', 'h2', 'h3', 'h4','h5', 'h6', 'hasownproperty', 'having', 'head',
'header', ...
'void', 'volatile', 'wbr', 'when', 'where', 'while', 'with', 'xref', 'yield'
)
```

Code 1 - Tokenizer keywords

The result of this lexer was a series of tokens with low frequency. This type of result is optimal for further investigations, using a series of customized grammar, one for each format. Using an error tolerant parser is possible to fit rules of format-oriented grammar. However, to do this analysis we must collect a big number of grammars, and requires that the formats of the fragments are known, and that there is a sufficient number of tokens. The process to parse token files using many different grammars is very heavy and slow.
The "reduced tokenizer" has a token set limited to 27 elements. The basic idea of this tokenizer is that, to recognize a big number of formats is not important to know each keyword, but we can limit the analysis to the sequence of special characters and special elements.

Special characters:

```
'LEFT_PARENTHESIS', 'RIGHT_PARENTHESIS', 'DOUBLE_LESS_THAN',
'DOUBLE_GREATER_THAN', 'LEFT_SQUARE_BRACKET', 'RIGHT_SQUARE_BRACKET',
'LEFT_CURLY_BRACKET', 'RIGHT_CURLY_BRACKET', 'LESS_THAN', 'GREATER_THAN',
'DOUBLE_MINUS','MINUS', 'DOT', 'COMMA', 'SEMICOLON', 'BACKSLASH', 'SLASH',
'COLON', 'EQUAL', 'NEWLINE'
```

Special elements:

```
'EMAIL', 'URL',  'LITERAL_STR', 'HEX_STR', 'NUMERIC_REAL', 'NUMERIC_INTEGER',
'NAME_ID', 'EMAIL_BOUNDARY'
```

The special characters are can be identified using a simple lexical analyser, while the special elements require a syntactical analyser. The reduced tokenizer produce a sequence of tokens, simpler to handle. Also for little fragment extracts substantial sequence.
Analysing the following fragment we can see the power of a tokenizer, and how the token frequency distribution can help to group similar file format without the previous knowledge of the format.

```
["m",[9,272,177],234095901],["m",[9,273,177],234029070],["m",[9,274,177],234094705],["m",[9,275,1
77],234094705],["m",[9,276,177],234091759],["m",[9,277,177],234087891],["m",[9,278,177],23405494
9],["m",[9,279,177],234092047],
```

After a lexer

```
LEFT_SQUARE_BRACKET LITERAL_STR COMMA LEFT_SQUARE_BRACKET NUMERIC_INTEGER
COMMA NUMERIC_INTEGER COMMA NUMERIC_INTEGER COMMA RIGHT_SQUARE_BRACKET COMMA
NUMERIC_INTEGER RIGHT_SQUARE_BRACKET …
```

After a parser

```
… LEFT_SQUARE_BRACKET NUMERIC_INTEGER COMMA NUMERIC_INTEGER COMMA
NUMERIC_INTEGER RIGHT_SQUARE_BRACKET …  --->  ARRAY
```

The final fragment can be rewritten as:

```
ARRAY COMMA ARRAY COMMA ARRAY …
```

In this way is simpler to look for identical file format in a set of fragments. Calculating the tokens frequency distribution in text fragments and using clustering techniques, we can separate file that shows a different fingerprint. Another machine learning technique exploitable is SVM, learning parameter in a configuration one-against-all. After learning tool training, we can use the parameters to separate a fixed format in a set of fragments.
A previous example used to present limits of byte entropy, can now be re-analysed using grammar tools.
The power of grammar tools in the analysis of little fragments is clear (Table 3 - Grammar process). The fragments that were indistinguishable using entropy analysis or byte frequency distribution are clearly different. This type of analysis reduce the space needed to store information about the document, but it preserve all the syntactical information about the file format, while analysis based on n-grams distribution miss all information about the text structure.

The grammar tools can recognize a file format, if we have a whole file, and a complete grammar. In this paragraph, a specific format is analysed, in order to write a grammar that verify and identify the file format. To verify if a file has a specific format, a file hat to fit the grammar for that specific format. To identify format using a grammar, a file need to be parsed using a set of grammars in order to see which grammar fits. The first problem of this type of approach is that we need a whole file, not a fragment, and the second problem is that we need the complete grammar for all formats that we want to recognize, the grammar are hard to write because there are many undocumented formats. For example many file fragments that contains temporary file data have unknown formats.

Table 3 - Grammar process

Text	Tokens	Grammar Rules
You are about to embark upon the Great Crusade, toward which we have striven these many months. The eyes of the world are upon you. The hopes and prayers of liberty-loving people everywhere march with you. In company with our brave Allies and brothers-in-arms on other Fronts, you will bring about the destruction of the German war machine, the elimination of Nazi tyranny over the oppressed peoples of Europe, and security for ourselves in a free world. — Eisenhower, Letter to Allied Forces.�	TAB, NAMEID, NAMEID, NAMEID, …,COMMA, NAMEID, NAMEID, NAMEID, …, DOT, NAMEID, NAMEID, NAMEID, …, DOT, … MINUS, MINUS, MINUS , NAMEID, COMMA NAMEID, NAMEID, NAMEID, NAMEID, DOT,	phrase, phrase, phrase, … phrase ---> text
,,,,,--- ....AACEEFFGGILNTTYaaaaaaa aaaaaaaaaaaaaaaaaabbbbbbbccc ccccddddddddddeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeeeeeeee eeeeeeefffffffgghhhhhhhhhhhhh hhhhhhhhiiiiiiiiiiiiiiiiiiiiikllllllllllll llmmmmmmmmmmnnnnnnnnnnnn nnnnnnnnnnnnnoooooooooooooo oooooooooooooooooooooopppppppp ppppprrrrrrrrrrrrrrrrrrrrrrrrrr rrrrrrrrsssssssssssssssssssssttttttt ttttttttttttttttttttttuuuuuuuuuuu uuuvvvvvvvvwwwwwwwwwwwwyy yyyyyyyyyyz�	SPACE, SPACE, SPACE, …, TAB, TAB, TAB,..., COMMA, COMMA, COMMA, .., MINUS, MINUS, MINUS , DOT,DOT, DOT , NAMEID	none

The study of a specific grammar can be useful, if used to look for a specific format. A fragment cannot fit a full grammar, but using the right too we can extract information that help to estimate how likely a fragment has a specific format. The tools used to implement this grammar handling need to have a good error reporting with many diagnostic information, the way to recover from parsing error and do a resynchronization, a solid ambiguity handling, the way to save and clone the parser status. The cloning and saving of the parser status give the means to test if the grammar fits, but starting from different point in the rule parser.

The format used for this test is the Portable Document Format (PDF). PDF is a typical file type format; many documents are distributed using this file type. In forensic analysis of disk, fragments of PDF files are common. Typical carvers' techniques that use header or footer to detect file type do not identify the correct file type without the first block. Analysing document for Standard ISO 32000 (PDF 32000-1:2008), grammar rules is possible to write a grammar that fit pdf documents.

Lexical rules

```
lex: tokens   = ('trailer', 'startxref', 'xref', 'PDF_HEADER', 'PDF_HEADER_CMT',
'XREF_ENTRY', 'INDIRECTOBJ_OPEN', 'INDIRECTOBJ_CLOSE', 'INDIRECTOBJ_REF',
'STREAM_OPEN', 'STREAM_VALUE', 'NAME', 'BOOLEAN_TRUE', 'BOOLEAN_FALSE',
'LEFT_PARENTHESIS', 'RIGHT_PARENTHESIS', 'DOUBLE_LESS_THAN', 'DOUBLE_GREATER_THAN',
'LEFT_SQUARE_BRACKET', 'RIGHT_SQUARE_BRACKET', 'LESS_THAN', 'GREATER_THAN',
'LITERAL_STR', 'HEX_STR', 'NAME_OBJECT', 'NUMERIC_REAL', 'NUMERIC_INTEGER')

lex: states   = {'paropen': 'exclusive', 'INITIAL': 'inclusive', 'nameobject':
'exclusive', 'stream': 'exclusive', 'lessopen': 'exclusive'}

lex: Adding rule t_paropen_LEFT_PARENTHESIS -> '\(' (state 'paropen')
lex: Adding rule t_paropen_RIGHT_PARENTHESIS -> '\)' (state 'paropen')
lex: Adding rule t_INDIRECTOBJ_REF -> '[0-9]+\ [0-9]+\ R' (state 'INITIAL')
lex: Adding rule t_XREF_ENTRY -> '[0-9]{10}\ [0-9]{5}\ [fn]' (state 'INITIAL')
lex: Adding rule t_PDF_HEADER_CMT -> '%(?!PDF).*' (state 'INITIAL')
lex: Adding rule t_DOUBLE_LESS_THAN -> '<<' (state 'INITIAL')
lex: Adding rule t_DOUBLE_GREATER_THAN -> '>>' (state 'INITIAL')
lex: Adding rule t_INDIRECTOBJ_OPEN -> '[0-9]+\ [0-9]+\ obj' (state 'INITIAL')
lex: Adding rule t_INDIRECTOBJ_CLOSE -> 'endobj' (state 'INITIAL')
lex: Adding rule t_NUMERIC_REAL -> '[-+]?[0-9]+\.[0-9]+' (state 'INITIAL')
lex: Adding rule t_NUMERIC_INTEGER -> '[0-9]+\b' (state 'INITIAL')
lex: Adding rule t_newline -> '\n+' (state 'INITIAL')
lex: Adding rule t_LEFT_PARENTHESIS -> '\(' (state 'INITIAL')
lex: Adding rule t_RIGHT_PARENTHESIS -> '\)' (state 'INITIAL')
lex: Adding rule t_LESS_THAN -> '<' (state 'INITIAL')
lex: Adding rule t_GREATER_THAN -> '>' (state 'INITIAL')
lex: Adding rule t_NAME_OBJECT -> '/' (state 'INITIAL')
lex: Adding rule t_LEFT_SQUARE_BRACKET -> '\[' (state 'INITIAL')
lex: Adding rule t_RIGHT_SQUARE_BRACKET -> '\]' (state 'INITIAL')
lex: Adding rule t_STREAM_OPEN -> 'stream' (state 'INITIAL')
lex: Adding rule t_NAME -> '[a-zA-Z_][a-zA-Z0-9_\-]*' (state 'INITIAL')
lex: Adding rule t_PDF_HEADER -> '%PDF-[1-2]\.[0-9]' (state 'INITIAL')
lex: Adding rule t_BOOLEAN_FALSE -> 'false' (state 'INITIAL')
lex: Adding rule t_BOOLEAN_TRUE -> 'true' (state 'INITIAL')
lex: Adding rule t_nameobject_HEX -> '\#[0-9a-fA-F]{2}' (state 'nameobject')
lex: Adding rule t_nameobject_CHAR -> '[\w\-_]' (state 'nameobject')
lex: Adding rule t_nameobject_WS -> '[\r\n\x20/\(\)<>\[\]]' (state 'nameobject')
lex: Adding rule t_stream_STREAM_VALUE -> 'endstream' (state 'stream')
lex: Adding rule t_lessopen_GREATER_THAN -> '>' (state 'lessopen')
```

Grammar rules

```
Rule 0     S' -> file
Rule 1     file -> header body_el
Rule 2     body_el -> body xref_ trailer_ eof
Rule 3     body_el -> body eof
Rule 4     body_el -> body_el body xref_ trailer_ eof
Rule 5     empty -> <empty>
Rule 6     header -> header PDF_HEADER
Rule 7     header -> header PDF_HEADER_CMT
Rule 8     header -> PDF_HEADER
Rule 9     header -> PDF_HEADER_CMT
Rule 10    body -> body indirect_obj
Rule 11    body -> indirect_obj
Rule 12    body -> empty
Rule 13    trailer_ -> trailer dictionary_obj
Rule 14    xref_ -> xref xref_body
Rule 15    xref_body -> xref_cnt
Rule 16    xref_body -> xref_cnt xref_el
Rule 17    xref_body -> xref_body xref_cnt xref_el
Rule 18    xref_cnt -> NUMERIC_INTEGER NUMERIC_INTEGER
Rule 19    xref_el -> XREF_ENTRY
Rule 20    xref_el -> xref_el XREF_ENTRY
Rule 21    eof -> startxref NUMERIC_INTEGER
Rule 22    pdf -> boolean
Rule 23    pdf -> numeric
Rule 24    pdf -> literal_str
Rule 25    pdf -> hex_str
Rule 26    pdf -> dictionary_obj
Rule 27    pdf -> array_obj
Rule 28    pdf -> stream_obj
Rule 29    pdf -> INDIRECTOBJ_REF
Rule 30    pdf -> NAME
Rule 31    pdf -> LITERAL_STR
Rule 32    pdf -> HEX_STR
Rule 33    pdf -> NAME_OBJECT
Rule 34    boolean -> BOOLEAN_TRUE
Rule 35    boolean -> BOOLEAN_FALSE
Rule 36    numeric -> NUMERIC_REAL
Rule 37    numeric -> NUMERIC_INTEGER
Rule 38    literal_str -> LEFT_PARENTHESIS LITERAL_STR RIGHT_PARENTHESIS
Rule 39    hex_str -> LESS_THAN HEX_STR GREATER_THAN
Rule 40    array_obj -> LEFT_SQUARE_BRACKET array_el RIGHT_SQUARE_BRACKET
Rule 41    array_el -> pdf
Rule 42    array_el -> array_el pdf
Rule 43    dictionary_obj -> DOUBLE_LESS_THAN dictionary_pair DOUBLE_GREATER_THAN
Rule 44    dictionary_obj -> DOUBLE_LESS_THAN DOUBLE_GREATER_THAN
Rule 45    stream_obj -> STREAM_OPEN STREAM_VALUE
Rule 46    name_obj -> NAME_OBJECT
Rule 47    dictionary_pair -> name_obj pdf
Rule 48    dictionary_pair -> dictionary_pair name_obj pdf
Rule 49    indirect_obj -> INDIRECTOBJ_OPEN indirect_obj_el INDIRECTOBJ_CLOSE
Rule 50    indirect_obj_el -> pdf
Rule 51    indirect_obj_el -> indirect_obj_el pdf
```

Using PLY - Python Lex and Yacc that has Pure-Python implementation of an LALR (1) parser, the error tolerant parser has been implemented. The process of writing the rules and the test of resynchronization and error handling procedures are format specific and time consuming. This approach give expected results, and is possible to test if different fragments of the same format are contiguous, testing grammar rules in the junction point. However, we are looking for analysis method that apply to non-specialized format. The next step is to identify if it is possible to generate automatically the grammar rules analysing the content of a fragment.

3.7 Grammar Induction

If a file has not a recognizable format, a further step is required. Our study proceed Analysing the problem of fragments non-recognizable or undocumented format in the fragments. The technique focused on recurring patterns hunting finalized to act a grammar induction. When we find a recurring pattern in a sequence of tokens, this is a

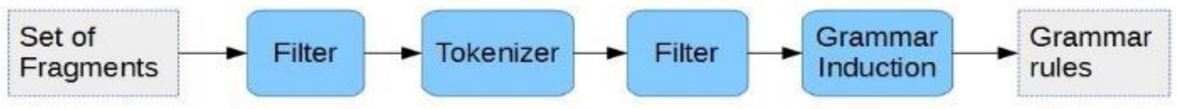


Figure 11 - Grammar induction

good indicator for the presence of a mark-up language, a sentence in natural language, a fragment of configuration file or a fragment of logs.

The problem of recurring patterns discovering is a common problem in fields as time series analysis (Kiran, Shang, Toyoda, & Kitsuregawa, 2015) (Li, Lin, & Oates, 2012) (Senin, et al., 2014) and in genomics. Recurring patterns are called motifs, their primary application was in data compression, but recently the same techniques are used for rules association in data mining. In DNA analysis, motifs are important for their biological significance (Nguyen, K., Tran, D., Wanli Ma, & Sharma, D., 2014).
An efficient way to discover recurring pattern is grammar induction. In (Li, Lin, & Oates, 2012) motifs are discovered using Sequitur algorithm. Sequitur algorithm (Nevill-Manning & Witten, 1997) infers hierarchical structure from a sequence of symbols. Sequitur acts a greedy grammar induction, and inducts a non-optimal (or local optimal) context-free grammar. The Sequitur algorithm elaborates an input sequence of symbols online, applying two conditions:

Di-grams uniqueness - impose that each di-gram in the sequence must be unique, so for any redundancy in the input string a rule is added.

Rule utility - Second condition requires that each rule is used at least once, if a rule is no longer used, it is removed.

Another algorithm, for grammar induction is Re-Pair (Recursive Pairing) (Moffat & N. Jesper Larsson, 1999), this uses an offline procedure. This algorithm iterates over the sequence and substitute the most frequent pair of symbols with a new symbol. The procedure is repeated until no pair of symbols appear twice. An algorithm is online if operates on a flow, and does not need to know all the sequence, while an offline procedure requires full sequence when it starts. In a forensic point of view, this is not a constraint.
The final product of grammar induction (

) is a set of rules and each rule identifies a motif. For each rule, we can define the frequency (number of occurrences) in the fragment.
Some files analysis generate a very large number of rules. This behaviour is typical when we analyse a fragment of code (ex: JavaScript). Analysing multiple files, with the same file format, using grammar induction, we obtain different sets of rules, the rules inducted are not identical but only similar, this is because the inducted grammar is only a local optimal, so rules structure depends on the contest in which the recurrences are found. Also analysing the same file using different grammar induction algorithms, they extract different sets of rules, each grammar inductions algorithms reacts differently to the context.
Analysing different corpora we can notice that using little files or little portion of file, rules generated are simpler to understand, while big files sometimes generate rule, without a meaning for a human being. These rules are effects of the procedures used in grammar induction, these procedures are based on the research of recursive patterns, dealing with

big files the probability to find recursive sequence, not related to the format, grow. In big files, also the characteristics connected to the style of the particular document become recursive, not only the elements of the file format.

Considering that rules, come from grammar induction that depend on the context. We use concept of distance or similarity to compare rules. Rules are strings that we can compare using strings distance metrics. Grammar induction generates, from each file, a set of rules, according to recurring sequences on it. For each rule, we can estimate the coverage of the rule against the file.

Equation 9 - Rule coverage

$$C_{Rx} = \frac{len\ (R_x\ ) * N_{Rx}}{len(F)}$$

Where:
C_{Rx}= coverage acted due to R_x
F = File under analysis
R_x = X^{th} rule extracted from F
N_{Rx} = number of occurrences of R_x if F

Considering two files F1 and F2, the number of rules induced from F1 are *n* and from the F2 are *m*. To identify similarity between fragment files we compare the two sets of induced rules. The results fit in a $n * m$ matrix, where the *n* rows are the rules of the first file (F1), while the *m* columns are the rules of the second file (F2), the elements $d_{i,j}$of the matrix are the Levenshtein distances between rule R_i of the first file and rule R_j of the second file.

Equation 10 - Distance matrix

$$DistanceMatrix = \begin{bmatrix} d_{1,1} & \cdots & d_{1,m} \\ \vdots & \ddots & \vdots \\ d_{n,1} & \cdots & d_{n,m} \end{bmatrix}$$

The coverage values are in a vector:

Equation 11 - Coverage vector file F1

$$C_{F1} = \ [C_{R1}\ C_{R2}\ \cdots\ C_{Rn}]$$

Equation 12 - Coverage vector file F2

$$C_{F2} = \ [C_{R1}\ C_{R2}\ \cdots\ C_{Rm}]$$

To evaluate the distance between the two files, we compute the dot product between distance matrix and the coverage vectors. The result is a scalar that take in consideration distance from each rule, and coverage of each rule.

Equation 13 - Distance between two files

$$Distance_{F1,F2} = \ C_{F1} \cdot Distance\ Matrix\ \cdot\ C'_{F2}$$

This procedure is time/cpu consuming, we try to identify if a better or faster solution exists.

Table 4 - Table Type Styles

Strings		Distance					
		H	L	J	JT	DL	JW
aaaa	abababab	6	4	0.5	1.0	4	0.15
aaaa	bbbb	4	4	1	1.0	4	1.0
aaaa	aaaabbbb	4	4	0.5	0.75	4	0.1
aaaa	bbbbaaaa	8	4	0.5	0.75	4	0.29
abcd	a	3	3	0.75	-	3	0.25
abcd	dddaaabbbccc	12	9	0.0	1.0	9	0.44
abcd	aabcaabbbcdcc	12	9	0.0	0.8	9	0.34
bbba	abab	3	3	0.0	1.0	2	0.28

H – Hamming	L – Levenshtein
J – Jaccard	JT – Jaccard trigrams
DL – Damerau Levenshtein	JW – Jaro- Winkler

String Distance Measures

To select the string distance algorithm to use, we analysed a set of known algorithms, to identify which of this presents desirable characteristics (Table 4 - Table Type Styles). Jaccard and Jaro-Winkler distances range from 0.0 to 1.0, while Hamming Levenshtein distances count the number of operations necessaries to transform a string into another, the range depends on the length of the string under analysis.

Hamming Distance

The Hamming distance is the number of characters different in two strings of the same length; the characters are compared using the corresponding position in the two strings. A generalized version of Hamming distance defines the case of different length, given two strings $s(|s| = m)$, $t(|t| = n)$ if $m > n$, $Ham(s,t) = Ham(s[1...n], t[1...n]) + (m-n)$. This type of measure does not fit ours needs because, grammar induction is sensible to the context, and the rule inducted are not bounded to the position. Moreover, if a string s1 is contained in s2, but s1 is not at the start of string s2, their distance will be the same as s1 and s2 have nothing in common.

Jaccard Distance or Jaccard Similarity Coefficient

The Jaccard coefficient over two set of symbols is calculated as the size of intersection divided by the size of union. If is identical to the Jaccard coefficient is 1, while if and have no symbols in common the distance is 0.

Equation 14 - Jaccard similarity Coefficient

$$J(s_1, s_2) = \frac{|s_1 \cap s_2|}{|s_1 \cup s_2|}$$
$$0 \leq J(s_1, s_2) \leq 1$$

Equation 15 - Jaccard distance

$$DJ(s_1, s_2) = 1 - J(s_1, s_2)$$

The Jaccard distance does not take in consideration the order of the sequence of symbols in the strings and the number of times a symbols is repeated. This is not a desired characteristic for our problem, but better result can be obtained, if long sequence are used. Using 3-grams instead of single symbols the result obtained reaches a good level in identification of similar rules. Tri-gram keeps in consideration the order of the symbols in the sequence. Jaccard tri-grams distance structure perform correctly also in the case of long and repetitive sequence.

Levenshtein Distance

The Levenshtein distance is called edit distance and is the minimum edit operation to transform a string into another (Levenshtein, 1996) (Lavoie & Merlo, E., 2012). Levenshtein distance is a metric. The distance is computed according to the number of some basic string operations, the operations are insert, delete and substitution. In Levenshtein distance, each operation has the same cost.
This type of metrics does not fit ours needs, because all operation have the same weight, the second example shows that a complete different set of tokens give as a result the same distance.

Damerau–Levenshtein Distance

The Damerau-Levenshtein distance (Damerau, 1964) starting from Levenshtein, adds the transposition of two adjacent characters. The Damerau-Levenshtein is a metric. The operation added is justified by the fact that 80% of human misspelling is due to transposition. It is used in Fraud detection and natural language analysis. We are looking for similarity measure of motifs, and transposition addition is not suitable for our purpose.

Jaro-Winkler Distance

Jaro-Winkler is a similarity measure (Winkler, 1990). The values are between (0;1), where 0 means no similarity, while 1 means exact math. Given two string s_1 and s_2, the procedure to compute the similarity (DJ) is the following:

1. Compute m as the number of matching characters;
2. Compute t as half of the number of transpositions.

Equation 16 - Jaro distance

$$DJ = \begin{cases} 0 & if\ m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{|s_1|}\right) & if\ m \neq 0 \end{cases}$$

Matching characters: two characters are matching if they are identical and their distance is less than d, where d is:

Equation 17 - Character distance

$$d = \frac{\max(|s_1|, |s_2|)}{2} - 1$$

Transposition: is the half of the number of matching characters but with different sequence order. Jaro-Winkler use the concept of prefix, used to obtain better prefix in string that match from the beginning.

Equation 18 - Jaro-Winkler distance

$$DW = DJ + (L * p * (1 - DJ))$$

Where

L: length of the common prefix max 4 chars

p: scale factor usually 0.1, max value 0.25

Some of the characteristics feet ours needs, distance of matching character is limited and transposition act a reduction of the similarity. The concept of Jaro-Winkler similarity is closer to what we are looking for.

Rule Analysis

The distance suitable for our purpose are metric are Jaro-Winkler and Jaccard tri-grams, for our need we used the Jaccard 3-gram distance, this metrics give good results, because takes care of the order of symbols in the sequence, and this is fundamental in grammar rules comparing. A second desired characteristic is that, common parts between rules can be located in different point of the rules under comparison.

For example using the procedures presented in the analysis of a series of fragments extracted from a gps navigator. We obtained the result in Table 5 - Rules comparison. The separation is visible and clear (Table 5 - Rules comparison), but the number of file analysed is limited and a statistical significance must be proved using large corpora. This is a brief analysis to validate the path for fragment analysis; this study requires more in-depth research. Analysis of big corpora requires a previous work to obtain better performances.

1st file type	2nd file type	Mean rules distance (C.I. 95%)
Files containing addresses	Files containing addresses	0.465333496 (+/- 0.1132103)
Files containing addresses	Files NOT containing addresses	0,904666667 (+/- 0.03688051)
Files containing configuration	Files NOT containing configuration	0,9728 (+/- 0.02100576)
Files containing configuration	Files containing configuration	0,116402116 (+/- 0.1481194)
Files containing addresses	Files containing configuration	0,968666667 (+/- 0.00474384)

Table 5 - Rules comparison

3.8 Latent Semantic Analysis/Indexing

Latent semantic indexing (Figure 12 - Latent Semantic Analysis) (Dumais, 2004) (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990) is a method for automatic indexing and retrieval. The method discover automatically the semantic structure of a document. This method use the SVD[46] to transform a matrix of terms and documents in new matrix where dimension are orthogonal and dimensions are a linear combination of the previous one. The new dimension are ordered by higher variation. This help to reduce dimensionality of the document analysis problem. The analysis of the documents we find in a storage device under analysis using LSI, creates automatically a representation of the semantic structure of the most frequent type of documents. When a fragment of a file is recovered from the same devices, we can compute the distance to the documents previously analysed and identify similarity to other documents in the system, all the term in the fragments are weighted according to the space transformation obtained using LSI technique.
Inside a fragment, there are format-tags and phrases, before use LSA/LSI (Figure 12 - Latent Semantic Analysis) we need to separate sentences in natural language from mark-up elements, this can be accomplished using grammar. After the grammar analysis, it is possible to identify which elements have a high probability to be sentences in natural language. Using only these parts of the text, we can adopt a Latent Semantic Analysis. This technique can give a "semantic clustering" of natural language fragments.
Latent semantic analysis (LSA) is also known as Latent Semantic Indexing (LSI).

- Each document is represented as a "bag of words", and for each word the frequency in the document is computed;
- Concept are represented as a pattern of word that appear in a document;
- Is assumed that each word as one meaning.

Analysis require stop-word removal. Since stop-words are not important for the meaning.
All words are considered if they appear in more than two sentences.

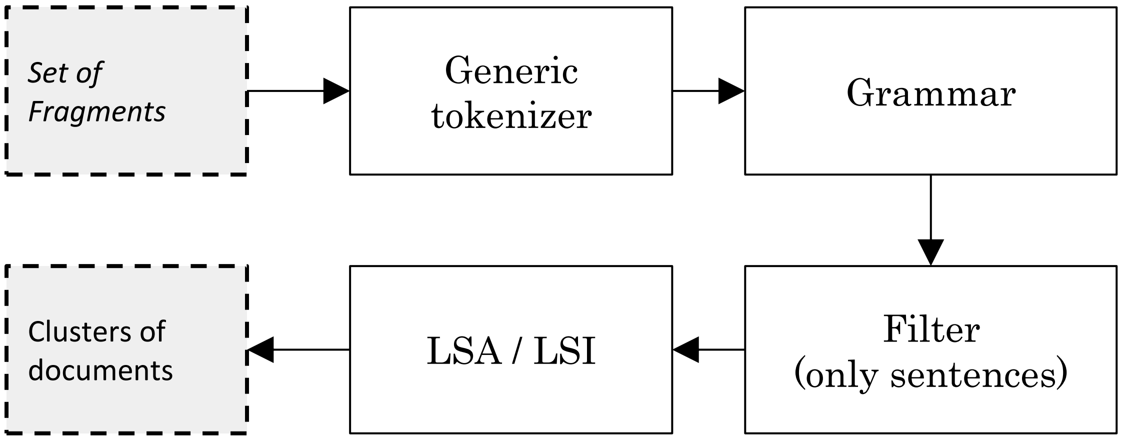


Figure 12 - Latent Semantic Analysis

[46] SVD stands for Singular Value Decomposition. SVD method identify and order the data according to the the dimensions that shows most variance.

TFIDF (term frequency - inverse document frequency) is a method to give more weight to less common word.

Each count is replaced with the formula:

Equation 19 - Term Frequency Inverse Document Frequency

$$TFIDF_{i,j} = \left(\frac{N_{i,j}}{N_{*,j}}\right) * log\left(\frac{D}{D_j}\right)$$

Ni,j = the number of times word i appears in document j (the original cell count).
N*,j = the number of total words in document j (just add the counts in column j).
D = the number of documents (the number of columns).
Di = the number of documents in which word i appears (the number of non-zero columns in row i).

3.9 Comment on Results

This is a path, and the results illustrate that using all the techniques presented here is possible extract information from these fragments. However, we need to implement these procedures in an integrated framework. Performance needs to be improved. Grammar induction algorithms used was Sequitur and RePair, and new technique are now available, that need a depth study. The rule analysis is a distance comparison, with some consideration about coverage of rules against the pertaining fragment. A further work needs depth study about grammar induction and structure of the extracted rules. The paper shows how to extract features; useful for data mining procedure. Next chapter (4 Fragment Classification Evaluation) discusses the tools developed to test analysis studied in this chapter.

4 Fragment Classification Evaluation

The methods exposed in the previous chapter need a scientific evaluation. This chapter identifies strength and weakness of these methods. The evaluation model shows how the new features increase classification accuracy.
Tests are conducted using "Digital Corpora" file repository (Corpora, 2015). The Corpora files are a standardized set of files prepared to make comparable, results obtained from different analysis (Garfinkel S. P., 2009). The reproducibility is a fundamental requirement for any study, the forensic reproducibility is a requirement for evidence analysis presentation in a proceeding. The procedure used to create the test set starts from 200 files randomly extracted from the "Digital Corpora". I limited the study to 4 file formats (rtf, txt, log, js), to create a file set stratified, each file format is equally represented in the test set. Then each file in the file set is fragmented using a fixed length of 4Kbytes. The last step creates the test set randomly selecting 200 fragments that compose the test set. The selection of the fragments used for the test set respects the stratification requirements.
To compare the analysis results I will use the confusion matrix (Table 6 - Generic confusion matrix) and the Accuracy (Equation 20 - Accuracy), Precision (Equation 21 - Precision), Recall (Equation 22 - Recall) and Miss Rate (Equation 23 - Miss Rate) parameters (Powers, 2007).

	Prediction positive	**Prediction negative**	
Condition positive	TP (true positive)	FN (false negative)	CP (tot positive condition)
Condition negative	FP (false positive)	TN (true negative)	CN (tot negative condition)
	PP (tot positive prediction)	PN (tot negative prediction)	1

Table 6 - Generic confusion matrix

Equation 20 - Accuracy

$$\text{Accuracy} = \frac{TP+TN}{Total\ sample}$$

Equation 21 - Precision

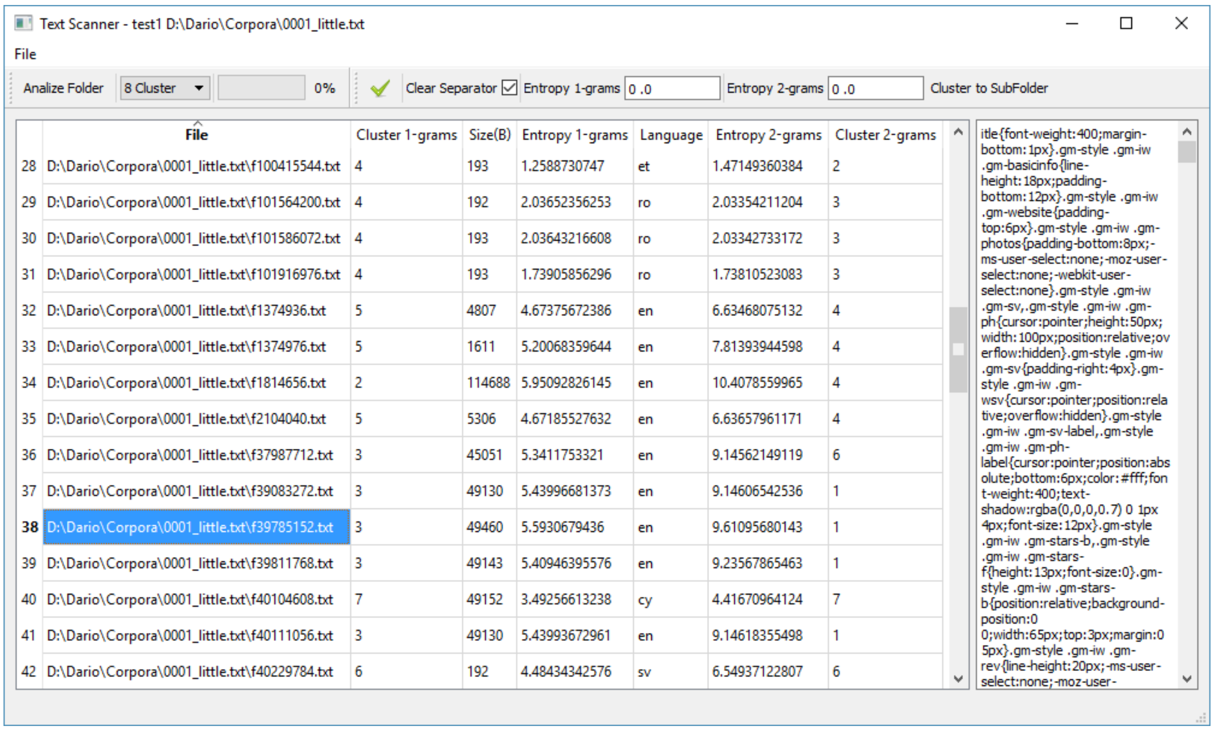
$$\text{Precision}=\frac{TP}{\sum Prediction\ positice}$$

Equation 22 - Recall

$$\text{Recall}=\frac{TP}{\sum Condition\ positive}$$

Equation 23 - Miss Rate

$$\text{Miss rate} = \frac{FN}{\sum Condition\ positive}$$


Text Scanner - test1 D:\Dario\Corpora\0001_little.txt

File

Analize Folder | 8 Cluster | 0% | Clear Separator ☑ | Entropy 1-grams 0 .0 | Entropy 2-grams 0 .0 | Cluster to SubFolder

	File	Cluster 1-grams	Size(B)	Entropy 1-grams	Language	Entropy 2-grams	Cluster 2-grams
28	D:\Dario\Corpora\0001_little.txt\f100415544.txt	4	193	1.2588730747	et	1.47149360384	2
29	D:\Dario\Corpora\0001_little.txt\f101564200.txt	4	192	2.03652356253	ro	2.03354211204	3
30	D:\Dario\Corpora\0001_little.txt\f101586072.txt	4	193	2.03643216608	ro	2.03342733172	3
31	D:\Dario\Corpora\0001_little.txt\f101916976.txt	4	193	1.73905856296	ro	1.73810523083	3
32	D:\Dario\Corpora\0001_little.txt\f1374936.txt	5	4807	4.67375672386	en	6.63468075132	4
33	D:\Dario\Corpora\0001_little.txt\f1374976.txt	5	1611	5.20068359644	en	7.81393944598	4
34	D:\Dario\Corpora\0001_little.txt\f1814656.txt	2	114688	5.95092826145	en	10.4078559965	4
35	D:\Dario\Corpora\0001_little.txt\f2104040.txt	5	5306	4.67185527632	en	6.63657961171	4
36	D:\Dario\Corpora\0001_little.txt\f37987712.txt	3	45051	5.3411753321	en	9.14562149119	6
37	D:\Dario\Corpora\0001_little.txt\f39083272.txt	3	49130	5.43996681373	en	9.14606542536	1
38	D:\Dario\Corpora\0001_little.txt\f39785152.txt	3	49460	5.5930679436	en	9.61095680143	1
39	D:\Dario\Corpora\0001_little.txt\f39811768.txt	3	49143	5.40946395576	en	9.23567865463	1
40	D:\Dario\Corpora\0001_little.txt\f40104608.txt	7	49152	3.49256613238	cy	4.41670964124	7
41	D:\Dario\Corpora\0001_little.txt\f40111056.txt	3	49130	5.43993672961	en	9.14618355498	1
42	D:\Dario\Corpora\0001_little.txt\f40229784.txt	6	192	4.48434342576	sv	6.54937122807	6

```
itle{font-weight:400;margin-
bottom:1px}.gm-style .gm-iw
.gm-basicinfo{line-
height:18px;padding-
bottom:12px}.gm-style .gm-iw
.gm-website{padding-
top:6px}.gm-style .gm-iw .gm-
photos{padding-bottom:8px;-
ms-user-select:none;-moz-user-
select:none;-webkit-user-
select:none}.gm-style .gm-iw
.gm-sv,.gm-style .gm-iw .gm-
ph{cursor:pointer;height:50px;
width:100px;position:relative;ov
erflow:hidden}.gm-style .gm-iw
.gm-sv{padding-right:4px}.gm-
style .gm-iw .gm-
wsv{cursor:pointer;position:rela
tive;overflow:hidden}.gm-style
.gm-iw .gm-sv-label,.gm-style
.gm-iw .gm-ph-
label{cursor:pointer;position:abs
olute;bottom:6px;color:#fff;fon
t-weight:400;text-
shadow:rgba(0,0,0,0.7) 0 1px
4px;font-size:12px}.gm-style
.gm-iw .gm-stars-b,.gm-style
.gm-iw .gm-stars-
f{height:13px;font-size:0}.gm-
style .gm-iw .gm-stars-
b{position:relative;background-
position:0
0;width:65px;top:3px;margin:0
5px}.gm-style .gm-iw .gm-
rev{line-height:20px;-ms-user-
select:none;-moz-user-
```

Figure 13 - SOM software

4.1 N-Grams analysis

The first tool I created was "SOM" (Figure 13 - SOM software). This software uses entropy and cluster analyses based on 1-gram, 2-grams. It computes clusters using k-mean technique. The program allows filtering files using n-grams entropy; it can make the elaboration leaving out separator. Using the clustering outputs, the program may separate files in a different folder, one folder for each cluster. The number of cluster (k) is fixed by the user and the program computes the silhouette coefficient to estimate the clustering process effectiveness.

4.1.1 1-Gram

The 1-gram analysis computes the entropy of the files and the bytes frequency distribution (BFD). The entropy tell us the minimum number of symbols needed to code each text; the information carried by this parameter shows how many bits we need to code the different character in a text. The entropy parameter does not discriminate files type.

File	Cluster 1-grams	Entropy 1-grams
…		
.\Corpora\...\TestEntropy_48_4	3	4,00000
.\Corpora\...\TRI_2005_FL.rtf	3	4,35268
.\Corpora\...\TRI_1987_FL.rtf	3	4,38282
.\Corpora\...\editors.js	2	4,38508
.\Corpora\...\TRI_2005_AR.rtf	3	4,39520
.\Corpora\...\CBP_WQ_2008.rtf	3	4,40209
.\Corpora\...\Adventures of Sherlock Holmes.txt	2	4,40476
.\Corpora\...\TRI_1987_AR.rtf	3	4,41644
.\Corpora\...\How to analyze people on sight.txt	2	4,43429
.\Corpora\...\G8_SCHOOL07.rtf	3	4,43523
.\Corpora\...\Adventures of Huckleberry Finn.txt	2	4,43763
.\Corpora\...\001500.log	2	4,44942
.\Corpora\...\codebook for W1 data.rtf	2	4,45133

Table 7 – Excerpt of 1-grams analysis of different file format

Cluster	Rtf	Log	Js	Txt
0	0%	0	2,04 %	0
1	38,67%	11,76 %	0	4,17 %
2	28,0%	41,18%	97,96%	45,83 %
3	33,33	47,06 %	0	50%
Tot.	100%	100%	100%	100%

Table 8 - File type classification using 1-gram clustering

Cluster 2 RTF	Prediction positive	Prediction negative	
Condition positive	TP = 17,58 %	FN = 27,88 %	45,46
Condition negative	FP = 1,82 %	TN = 52,72 %	54,54
	19,4 %	80,6 %	100 %

Table 9 - Confusion matrix - 1gram clustering analysis - format RTF

Accuracy	Precision	Recall	Miss Rate
0,703	0,906	0,387	0,613

Table 10 - 1-Grams evaluation analysis

The analysis of 1-grams shows property inferable from single symbols analysis and their distribution. In a limited range of entropy (from 4 to 4.45), we can find all different format types (Table 7 – Excerpt of 1-grams analysis of different file format). Different file type has the same number of symbols and the same byte frequency distribution.
The “Cluster 1-gram” compute uses the frequency distribution, of ASCII symbols (from 0x00 to 0xFF), as coordinates. The space under analysis has 256 dimensions. Each file represent a point in a 256 dimensions space. The clustering procedure uses k-means algorithm and Euclidean distance.
Analysing our set of files, the best silhouette coefficient (Equation 5 - Silhouette coefficient) is 0.29145 obtained for k=8. The clustering process considered k in a range from 3 to 8. The silhouette values range from 0.25605 (k=5) to 0.29145 (k=8), this limited range indicates that the parameters considered for the analysis are not correlated to the file format. The number of different format is 4 and for k=4 the silhouette value equals to 0,26976, this is not the best result because of the weak link between file format and byte frequency distribution. The analysis of results considers k=4. The result of the analysis are in next table (Table 8 - File type classification using 1-gram clustering). This result was largely predictable, because there is little correlation between byte distribution and file formats.
The analysis in “Table 9 - Confusion matrix - 1gram clustering analysis - format RTF” supposes that cluster 1 identify rtf format.
The code used in this simple test uses Orange library of function to cluster samples. The k-means algorithm uses an initialization of centroids positions called “init_diversity” (Code 2 - Orange K-means function); this parameter returns a set of centroids where the first one is a data point being the farthest away from the centre of the data. The distance function used is: Euclidean. The code do not use the entire 256 dimension, but to improve performances it limits the analysis to the dimensions different from zero for all the elements. The selected formats contain only textual fragments, therefore the only ASCII code present are limited to printable characters.

```
km = Orange.clustering.kmeans.Clustering(tab_data, clusterNumber ,
initialization=Orange.clustering.kmeans.init_diversity)
```

Code 2 - Orange K-means function

File	Cluster 1-grams	Cluster 2-grams	Entropy 1-grams	Entropy 2-grams	Language
.\TestEntropy_48_4	3	1	4,00000	3,99995	xx
.\TRI_1987_IL.rtf	3	3	4,20879	6,95876	en
.\TRI_2005_IL.rtf	3	3	4,25951	7,16047	en
.\TRI_2005_CA.rtf	3	3	4,26974	7,15304	en
.\TRI_1987_IN.rtf	3	3	4,27373	7,05687	en
.\TRI_1987_GA.rtf	3	3	4,30207	7,09040	en
.\Grimms fairy tales.txt	2	1	4,33682	7,65808	en
.\Frankenstein.txt	2	1	4,34158	7,76652	en
.\Metamorphosis.txt	2	1	4,34346	7,72217	en
.\TRI_1987_CT.rtf	3	3	4,34674	7,10815	en
.\Gullivers Travels.txt	2	1	4,34857	7,74420	en
.\TRI_2005_FL.rtf	3	3	4,35268	7,25549	en
.\TRI_1987_FL.rtf	3	3	4,38282	7,20326	en
.\editors.js	2	1	4,38508	6,31572	ca
.\TRI_2005_AR.rtf	3	3	4,39520	7,33914	en
.\CBP_WQ_2008.rtf	3	1	4,40209	6,31224	en
.\Adventures of Sherlock Holmes.txt	2	1	4,40476	7,85661	en
.\TRI_1987_AR.rtf	3	3	4,41644	7,26016	en
.\How to analyze people on sight.txt	2	1	4,43429	7,91287	en
.\G8_SCHOOL07.rtf	3	2	4,43523	7,52134	en
.\Adventures of Huckleberry Finn.txt	2	1	4,43763	7,86933	en
.\001500.log	2	1	4,44942	6,40302	sl
.\codebook for W1 data.rtf	2	1	4,45133	7,04080	en

Table 11 - Excerpt of di-grams analysis of different file formats

4.1.2 Di-Gram

Di-Grams are combination of two symbols, the possible combination are 65536, in a text, not all combinations are possible, the distribution of di-grams allow recognizing different spoken languages, the analysis test if it may classify different file formats. The "Table 11 - Excerpt of di-grams analysis of different file format" reports all the features computed by SOM code: digrams entropy, digrams frequency distribution and "digrams cluster". The entropy computed using digrams enhance information given by entropy computed on a single symbol.
The entropy computed using di-grams gives little more information to discriminate over file formats. However, in a restricted range of digrams entropy, all formats (rtf, js, txt and log) are present.
The cluster computation made using di-grams distribution give better results if compared with 1-grams. The better Silhouette coefficient considering k in a range 3 to 8, is 0,50059 obtained for k=3, however the number of different format is 4 and for k=4 the silhouette value equals to 0,30130. The analysis of results consider k=4.

Cluster	Rtf	Log	Js	Txt
0	50,67%	5,88%	0%	20,83%
1	30,67%	0%	0%	0
2	14,67%	94,12%	100%	79,17%
3	4%	0	0	0

Table 12- File type classification using di-gram clustering

Cluster 0 RTF	Prediction positive	Prediction negative	
Condition positive	TP = 23,03 %	FN = 22,42 %	45,45
Condition negative	FP = 3,64 %	TN = 50,91 %	54,55
	26,67 %	73,33 %	100 %

Table 13 - Confusion matrix - di-gram clustering analysis - format RTF

Accuracy	Precision	Recall	Miss Rate
0,7394	0,8636	0,5067	0,4933

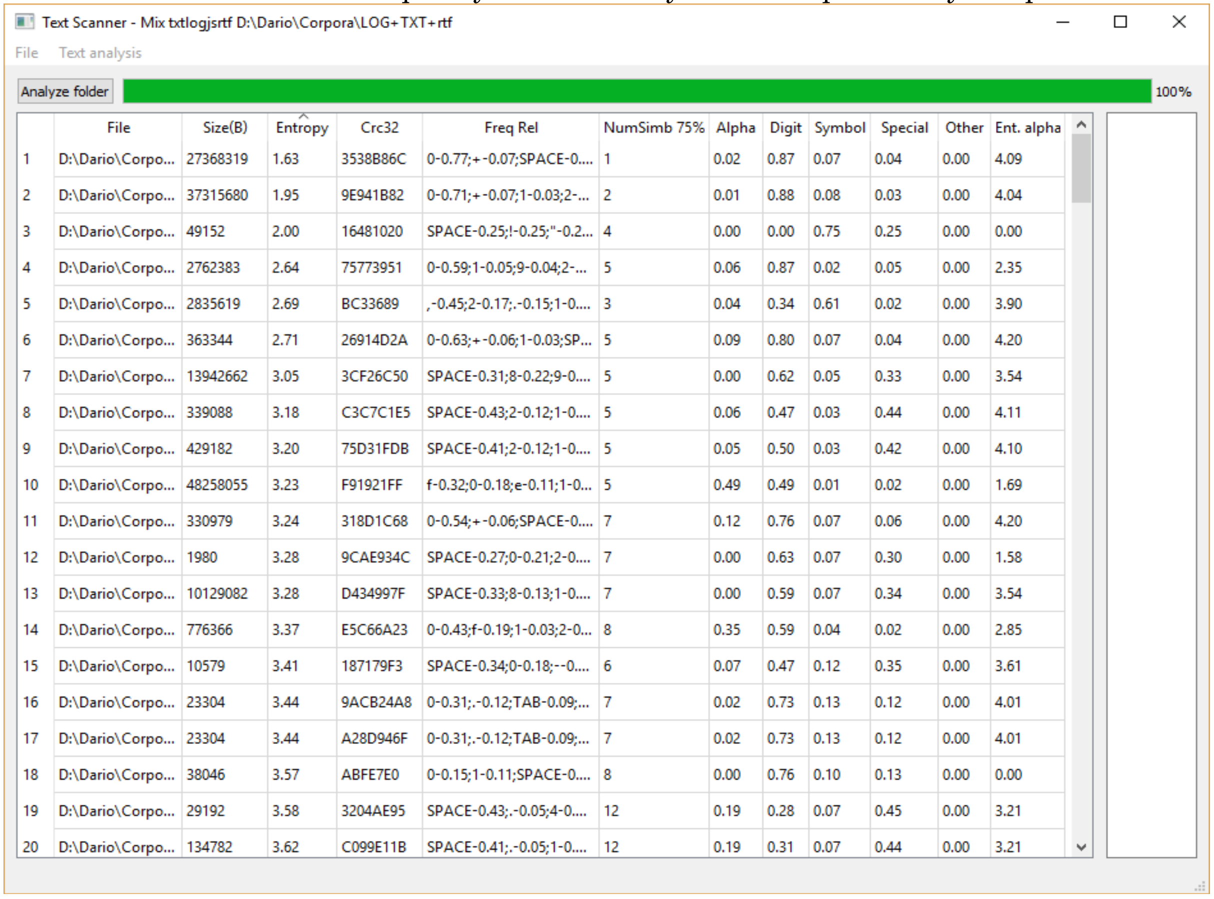
Table 14 - Di-grams analysis evaluation

The Recall and Miss Rate are better then 1-gram solution, but Accuracy and Precision are comparable, this suggests that di-gram is not the best feature for clustering file formats. The results reported in (Table 12- File type classification using di-gram clustering) shows that the cluster 3 contains only rtf files, and all js files are in cluster 1, this tell us that if a file has a recognizable grammar, this trait can be used for recognition.

4.1.3 Text Analysis

To overcome problems of 1-gram analysis many researcher used different approach. To replicate some of their results I realized a software called "TextScanner" (Figure 14 - TextScanner GUI), a partial output is in Table 15 - TextScanner export data excerpt.
A specific function allows saving the values resulting from analysis in comma separated value "csv" format and in Orange tab data format "tab" (see Table 15 - TextScanner export data excerpt).
The table contains the following feature:

- File: file name;
- Entropy: entropy of whole file Ent. Alpha: entropy computed using only alphabetical characters;
- Crc32: CRC32 value;
- Freq Rel.: most common characters whit their relative frequency;
- NumSimb 75%: number of characters needed to cover 75% of the text;
- Alpha: cumulative frequency of alphabetical characters;
- Digit: cumulative frequency of numerical characters;
- Symbol: cumulative frequency of symbol characters "()[],;-_=? …";
- Special: cumulative frequency of whitespaces characters (other than SPACE);
- Other: cumulative frequency of all other symbols not previously computed;



	File	Size(B)	Entropy	Crc32	Freq Rel	NumSimb 75%	Alpha	Digit	Symbol	Special	Other	Ent. alpha
1	D:\Dario\Corpo...	27368319	1.63	3538B86C	0-0.77;+-0.07;SPACE-0....	1	0.02	0.87	0.07	0.04	0.00	4.09
2	D:\Dario\Corpo...	37315680	1.95	9E941B82	0-0.71;+-0.07;1-0.03;2-...	2	0.01	0.88	0.08	0.03	0.00	4.04
3	D:\Dario\Corpo...	49152	2.00	16481020	SPACE-0.25;!-0.25;"-0.2...	4	0.00	0.00	0.75	0.25	0.00	0.00
4	D:\Dario\Corpo...	2762383	2.64	75773951	0-0.59;1-0.05;9-0.04;2-...	5	0.06	0.87	0.02	0.05	0.00	2.35
5	D:\Dario\Corpo...	2835619	2.69	BC33689	,-0.45;2-0.17;.-0.15;1-0....	3	0.04	0.34	0.61	0.02	0.00	3.90
6	D:\Dario\Corpo...	363344	2.71	26914D2A	0-0.63;+-0.06;1-0.03;SP...	5	0.09	0.80	0.07	0.04	0.00	4.20
7	D:\Dario\Corpo...	13942662	3.05	3CF26C50	SPACE-0.31;8-0.22;9-0....	5	0.00	0.62	0.05	0.33	0.00	3.54
8	D:\Dario\Corpo...	339088	3.18	C3C7C1E5	SPACE-0.43;2-0.12;1-0....	5	0.06	0.47	0.03	0.44	0.00	4.11
9	D:\Dario\Corpo...	429182	3.20	75D31FDB	SPACE-0.41;2-0.12;1-0....	5	0.05	0.50	0.03	0.42	0.00	4.10
10	D:\Dario\Corpo...	48258055	3.23	F91921FF	f-0.32;0-0.18;e-0.11;1-0...	5	0.49	0.49	0.01	0.02	0.00	1.69
11	D:\Dario\Corpo...	330979	3.24	318D1C68	0-0.54;+-0.06;SPACE-0....	7	0.12	0.76	0.07	0.06	0.00	4.20
12	D:\Dario\Corpo...	1980	3.28	9CAE934C	SPACE-0.27;0-0.21;2-0....	7	0.00	0.63	0.07	0.30	0.00	1.58
13	D:\Dario\Corpo...	10129082	3.28	D434997F	SPACE-0.33;8-0.13;1-0....	7	0.00	0.59	0.07	0.34	0.00	3.54
14	D:\Dario\Corpo...	776366	3.37	E5C66A23	0-0.43;f-0.19;1-0.03;2-0...	8	0.35	0.59	0.04	0.02	0.00	2.85
15	D:\Dario\Corpo...	10579	3.41	187179F3	SPACE-0.34;0-0.18;--0....	6	0.07	0.47	0.12	0.35	0.00	3.61
16	D:\Dario\Corpo...	23304	3.44	9ACB24A8	0-0.31;.-0.12;TAB-0.09;...	7	0.02	0.73	0.13	0.12	0.00	4.01
17	D:\Dario\Corpo...	23304	3.44	A28D946F	0-0.31;.-0.12;TAB-0.09;...	7	0.02	0.73	0.13	0.12	0.00	4.01
18	D:\Dario\Corpo...	38046	3.57	ABFE7E0	0-0.15;1-0.11;SPACE-0....	8	0.00	0.76	0.10	0.13	0.00	0.00
19	D:\Dario\Corpo...	29192	3.58	3204AE95	SPACE-0.43;.-0.05;4-0....	12	0.19	0.28	0.07	0.45	0.00	3.21
20	D:\Dario\Corpo...	134782	3.62	C099E11B	SPACE-0.41;.-0.05;1-0....	12	0.19	0.31	0.07	0.44	0.00	3.21

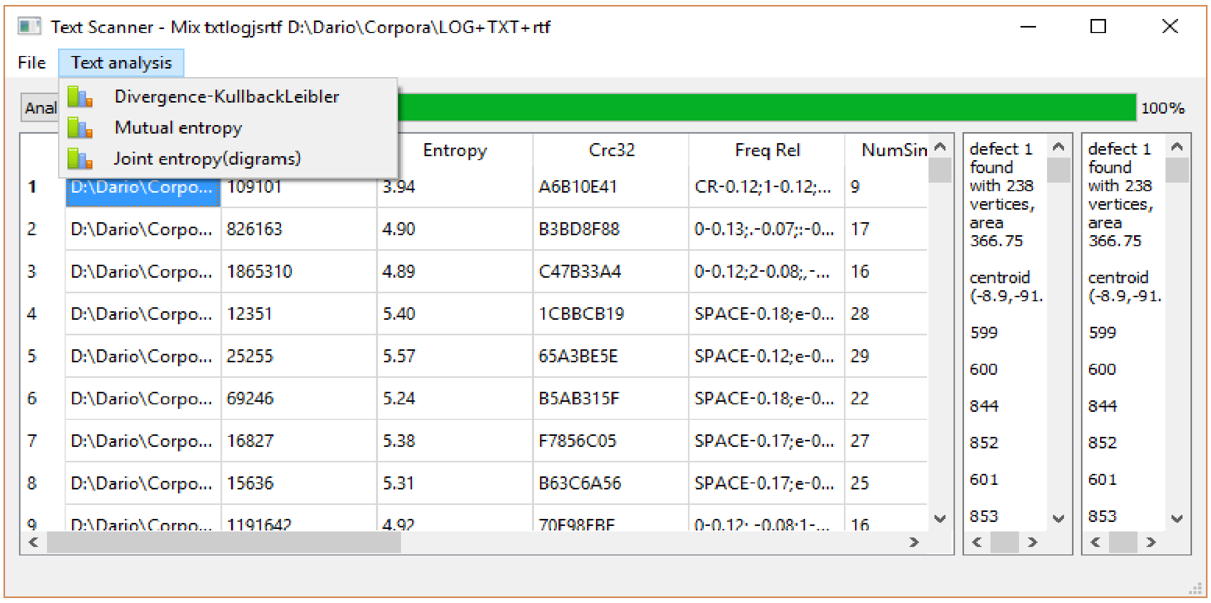
Figure 14 - TextScanner GUI

Table 15 - TextScanner export data excerpt

File	Entropy	Crc32	Freq Rel	NumSimb 75%	Alpha	Digit	Symbol	Special	Other	Ent. alpha
--\000697.log	3.94	A6B10E41	CR-0.12; 1-0.12; TAB-0.12; 3-0.07; 2-0.07;	9	0.05	0.68	0.02	0.26	0.00	3.69
--\000698.log	4.90	B3BD8F88	0-0.13; .-0.07; :-0.06; ,-0.06; SPACE-0.05;	17	0.24	0.46	0.23	0.07	0.00	4.23
--\000699.log	4.89	C47B33A4	0-0.12; 2-0.08; ,-0.07; 1-0.07; .-0.06;	16	0.22	0.49	0.24	0.06	0.00	4.21
--\000741.log	5.40	1CBBCB19	SPACE-0.18; e-0.05; t-0.03; --0.03; r-0.03;	28	0.53	0.11	0.16	0.20	0.00	4.25
--\Adventures of Huckleberry Finn.txt	4.56	BBB4975A	SPACE-0.18; e-0.08; t-0.07; o-0.06; a-0.06;	13	0.73	0.00	0.05	0.22	0.00	4.18
--\Adventures of Sherlock Holmes.txt	4.55	1728F719	SPACE-0.16; e-0.09; t-0.07; a-0.06; o-0.06;	13	0.75	0.00	0.04	0.21	0.00	4.18
--\Adventures of Tom Sawyer.txt	4.60	7D21F1F6	SPACE-0.16; e-0.09; t-0.07; o-0.06; a-0.06;	14	0.75	0.00	0.05	0.20	0.00	4.19
--\Alices Adventures in Wonderland.txt	4.64	E1E4F248	SPACE-0.17; e-0.09; t-0.07; o-0.06; a-0.05;	14	0.73	0.00	0.05	0.21	0.00	4.17
--\DataGov_CnP_FY08_Data_File.rtf	4.95	BBBA3E1A	,-0.19; 1-0.07; 2-0.05; 0-0.05; 3-0.04;	19	0.30	0.40	0.25	0.05	0.00	4.20
--\DataGov_VHA_FY07_Employability_Survey_Data.rtf	2.69	BC33689	,-0.45; 2-0.17; .-0.15; 1-0.09; 0-0.02;	3	0.04	0.34	0.61	0.02	0.00	3.90
--\DataGov_VHA_FY09_Hospital_Infrastructure_Data.rtf	5.30	A1148BBC	,-0.06; f-0.06; \-0.05; e-0.05; SPACE-0.05;	22	0.52	0.26	0.16	0.06	0.00	4.22
--\ecdd.rtf	5.07	BD82D36	f-0.10; 0-0.08; d-0.06; e-0.05; \-0.05;	19	0.62	0.26	0.07	0.05	0.00	4.10

File	Entropy	Crc32	Freq Rel	NumSimb 75%	Alpha	Digit	Symbol	Special	Other	Ent. alpha
--\dragdrop.js	4.80	AD30FCD3	SPACE-0.19; e-0.08; t-0.06; n-0.05; o-0.05;	16	0.62	0.01	0.16	0.22	0.00	4.09
--\editors (2).js	4.71	815CAA86	SPACE-0.25; e-0.07; t-0.06; n-0.04; o-0.04;	17	0.55	0.01	0.15	0.28	0.00	4.19
--\editors-2fb829b7cfa41e420407444116c345ef.js	4.71	815CAA86	SPACE-0.25; e-0.07; t-0.06; n-0.04; o-0.04;	17	0.55	0.01	0.15	0.28	0.00	4.19
--\editors.js	4.45	80CAE8E1	r-0.12; e-0.09; /-0.08; i-0.08; SPACE-0.06;	13	0.78	0.02	0.12	0.08	0.00	4.06
--\effects.js	4.87	6A1D8678	SPACE-0.18; e-0.09; t-0.07; n-0.05; i-0.05;	17	0.62	0.01	0.17	0.21	0.00	4.08

These analyses show that some features value relate to specific file types. For example, textual formats have very low zero values for digit and symbols feature. While "rtf" format has low value for "special" feature. However, the format specifications describe these traits. Our goals is discovering of the format and this task will be carried, out using grammar analysis (3.5 , 4.2 Grammar analysis).
The code of TextScanner software (Figure 15 - Text analysis in TextScanner tool) contains other tools: Kullback Leibler divergence, mutual entropy, joint entropy (digrams).
The "Divergence Kullback Leibler" (KL divergence) (S., 1959) (Kullback & Leibler, R., 1951) tool uses a reference file and computer the divergence values. The KL divergence is referred as relative entropy or information divergence. The information divergence is defined as the divergence between two probability distribution p(x) and q(x). The reference file gives an estimation of the probability distribution q(x) in a fixed filed type.


Text Scanner - Mix txtlogjsrtf D:\Dario\Corpora\LOG+TXT+rtf
File
Text analysis
Divergence-KullbackLeibler
Mutual entropy
Joint entropy(digrams)
Anal
100%

			Entropy	Crc32	Freq Rel	NumSin
1	D:\Dario\Corpo...	109101	3.94	A6B10E41	CR-0.12;1-0.12;...	9
2	D:\Dario\Corpo...	826163	4.90	B3BD8F88	0-0.13;.-0.07;:-0...	17
3	D:\Dario\Corpo...	1865310	4.89	C47B33A4	0-0.12;2-0.08;,-...	16
4	D:\Dario\Corpo...	12351	5.40	1CBBCB19	SPACE-0.18;e-0...	28
5	D:\Dario\Corpo...	25255	5.57	65A3BE5E	SPACE-0.12;e-0...	29
6	D:\Dario\Corpo...	69246	5.24	B5AB315F	SPACE-0.18;e-0...	22
7	D:\Dario\Corpo...	16827	5.38	F7856C05	SPACE-0.17;e-0...	27
8	D:\Dario\Corpo...	15636	5.31	B63C6A56	SPACE-0.17;e-0...	25
9	D:\Dario\Corpo...	1191642	4.92	70F98FBF	0-0.12;-0.08;1-...	16

```
defect 1
found
with 238
vertices,
area
366.75

centroid
(-8.9,-91.

599
600
844
852
601
853
```

```
defect 1
found
with 238
vertices,
area
366.75

centroid
(-8.9,-91.

599
600
844
852
601
853
```

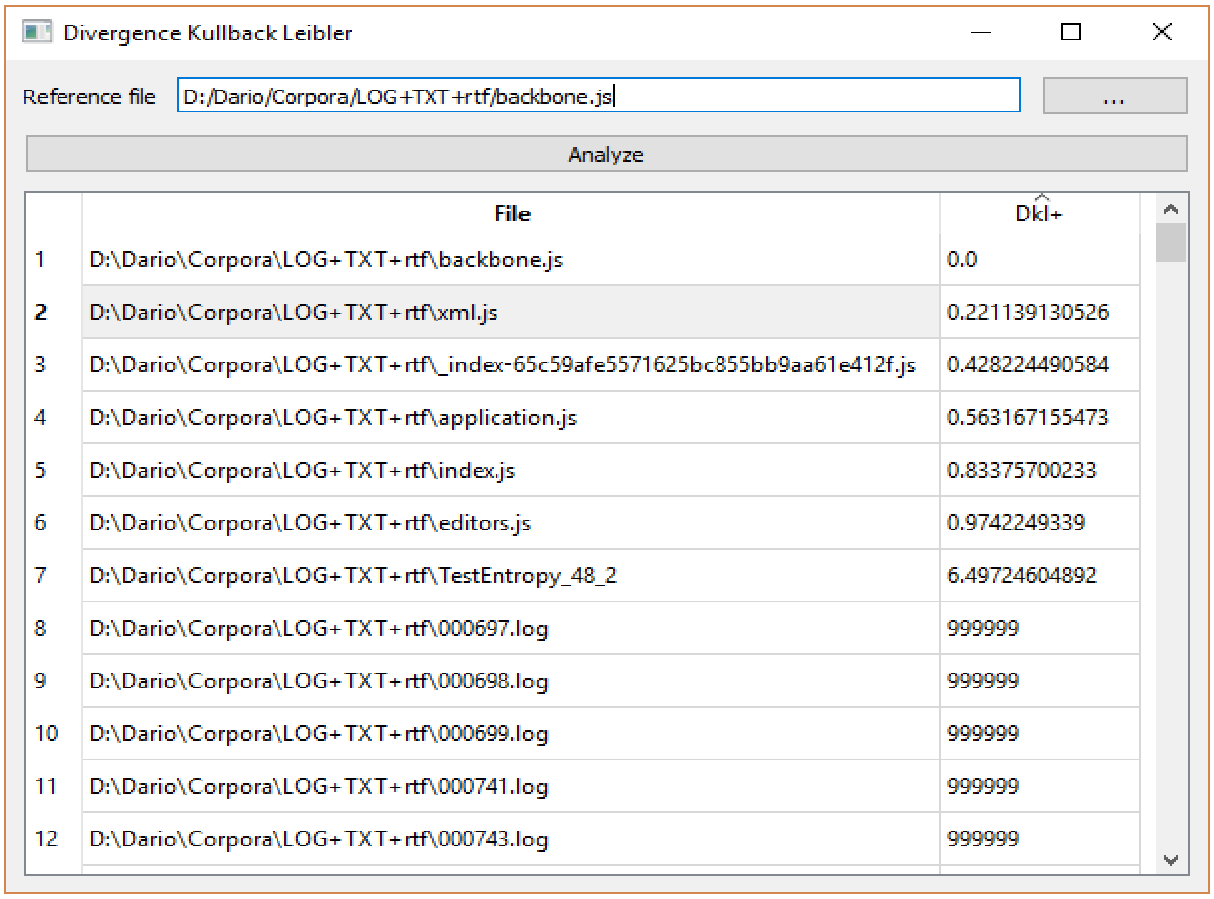
Figure 15 - Text analysis in TextScanner tool

For each file in our set, the program computes the characters frequency distribution. The two distribution give the divergence using the formula (Equation 24 - Kullback Leibler Divergence). The (Figure 16 – Kullback Leibler computing interface) shows the results of a D_{KL} computed using a JavaScript file as reference. The figures shows a divergence equal to zero for the same file, a low divergence for files of the same format, and a high value for different format files.

Equation 24 - Kullback Leibler Divergence

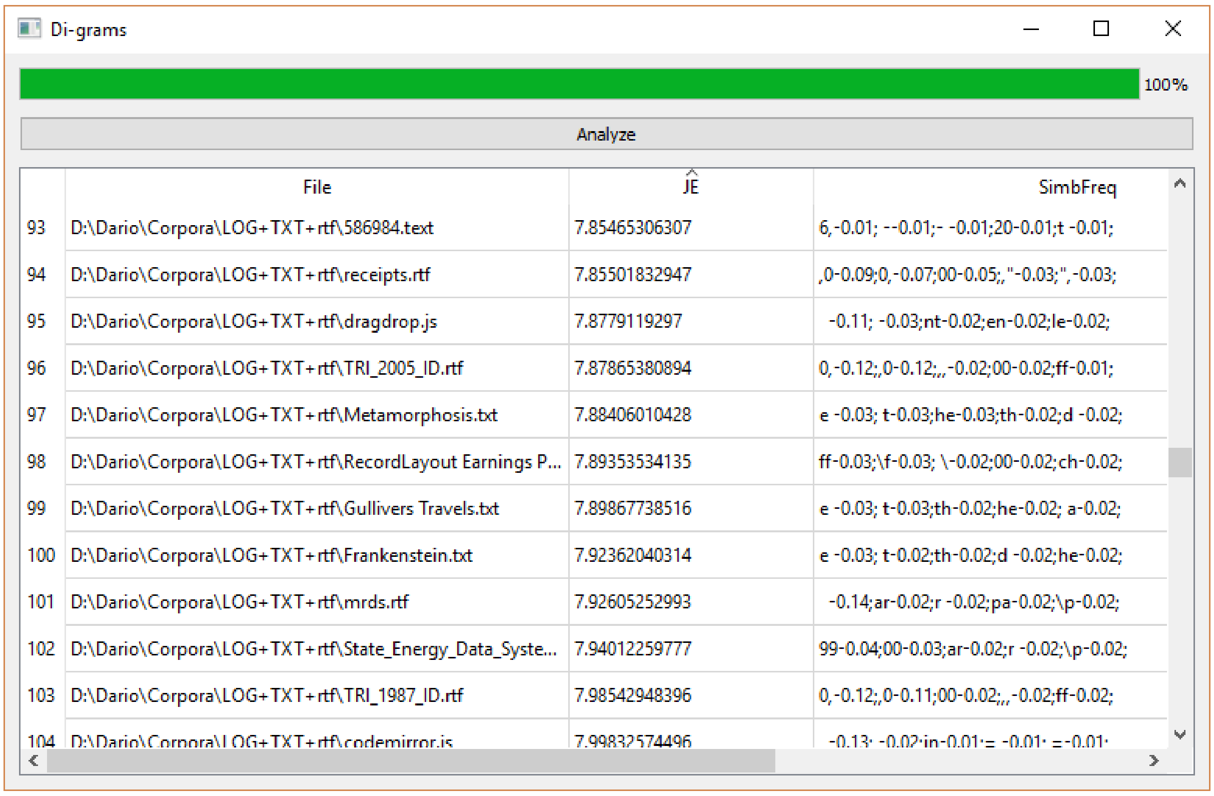
$$D_{KL}\big(p(x) \parallel q(x)\big) = \sum_{x \in X} p(x) \ln \frac{p(x)}{q(x)}$$

D_{KL} divergence is a measure (but not a metric) of the non-symmetric difference (or distance) between two probability distributions P and Q. Using an analogy with information theory, KL divergence is a relative entropy of P with respect to Q. In the context of coding theory, Kullback–Leibler divergence is a measure of the expected number of extra bits required to code samples from P using a code optimized for Q rather than the code optimized for P. The D_{KL} cannot be used to cluster files, because clustering algorithm requires the distance measure to be a metric.
The function called Joint Entropy is the entropy computed on digrams; the term “joint” indicates the entropy of the concatenation of two characters. This code function shows for each file the first 5 di-gram in order of digrams frequency. For each file, the code shows a frequency graph of two characters combination (Figure 18 - TextScanner – Di-gram frequency graph). The graphical representation of digrams distribution shows in a glance how the file is composed.


Divergence Kullback Leibler
Reference file
D:/Dario/Corpora/LOG+TXT+rtf/backbone.js
...
Analyze

	File	Dkl+
1	D:\Dario\Corpora\LOG+TXT+rtf\backbone.js	0.0
2	D:\Dario\Corpora\LOG+TXT+rtf\xml.js	0.221139130526
3	D:\Dario\Corpora\LOG+TXT+rtf_index-65c59afe5571625bc855bb9aa61e412f.js	0.428224490584
4	D:\Dario\Corpora\LOG+TXT+rtf\application.js	0.563167155473
5	D:\Dario\Corpora\LOG+TXT+rtf\index.js	0.83375700233
6	D:\Dario\Corpora\LOG+TXT+rtf\editors.js	0.9742249339
7	D:\Dario\Corpora\LOG+TXT+rtf\TestEntropy_48_2	6.49724604892
8	D:\Dario\Corpora\LOG+TXT+rtf\000697.log	999999
9	D:\Dario\Corpora\LOG+TXT+rtf\000698.log	999999
10	D:\Dario\Corpora\LOG+TXT+rtf\000699.log	999999
11	D:\Dario\Corpora\LOG+TXT+rtf\000741.log	999999
12	D:\Dario\Corpora\LOG+TXT+rtf\000743.log	999999

Figure 16 – Kullback Leibler computing interface



	File	JE	SimbFreq
93	D:\Dario\Corpora\LOG+TXT+rtf\586984.text	7.85465306307	6,-0.01; --0.01;- -0.01;20-0.01;t -0.01;
94	D:\Dario\Corpora\LOG+TXT+rtf\receipts.rtf	7.85501832947	,0-0.09;0,-0.07;00-0.05;,"-0.03;",-0.03;
95	D:\Dario\Corpora\LOG+TXT+rtf\dragdrop.js	7.8779119297	-0.11; -0.03;nt-0.02;en-0.02;le-0.02;
96	D:\Dario\Corpora\LOG+TXT+rtf\TRI_2005_ID.rtf	7.87865380894	0,-0.12;,0-0.12;,,-0.02;00-0.02;ff-0.01;
97	D:\Dario\Corpora\LOG+TXT+rtf\Metamorphosis.txt	7.88406010428	e -0.03; t-0.03;he-0.03;th-0.02;d -0.02;
98	D:\Dario\Corpora\LOG+TXT+rtf\RecordLayout Earnings P...	7.89353534135	ff-0.03;\f-0.03; \-0.02;00-0.02;ch-0.02;
99	D:\Dario\Corpora\LOG+TXT+rtf\Gullivers Travels.txt	7.89867738516	e -0.03; t-0.03;th-0.02;he-0.02; a-0.02;
100	D:\Dario\Corpora\LOG+TXT+rtf\Frankenstein.txt	7.92362040314	e -0.03; t-0.02;th-0.02;d -0.02;he-0.02;
101	D:\Dario\Corpora\LOG+TXT+rtf\mrds.rtf	7.92605252993	-0.14;ar-0.02;r -0.02;pa-0.02;\p-0.02;
102	D:\Dario\Corpora\LOG+TXT+rtf\State_Energy_Data_Syste...	7.94012259777	99-0.04;00-0.03;ar-0.02;r -0.02;\p-0.02;
103	D:\Dario\Corpora\LOG+TXT+rtf\TRI_1987_ID.rtf	7.98542948396	0,-0.12;,0-0.11;00-0.02;,,-0.02;ff-0.02;
104	D:\Dario\Corpora\LOG+TXT+rtf\codemirror.js	7.99832574496	-0.13; -0.02;in-0.01;= -0.01; =-0.01;

Figure 17 - Entropy computed in digrams

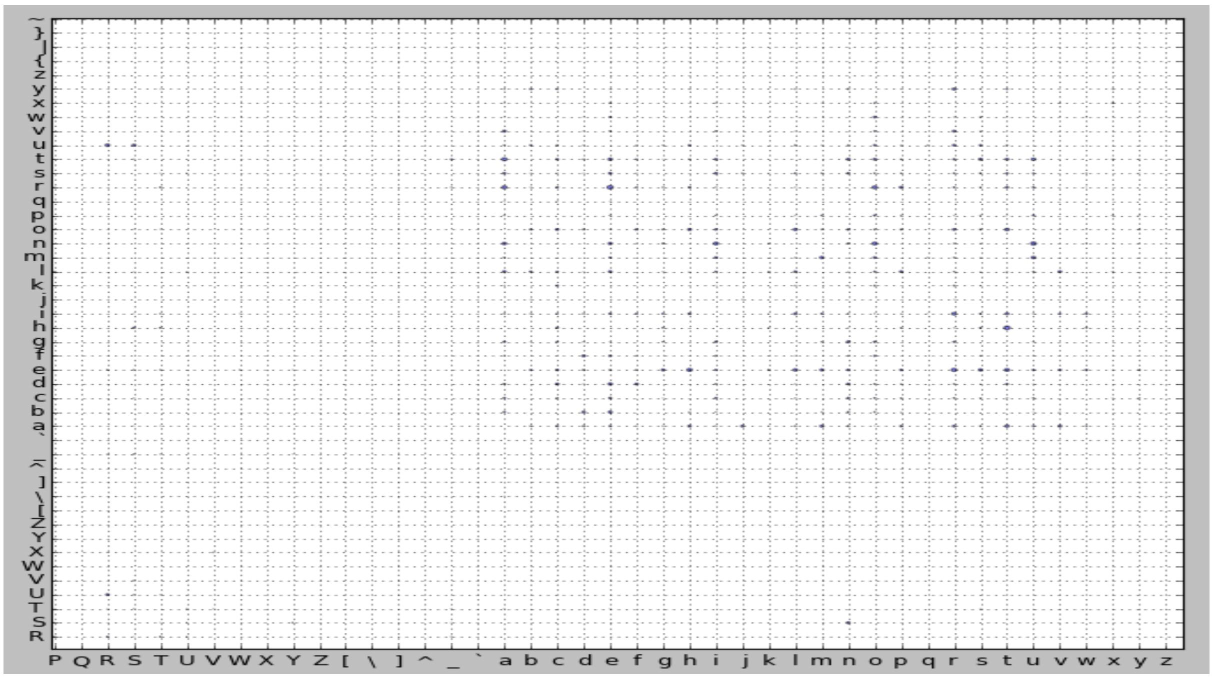


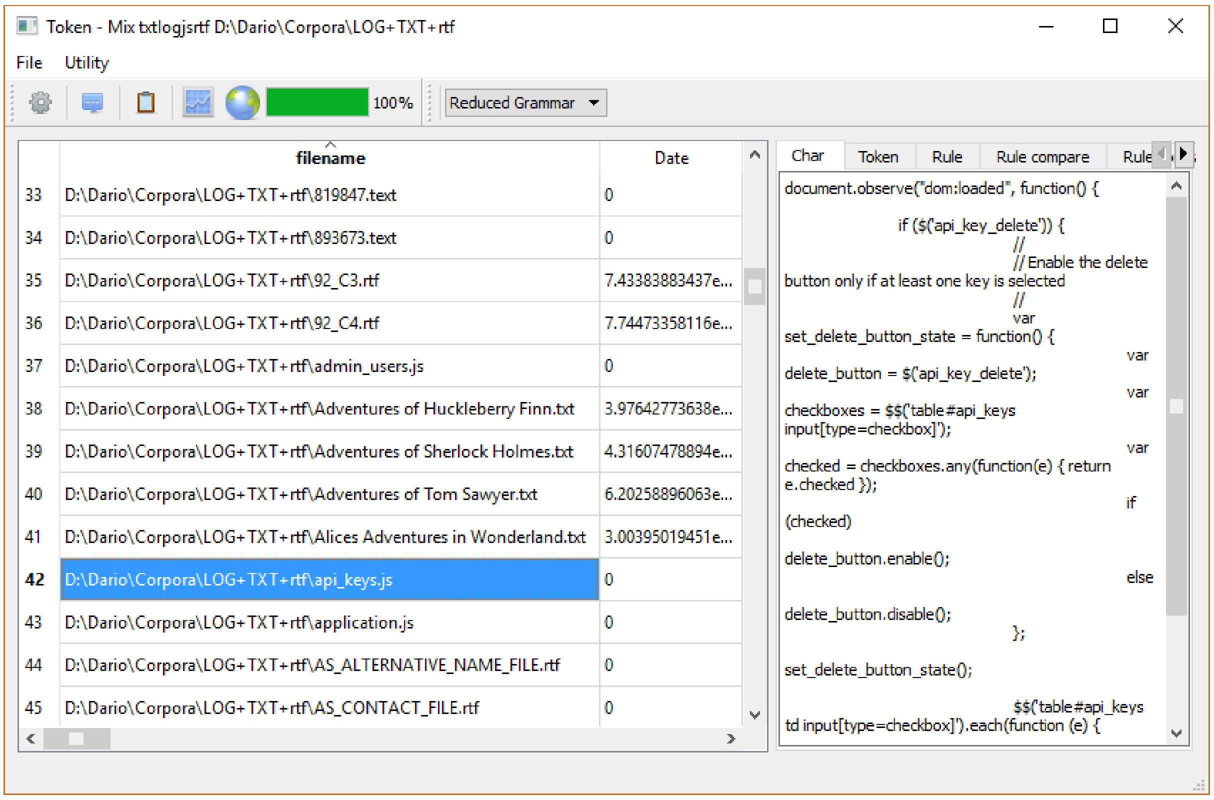
Figure 18 - TextScanner – Di-gram frequency graph

4.2 Grammar analysis

The n-grams analyses show some lack in file formats discrimination. To discriminate between different formats, we must assign to each sequence of bytes, the right meaning. The grammar analysis helps in this task using lexical and semantic analyses. The tool called Token (Figure 19 - Token software "char view") contains the analysis presented in the previous chapter (3.5).

4.2.1 Token computing

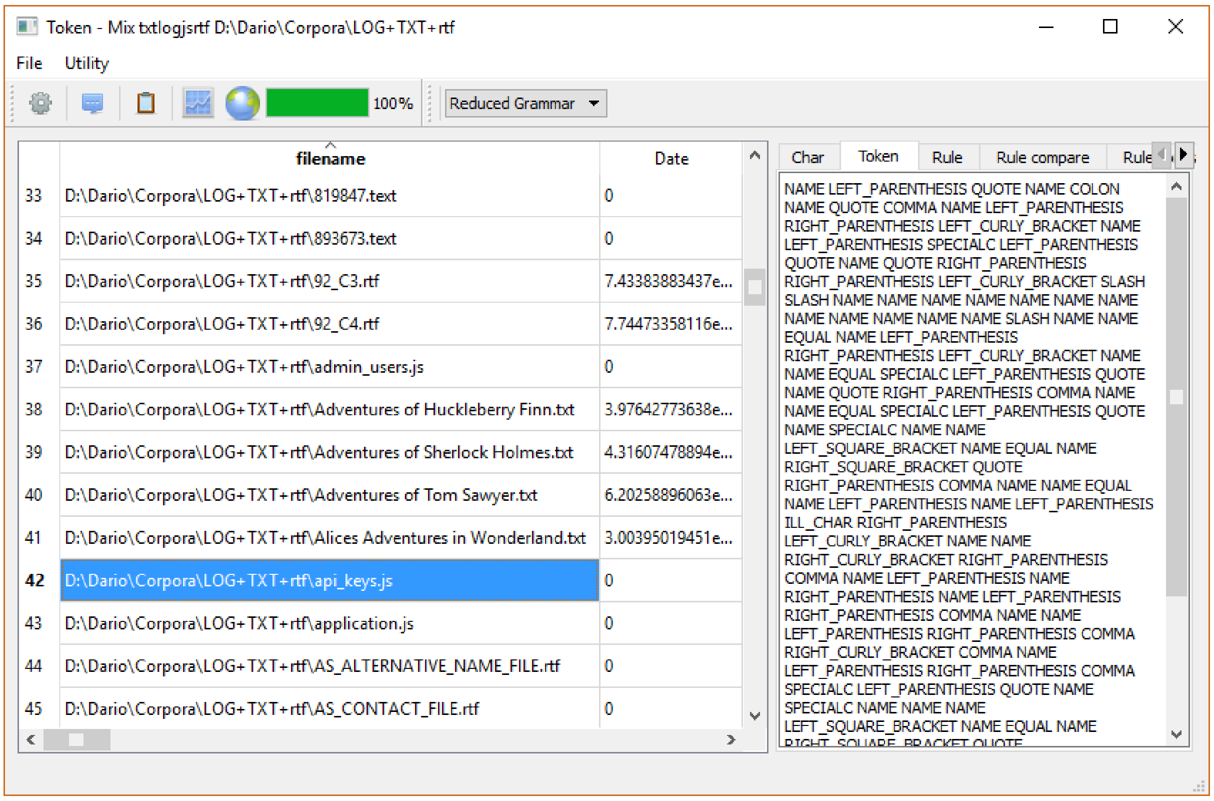
The application developed applies a lexical analysis to identify tokens (Figure 19 - Token software "char view"). The program can accomplish the operation using three different grammar “Reduced grammar”, “Large grammar” and a specialized grammar “PDF grammar”.
The three grammars differ for the set of lexical rules, they generate very different output. The resulting tokens (Figure 20 - Token software "token view") of “Reduced grammar” and “Large grammar” are analysed to infer grammar rules.
The “PDF grammar” generates a special purpose tokenized output. The output file can be analysed from a PDF parser to identify if the token sequence fits the PDF grammar. This technique divide fragment of PDF files from other fragments type. The “PDF grammar” analysis applied on a whole files detect the perfect fitting, while applied on a fragment outputs a partial fitting, this output give a measure of the similarity with respect to a pdf grammar.
The code outputs also a token frequency for each file; the token frequency gives information about the file format, we can use these values to act further analyses. For example, a high frequency of “DATE” is typical of a log file; the same consideration fits for the token “NUMERIC”. The program store resulting values in Orange tab data format “tab” to be used in further analyses.



	filename	Date
33	D:\Dario\Corpora\LOG+TXT+rtf\819847.text	0
34	D:\Dario\Corpora\LOG+TXT+rtf\893673.text	0
35	D:\Dario\Corpora\LOG+TXT+rtf\92_C3.rtf	7.43383883437e...
36	D:\Dario\Corpora\LOG+TXT+rtf\92_C4.rtf	7.74473358116e...
37	D:\Dario\Corpora\LOG+TXT+rtf\admin_users.js	0
38	D:\Dario\Corpora\LOG+TXT+rtf\Adventures of Huckleberry Finn.txt	3.97642773638e...
39	D:\Dario\Corpora\LOG+TXT+rtf\Adventures of Sherlock Holmes.txt	4.31607478894e...
40	D:\Dario\Corpora\LOG+TXT+rtf\Adventures of Tom Sawyer.txt	6.20258896063e...
41	D:\Dario\Corpora\LOG+TXT+rtf\Alices Adventures in Wonderland.txt	3.00395019451e...
42	D:\Dario\Corpora\LOG+TXT+rtf\api_keys.js	0
43	D:\Dario\Corpora\LOG+TXT+rtf\application.js	0
44	D:\Dario\Corpora\LOG+TXT+rtf\AS_ALTERNATIVE_NAME_FILE.rtf	0
45	D:\Dario\Corpora\LOG+TXT+rtf\AS_CONTACT_FILE.rtf	0

```
document.observe("dom:loaded", function() {
                if ($('api_key_delete')) {
                        //
                        // Enable the delete
button only if at least one key is selected
                        //
                        var
set_delete_button_state = function() {
                                        var
delete_button = $('api_key_delete');
                                        var
checkboxes = $$('table#api_keys
input[type=checkbox]');
                                        var
checked = checkboxes.any(function(e) { return
e.checked });
                                        if
(checked)
delete_button.enable();
                                        else
delete_button.disable();
                        };
set_delete_button_state();
                        $$('table#api_keys
td input[type=checkbox]').each(function (e) {
```

Figure 19 - Token software "char view"



	filename	Date
33	D:\Dario\Corpora\LOG+TXT+rtf\819847.text	0
34	D:\Dario\Corpora\LOG+TXT+rtf\893673.text	0
35	D:\Dario\Corpora\LOG+TXT+rtf\92_C3.rtf	7.43383883437e...
36	D:\Dario\Corpora\LOG+TXT+rtf\92_C4.rtf	7.74473358116e...
37	D:\Dario\Corpora\LOG+TXT+rtf\admin_users.js	0
38	D:\Dario\Corpora\LOG+TXT+rtf\Adventures of Huckleberry Finn.txt	3.97642773638e...
39	D:\Dario\Corpora\LOG+TXT+rtf\Adventures of Sherlock Holmes.txt	4.31607478894e...
40	D:\Dario\Corpora\LOG+TXT+rtf\Adventures of Tom Sawyer.txt	6.20258896063e...
41	D:\Dario\Corpora\LOG+TXT+rtf\Alices Adventures in Wonderland.txt	3.00395019451e...
42	D:\Dario\Corpora\LOG+TXT+rtf\api_keys.js	0
43	D:\Dario\Corpora\LOG+TXT+rtf\application.js	0
44	D:\Dario\Corpora\LOG+TXT+rtf\AS_ALTERNATIVE_NAME_FILE.rtf	0
45	D:\Dario\Corpora\LOG+TXT+rtf\AS_CONTACT_FILE.rtf	0

```
NAME LEFT_PARENTHESIS QUOTE NAME COLON
NAME QUOTE COMMA NAME LEFT_PARENTHESIS
RIGHT_PARENTHESIS LEFT_CURLY_BRACKET NAME
LEFT_PARENTHESIS SPECIALC LEFT_PARENTHESIS
QUOTE NAME QUOTE RIGHT_PARENTHESIS
RIGHT_PARENTHESIS LEFT_CURLY_BRACKET SLASH
SLASH NAME NAME NAME NAME NAME NAME NAME
NAME NAME NAME NAME NAME SLASH NAME NAME
EQUAL NAME LEFT_PARENTHESIS
RIGHT_PARENTHESIS LEFT_CURLY_BRACKET NAME
NAME EQUAL SPECIALC LEFT_PARENTHESIS QUOTE
NAME QUOTE RIGHT_PARENTHESIS COMMA NAME
NAME EQUAL SPECIALC LEFT_PARENTHESIS QUOTE
NAME SPECIALC NAME NAME
LEFT_SQUARE_BRACKET NAME EQUAL NAME
RIGHT_SQUARE_BRACKET QUOTE
RIGHT_PARENTHESIS COMMA NAME NAME EQUAL
NAME LEFT_PARENTHESIS NAME LEFT_PARENTHESIS
ILL_CHAR RIGHT_PARENTHESIS
LEFT_CURLY_BRACKET NAME NAME
RIGHT_CURLY_BRACKET RIGHT_PARENTHESIS
COMMA NAME LEFT_PARENTHESIS NAME
RIGHT_PARENTHESIS NAME LEFT_PARENTHESIS
RIGHT_PARENTHESIS COMMA NAME NAME
LEFT_PARENTHESIS RIGHT_PARENTHESIS COMMA
RIGHT_CURLY_BRACKET COMMA NAME
LEFT_PARENTHESIS RIGHT_PARENTHESIS COMMA
SPECIALC LEFT_PARENTHESIS QUOTE NAME
SPECIALC NAME NAME NAME
LEFT_SQUARE_BRACKET NAME EQUAL NAME
RIGHT_SQUARE_BRACKET QUOTE
```

Figure 20 - Token software "token view"

Using token frequency distribution the program can compute cluster and the table 6 (Table 16 -File type classification using token clustering) shows the results.
The table show a good separation of sample in the cluster, the mean silhouette value is 0.474926840849. In (Figure 22 - Silhouette values) we can observe the distribution of the silhouette values. Computing the mean silhouette values for different number of clusters k (Figure 21 - Silhouette values for k value from 2 to 8), and computing the mean silhouette, we can see that the max value is for k=4. This is good, because the number of different format is 4.

Cluster	Rtf	Log	Js	Txt
0	0 %	100 %	0 %	0 %
1	6,25 %	0 %	7,14 %	100 %
2	93,75 %	0 %	0 %	0 %
3	0 %	0 %	92,86 %	0 %

Table 16 -File type classification using token clustering

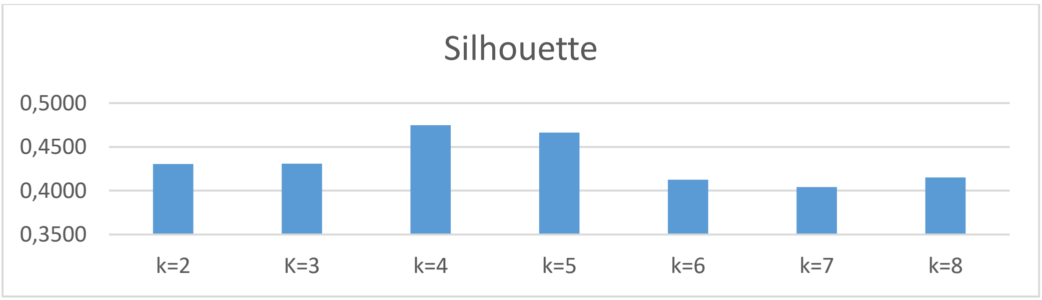


Figure 21 - Silhouette values for k value from 2 to 8

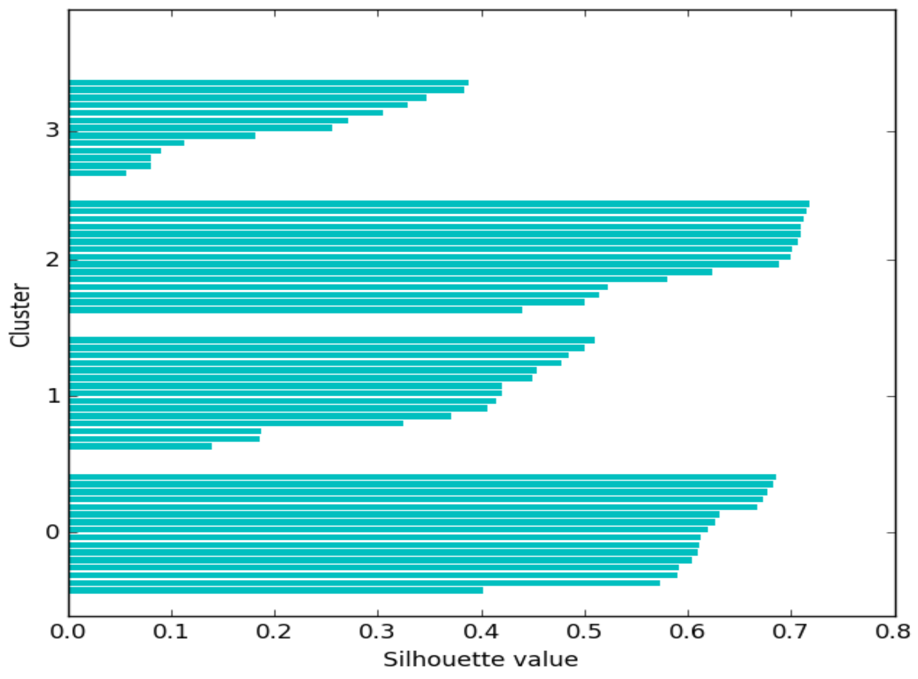


Figure 22 - Silhouette values

Table 17 - RTF format/Cluster 2 confusion Matrix

Cluster 2 RTF	Prediction positive	Prediction negative	
Condition positive	TP = 25,42 %	FN = 1.69 %	27,11
Condition negative	FP = 0 %	TN = 72,89 %	72,89
	25,42	74,58	100

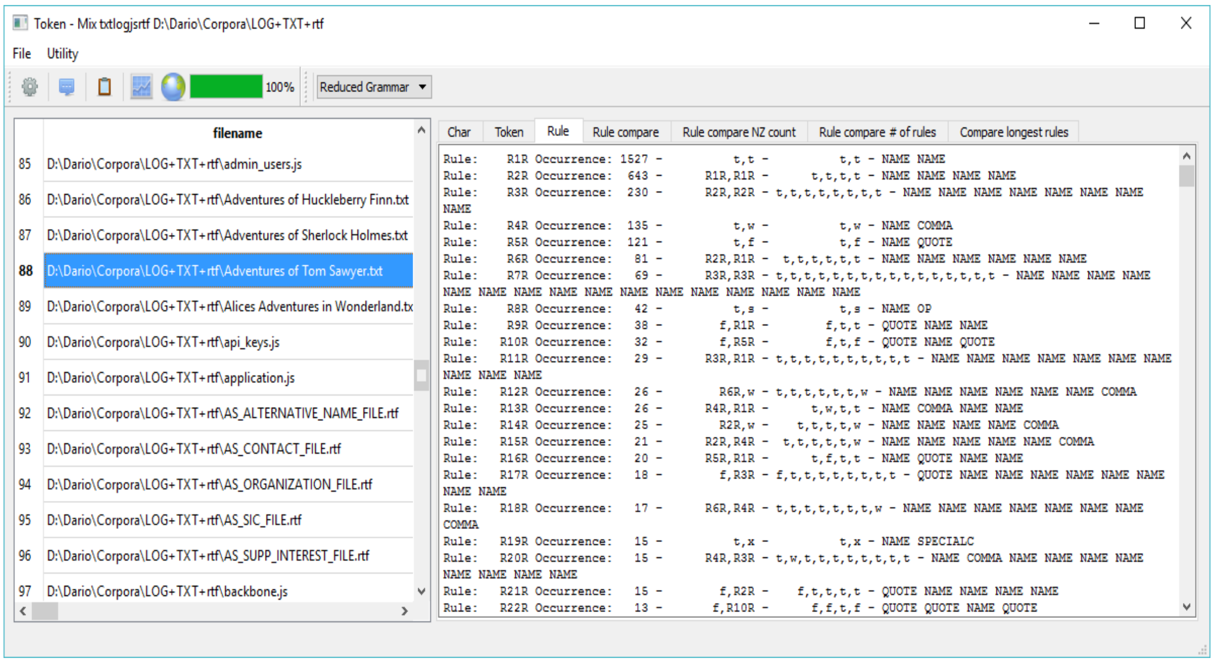
Accuracy	Precision	Recall	Miss Rate
0,9831	1	0,9376	0,0623

Table 18 - Token analysis evaluation

Identifying Cluster 2 as RTF file format, using the confusion matrix (Table 17 - RTF format/Cluster 2 confusion Matrix) we can compute evaluation measure as Accuracy (Equation 20 - Accuracy), precision (Equation 21 - Precision), recall (Equation 22 - Recall) and miss rate (Equation 23 - Miss Rate).
All this parameter defines a good performance of this unsupervised classification process. The same happens for other formats/clusters.

4.2.2 Rules computing

This last step analyses token sequence to extract “rules” (Figure 23 - Token software "rules"), this task looks for recurrent patterns. The code implements this task using the RePair algorithm (Moffat & N. Jesper Larsson, 1999), the code contains also the Sequitur (Nevill-Manning & Witten, 1997) algorithm (to run this algorithm we need to modify the calling function in the source code). The code computes a set of rules, then estimates how many times each rule fits over the whole file fragment.

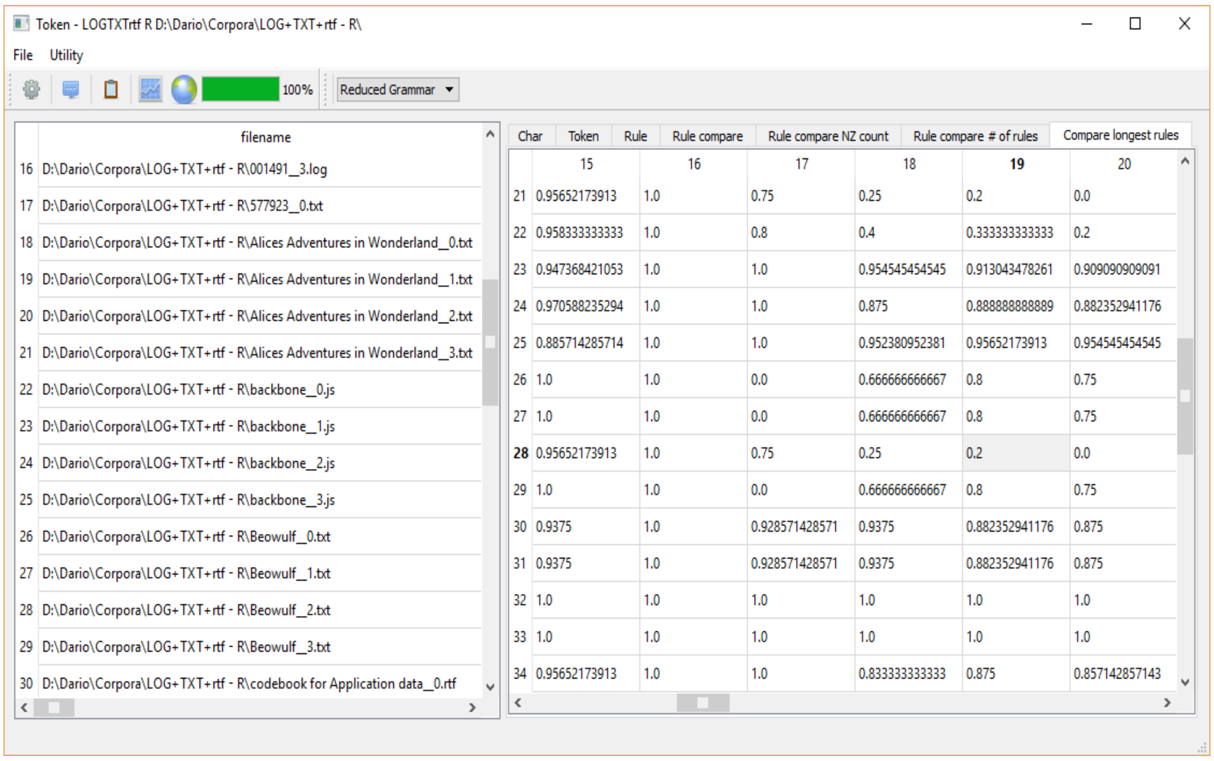

Token - Mix txtlogjsrtf D:\Dario\Corpora\LOG+TXT+rtf

	filename
85	D:\Dario\Corpora\LOG+TXT+rtf\admin_users.js
86	D:\Dario\Corpora\LOG+TXT+rtf\Adventures of Huckleberry Finn.txt
87	D:\Dario\Corpora\LOG+TXT+rtf\Adventures of Sherlock Holmes.txt
88	D:\Dario\Corpora\LOG+TXT+rtf\Adventures of Tom Sawyer.txt
89	D:\Dario\Corpora\LOG+TXT+rtf\Alices Adventures in Wonderland.tx
90	D:\Dario\Corpora\LOG+TXT+rtf\api_keys.js
91	D:\Dario\Corpora\LOG+TXT+rtf\application.js
92	D:\Dario\Corpora\LOG+TXT+rtf\AS_ALTERNATIVE_NAME_FILE.rtf
93	D:\Dario\Corpora\LOG+TXT+rtf\AS_CONTACT_FILE.rtf
94	D:\Dario\Corpora\LOG+TXT+rtf\AS_ORGANIZATION_FILE.rtf
95	D:\Dario\Corpora\LOG+TXT+rtf\AS_SIC_FILE.rtf
96	D:\Dario\Corpora\LOG+TXT+rtf\AS_SUPP_INTEREST_FILE.rtf
97	D:\Dario\Corpora\LOG+TXT+rtf\backbone.js

```
Rule:    R1R Occurrence: 1527 -        t,t -        t,t - NAME NAME
Rule:    R2R Occurrence:  643 -    R1R,R1R -    t,t,t,t - NAME NAME NAME NAME
Rule:    R3R Occurrence:  230 -    R2R,R2R - t,t,t,t,t,t,t,t - NAME NAME NAME NAME NAME NAME NAME
NAME
Rule:    R4R Occurrence:  135 -        t,w -        t,w - NAME COMMA
Rule:    R5R Occurrence:  121 -        t,f -        t,f - NAME QUOTE
Rule:    R6R Occurrence:   81 -    R2R,R1R -  t,t,t,t,t,t - NAME NAME NAME NAME NAME NAME
Rule:    R7R Occurrence:   69 -    R3R,R3R - t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t - NAME NAME NAME NAME
NAME NAME NAME NAME NAME NAME NAME NAME NAME NAME NAME NAME
Rule:    R8R Occurrence:   42 -        t,s -        t,s - NAME OP
Rule:    R9R Occurrence:   38 -      f,R1R -      f,t,t - QUOTE NAME NAME
Rule:   R10R Occurrence:   32 -      f,R5R -      f,t,f - QUOTE NAME QUOTE
Rule:   R11R Occurrence:   29 -    R3R,R1R - t,t,t,t,t,t,t,t,t,t - NAME NAME NAME NAME NAME NAME NAME
NAME NAME NAME
Rule:   R12R Occurrence:   26 -      R6R,w - t,t,t,t,t,t,w - NAME NAME NAME NAME NAME NAME COMMA
Rule:   R13R Occurrence:   26 -    R4R,R1R -    t,w,t,t - NAME COMMA NAME NAME
Rule:   R14R Occurrence:   25 -      R2R,w -  t,t,t,t,w - NAME NAME NAME NAME COMMA
Rule:   R15R Occurrence:   21 -    R2R,R4R -  t,t,t,t,t,w - NAME NAME NAME NAME NAME COMMA
Rule:   R16R Occurrence:   20 -    R5R,R1R -    t,f,t,t - NAME QUOTE NAME NAME
Rule:   R17R Occurrence:   18 -      f,R3R - f,t,t,t,t,t,t,t,t - QUOTE NAME NAME NAME NAME NAME NAME
NAME NAME
Rule:   R18R Occurrence:   17 -    R6R,R4R - t,t,t,t,t,t,t,w - NAME NAME NAME NAME NAME NAME NAME
COMMA
Rule:   R19R Occurrence:   15 -        t,x -        t,x - NAME SPECIALC
Rule:   R20R Occurrence:   15 -    R4R,R3R - t,w,t,t,t,t,t,t,t,t - NAME COMMA NAME NAME NAME NAME
NAME NAME NAME NAME
Rule:   R21R Occurrence:   15 -      f,R2R -  f,t,t,t,t - QUOTE NAME NAME NAME NAME
Rule:   R22R Occurrence:   13 -     f,R10R -    f,f,t,f - QUOTE QUOTE NAME QUOTE
```

Figure 23 - Token software "rules"

The code implements the rule similarity explained in the previous chapter (Rule Analysis) (Figure 24 - Token software rule comparison). The values represent a distance, then lower is the value and more similar are the file format. The column "Rule compare" reports the distance value computed using all the rule, the column "Rule compare NZ count" shows the number of non-zero distance in the full rules matrix compare. "Rule compare #of rule" indicates the total number of rules to compare to obtain the distance of the grammars inducted on the two files under analyses.


Token - LOGTXTrtf R D:\Dario\Corpora\LOG+TXT+rtf - R\

	filename
16	D:\Dario\Corpora\LOG+TXT+rtf - R\001491__3.log
17	D:\Dario\Corpora\LOG+TXT+rtf - R\577923__0.txt
18	D:\Dario\Corpora\LOG+TXT+rtf - R\Alices Adventures in Wonderland__0.txt
19	D:\Dario\Corpora\LOG+TXT+rtf - R\Alices Adventures in Wonderland__1.txt
20	D:\Dario\Corpora\LOG+TXT+rtf - R\Alices Adventures in Wonderland__2.txt
21	D:\Dario\Corpora\LOG+TXT+rtf - R\Alices Adventures in Wonderland__3.txt
22	D:\Dario\Corpora\LOG+TXT+rtf - R\backbone__0.js
23	D:\Dario\Corpora\LOG+TXT+rtf - R\backbone__1.js
24	D:\Dario\Corpora\LOG+TXT+rtf - R\backbone__2.js
25	D:\Dario\Corpora\LOG+TXT+rtf - R\backbone__3.js
26	D:\Dario\Corpora\LOG+TXT+rtf - R\Beowulf__0.txt
27	D:\Dario\Corpora\LOG+TXT+rtf - R\Beowulf__1.txt
28	D:\Dario\Corpora\LOG+TXT+rtf - R\Beowulf__2.txt
29	D:\Dario\Corpora\LOG+TXT+rtf - R\Beowulf__3.txt
30	D:\Dario\Corpora\LOG+TXT+rtf - R\codebook for Application data__0.rtf

	15	16	17	18	19	20
21	0.95652173913	1.0	0.75	0.25	0.2	0.0
22	0.958333333333	1.0	0.8	0.4	0.333333333333	0.2
23	0.947368421053	1.0	1.0	0.954545454545	0.913043478261	0.909090909091
24	0.970588235294	1.0	1.0	0.875	0.888888888889	0.882352941176
25	0.885714285714	1.0	1.0	0.952380952381	0.95652173913	0.954545454545
26	1.0	1.0	0.0	0.666666666667	0.8	0.75
27	1.0	1.0	0.0	0.666666666667	0.8	0.75
28	0.95652173913	1.0	0.75	0.25	0.2	0.0
29	1.0	1.0	0.0	0.666666666667	0.8	0.75
30	0.9375	1.0	0.928571428571	0.9375	0.882352941176	0.875
31	0.9375	1.0	0.928571428571	0.9375	0.882352941176	0.875
32	1.0	1.0	1.0	1.0	1.0	1.0
33	1.0	1.0	1.0	1.0	1.0	1.0
34	0.95652173913	1.0	1.0	0.833333333333	0.875	0.857142857143

Figure 24 - Token software rule comparison

The column “Compare longest rules” give a fast result, it operates comparing only the longest rule, the results interpretation is simple, the values equal to 0 if trigrams are identical, values equal to 1.0 if no trigrams are common.
The grammar induction produce a number of rules very high, this creates an heavy computational problem when we compute the distance between grammar using the full inducted grammar, a fast shortcut consists in the distance computation that consider only the longest rules, this methods gives a good approximation of the results.
The similarity computed, using grammar comparison, results in a syntactical and semantical analysis; this gives information about the structure of the content, not only of the format.

4.2.3 The computational load

The computational load restricts field of application of this algorithm. The only way to use this procedure is to compare a given files with a set of files. The method inducts a set of rules; my procedure uses all rules or the longest rule for the comparison. A better solution is, to identify the rules which best describes the format and/or the content of a file.
The algorithm, as is now, uses a full grammar analysis, it requires great amount computing power. The only way to get the analysis tasks done is to split the load using more computing elements. The Map Reduce (Lämmel, 2008) paradigm helps to distribute computing load. I have prepared a little computing cluster (three nodes) using Hadoop (The Apache Software Foundation, 2016), based on Debian operating system. I tested a simple version of my code for performance evaluation. I tested Hadoop and MapReduce technique in order to develop a scalable platform for grammar analyses. However, my original python code requires a lot of work to run on Hadoop-MapReduce. Therefore, after the proof of concept, I did not continue the test of this infrastructure; because the best solution is to re-implement all the code using Java programming language instead of Python language.

5. Fragments in Deduplicated File Systems

Deduplication splits files into fragments, which are stored in a chunk repository. Deduplication stores chunks that are common to multiple files only once. From a forensics point of view, a deduplicated device is very difficult to recover and it requires a specific knowledge of how this technology operates. Deduplication starts from a whole file, and transforms it in an organized set of fragments. In the recent past, it was reserved to datacenters, and used to reduce space for backups inside virtual tape library (VTL) devices. Now this technology is available in open source packages like OpenDedup, or directly as an operating system feature, as in Microsoft Windows Server or in ZFS. Recently Microsoft included this feature in Windows 10 Technical Preview. Digital investigation tools need to be improved to detect, analyze and recover the content of deduplicated file systems. Deduplication adds a layer to data access that needs to be investigated, in order to act correctly during seizure and further analysis. This research analyzes deduplication technology in the perspective of a digital forensic investigation.

5.1 Introduction

The architecture evolution of deduplicated file systems has been mature for production environment since many years, but now it is ready for office and consumer environment.
Digital forensic analyses are frequently required for many types of crimes, not only for cybercrime. In most cases, the practitioner has to extract some files from file systems, to restore some other from backups, and to analyze a bunch of digital media as USB disks, SD cards, and NAS storages. Analyses involving datacenters are done with the collaboration of data center staff and technology and deduplication is handled transparently. Now that this technology is arriving at a consumer level (Windows 10 Technical Preview-2016), a higher level of awareness is required. Seizing an usb disk of some TB, without knowledge of the presence of deduplicated volumes, makes it difficult and sometimes impossible to extract data.
The use of a deduplicated file system is transparent to the user, and gives optimal results in terms of space saving. The saving improvement estimated from Microsoft (El-Shimi, et al., 2012) using basic chunking is of 25.82% for Office-2007 documents (docx), and 9.96% for PDF. These values are calculated using GFS-US dataset.
This analysis explains how deduplication works, how we can identify a particular type of deduplication implementation, and how to reconstruct files for a specific configuration. Traditional data carvers do not recognize the presence of a deduplicated file system. Microsoft implements this feature as an extension of NTFS, adding a reparse point attribute in the file entry. Reading the NTFS Master File Table ($MFT) of a system with deduplication enabled, a forensic tool can show files and folder structures, but cannot extract the files' content. A similar problem was present the first time NTFS introduced files and folders compression.

5.2 Previous Work

Deduplication is studied from the point of view of algorithms and their efficiency (Muthitacharoen, Chen, & Mazieres, 2001) (Min, Yoon, & Won, Efficient Deduplication Techniques for Modern Backup Operation, 2011) and a brief introduction to new storage technologies in a forensics perspective is explained in this article (Carlton & Matsumoto, A survey of contemporary enterprise storage technologies from a digital forensics perspective, 2011). The authors indicate the need for thorough study using experimental data, and physical acquisition and underline the importance of markers that help to recognize storage technologies. Deduplication is used in smartphone memory analysis to detect duplicated pages (Park, Hyunji, & Sangjin, 2012), because flash memory pages are not deleted and rewritten when data is modified, but a new page is created according to Flash Translation Layer algorithm (Chung T.-S. , et al., 2009) (Park, Hyunji, & Sangjin, 2012), and the updated content is saved in it (Harnik, Pinkas, & Shulman-Peleg, 2010). Deduplication is also considered a useful technology to reduce space needed to archive digital evidence (Neuner, Mulazzani, Schrittwieser, & Weippl, 2015) (Neuner, Schmiedecker, & Weippl, 2016).

5.3 Deduplication

Deduplication is a process that works in order to reduce duplication of data on a device. Data deduplication is used in backup and archive processes, while network deduplication is used to reduce network bandwidth usage in some applications. Deduplication can be done at file-level (SIS Single Instance Storage) or at block-level.
The deduplication process is considered in-line if it is done before saving data on the device, while is considered a postprocess, if data is first stored on a device and then deduplicated, according to some parameters as file age, file type and file usage. An example of inline deduplication is OpenDedup, while an example of post-process is the deduplication engine integrated in Microsoft Windows Server (2012) (and 2016) (El-Shimi, et al., 2012).

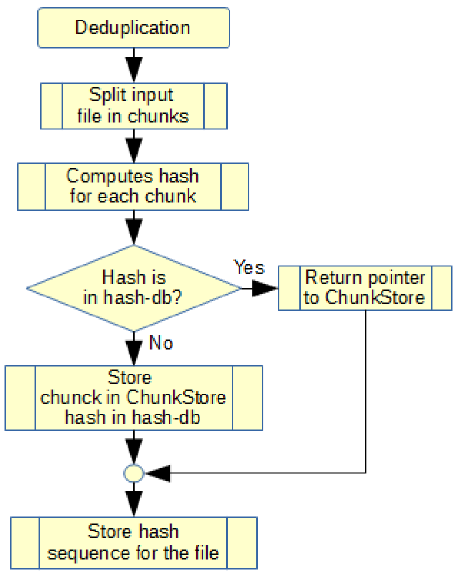

Deduplication
Split input file in chunks
Computes hash for each chunk
Hash is in hash-db?
Yes
Return pointer to ChunkStore
No
Store chunck in ChunkStore hash in hash-db
Store hash sequence for the file

Figure 25 - Deduplication process

A deduplicated file system acts the deduplication process against the whole file, to discover duplicated parts. The procedure (Figure 26 - Deduplication process) splits the file into fragments called chunks and for each chunk a hash is computed (OpenDedup uses a non-cryptographic hash algorithm named Murmurhash3 (Yamaguchi & Nishi, 2013) (APPLEBY, 2016), while Microsoft uses a cryptographic algorithm SHA256 (NIST, 2012)). All new hashes are stored in a database, and the relative chunk is stored in a chunkstore; a pointer to the position in chunkstore is saved together with the hash. The original file is transformed in a sequence of hashes; each hash is linked to the corresponding chunk. The procedure, that reconstructs original files after deduplication, is called rehydrating. Chunks that are common to multiple files are saved only once. If a very frequent chunk is lost, many files cannot be fully rehydrated. Different techniques are possible to avoid this problem. Microsoft stores multiple copies of the chunks that recur very often; OpenDedup uses SDFS file system that may use multiple nodes and spread each chunk inside more than one node.
The chunks may have a fixed length in the order of some kB (usually 32 kBe128 kB) or variable length. The use of fixed length chunks simplify hash computing, and storage is simple to organize. Using fixed length chunks, a little difference between two files generates a different set of chunks with different hashes: for example later versions of documents or source code. Variable length algorithms extract chunks using fingerprints in the text, in this case little additions to a file affect only the chunk that contains the addition. Fingerprints identification is done using Rabin fingerprints algorithm (Rabin, Fingerprinting by Random Polynomials, 1981). This algorithm uses a sliding window of a fixed number of bytes and computes a value (fingerprint) using polynomials. Using specific patterns of fingerprint values, deduplication systems cut original files in chunks. In this way, it is possible to extract common chunks in very similar files isolating the changing parts.
Deduplication is present in many products available for production usage:

- Data Domain File system (DDFS) (Zhu, Kai , & Patterson, 2008) is used in appliance of EMC DataDomain family;
- Zettabyte File System (ZFS) an open source file system originally designed by Sun Microsystems, now Oracle Corporation. ZFS implements deduplication from 2009;
- B-tree file system (BTRFS) stable from August 2014 can enable out-of-band data deduplication;
- LiveDFS (Ng, Ma, Wong, Lee, & Lui, 2011) implements inline deduplication and is designed for virtual machine storage repositories;
- OpenDedup based on SDFS is an inline file system used for backup purposes;
- Microsoft Windows 2012 file system is a post-process deduplication engine (Debnath, Sengupta, & Li, 2010).

Each implementation has proper strategies to reduce impact on system performance, and to reduce usage of memory and CPU.

5.4 Analysis

5.4.1 Low Level File System Analysis

The analysis of a real system allows acquiring and recognizing details, about characteristics of these file systems. The analysis of the structure and the acquisition of artifacts give a knowledge of how to operate. The elements analyzed here are present in all deduplicated file systems with different naming conventions and internal organizations.
The scope of the analysis is to detect a series of parameters needed to reconstruct data. Using these parameters, it is possible to infer configurations of the original systems and to run an instance of the original application and recover data automatically.
The knowledge of the structure of deduplicated file systems may help improve off-line tools. Off-line tools, like FTK Imager by AccessData (FTK, 2016) or Autopsy by Basis Technology (Autopsy, 2016), can navigate many different file systems, but considering W2012 deduplication these tools can only navigate files and folders structure, but they cannot extract the content of deduplicated files and the same happens with OpenDedup. These tools installed on a W2012 system, cannot open deduplicated files, even if they are accessible from file system.

OpenDedup

The first implementation analyzed in this paper is the SDFS file system, used in OpenDedup. OpenDedup allocates all necessary files in a regular file system using a software interface called File system in Userspace (FUSE). By means of FUSE it can write a virtual file system allocated in user storage space. When SDFS is mounted it operates as a regular file system, and deduplicated files are reached through their mount point. Unmounting the file system, it is possible to access the underlying structure (Figure 26 - SDFS Volumes structure). The basic structure of SDFS (Figure 26 - Deduplication process) is replicated for each volume.
The volume under analysis is called "\deduptest"; to navigate this file system, the starting point is the folder "\deduptest\files" (Figure 27 - SDFS files structure). In this folder, there is a replica of the folders and files structure of the mounted file system; each file and folder has metadata similar to the files in the mounted file system. The attribute relative to file size is not referred to the original deduplicated file, while the content of the file is relative to the deduplicated
storage structure. Files in this folder contain the pointers needed to support the deduplication structure. These files are pointer files.
Analyzing one of the pointer files (ex.: The Complete Works of William Shakespeare.txt) the first bytes report the creator, in our case OpenDedup (address 0x08: "org.opendedup.sdfs.io.MetaData DedupFile") and the version is at the end of the file (address 0x12E: "3.1.9"). The first two bytes in each file pointer (Table 19 - MetaData Dedup File) (stream_magic: 0xACED) are a typical marker for JavaSerialization protocol. This indicates that this file is a serialization of java data structure.
Inside this file there is the size of the original file (address 0x4A:00:00:00:00:00:55:4B:81 / 5.589.889 bytes e Table 20 - File size). Using this attribute, the metadata related to the

file system are now complete: owner, permission, path and filename are reported in the folder "\deduptest\files" while the size is inside the file pointer. Using pointers, we can rehydrate a file to its original content. To reconstruct the file we need a link to the structure that contains the sequence of chunks. Inside the file pointer there is a unique-identifier (address 0x6B: "$aac2f972-56e4-4fd5-9e1c- 8dddec187195" e Table 21 - Unique Identifier) that points to further elements in the structure.
In the folder "\deduptest\ddb" (Figure 28 - SDFS ddb structure) there is a set of two characters folders, these are the first two characters of the unique identifiers present in the file pointers. In this case, we have to look in the folder "\deduptest\ddb\aa": inside this folder there are the folders that map files relative to unique-identifiers starting with "aa". To access the map for the file under analysis the right folder is "\deduptest\ddb\aa \aac2f972-56e4-4fd5-9e1c-8dddec187195". The folder contains a file named "aac2f972-56e4-4fd5-9e1c- 8dddec187195.map". This file contains all the chunks hashes of the original file chunks. These hashes are in the proper order and each hash can be used as a pointer into the chunkstore.
The hashes sequence and the chunkstore are two elements common in all deduplication systems. The default hash algorithm in OpenDedup is murmur hash (mmh3) (APPLEBY, 2016) but this can be changed by a configuration parameter and other algorithms can be specified; Microsoft and DataDomain use SHA1/SHA256.
Mmh3 is a non-cryptographic hash algorithm simple to compute, with a reduced need of memory and computing power; it has a low collision rate, and so it is suitable for inline deduplication. The procedure computes mmh3 with a 128 bit length and with a "seed" (0x192A/decimal 6442), the value of which is written in the first two characters of each map file. The simplest way to verify mmh3 hash computing is by fixing the chunk length, splitting the file using chunk length and then computing mmh3 using the "seed" 6442.
The sequence (Table 22 - Hashes sequence) of hashes can be used to get the chunks. To discover where each chunk is saved some other steps are needed. In folder "\deduptest\chunkstore\hdb" there is a file called hashstore-sdfs-UNIQUEID (example:

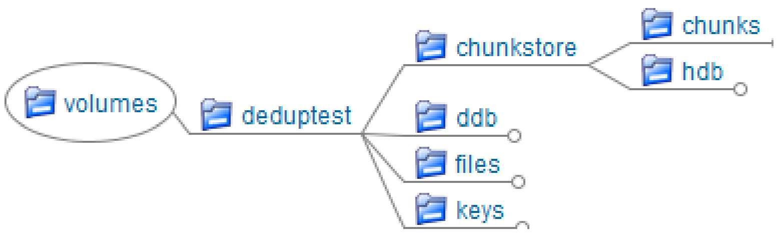


Figure 26 - SDFS Volumes structure

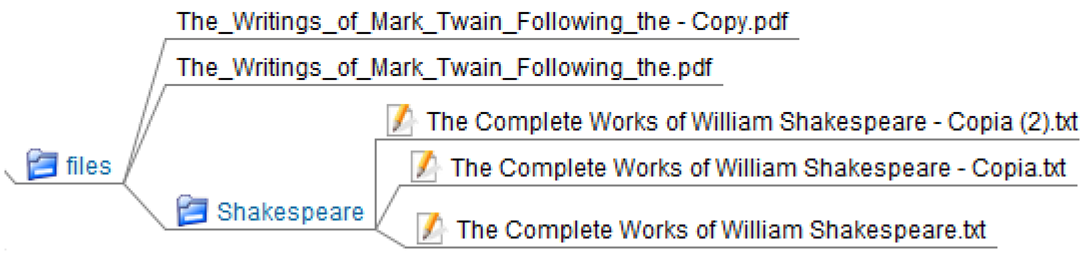

The_Writings_of_Mark_Twain_Following_the - Copy.pdf
The_Writings_of_Mark_Twain_Following_the.pdf
files
Shakespeare
The Complete Works of William Shakespeare - Copia (2).txt
The Complete Works of William Shakespeare - Copia.txt
The Complete Works of William Shakespeare.txt

Figure 27 - SDFS files structure

Table 19 - MetaData Dedup File

Hexadecimal	Text
0x0000 AC ED 00 05 73 72 00 27	¬í..Sr.'
0x0008 6F 72 67 2E 6F 70 65 6E	Org.Open
0x0010 64 65 64 75 70 2E 73 64	Dedup.sd
0x0018 66 73 2E 69 6F 2E 4D 65	fs.io.Me
0x0020 74 61 44 61 74 61 44 65	taDataDe
0x0028 64 75 70 46 69 6C 65	dupFile

Table 20 - File size

Hexadecimal	
0x0048	xx 00 00 00 00 00 55 4B
0x0050	81 xx xx xx xx xx xx xx

Table 21 - Unique Identifier

Hexadecimal	Text
0x0068 00 00 00 24 61 61 63 32	...$aac2
0x0070 66 39 37 32 2D 35 36 65	F972-56e
0x0078 34 2D 34 66 64 35 2D 39	4-4fd5-9
0x0080 65 31 63 2D 38 64 64 64	E1c-8ddd
0x0088 65 63 31 38 37 31 39 35	ec187195

hashstore-sdfs-97035D0AE223- D298-AABA-6EE996282BA8.keys) that contains as set of (key, value) pairs. The keys are the hashes, while the values are pointers in the chunkstore folder structure (Figure 29 - SDFS chunkstore structure). This file contains all the hashes computed for all the files in the deduplicated file system, and operates as an index.
The use of these pointers in the chunkstore structure (Table 23 - Chunkstore pointer) requires a simple transformation of the pointer from hexadecimal to decimal (ex: 0x7D6E755F20A339F5 / signed decimal / 9038290553609075189): now we know where to search for this chunk. In folder "\deduptest \chunkstore\chunks" there is a set of about two thousand folders. The name of these folders are something like "nnn" and "-nnn". The pointer for the chunk we are looking for, uses the first three number plus the sign: in this example, the folder we are looking for is "\deduptest\chunkstore\chunks\903" and the file "9038290553609075189.map".
The structure of this file (Table 24 - Chunkstore) is again (key, value): the key is as usual the hash, while the value is the offset in the file "9038290553609075189". This file contains the chunks and the structure is (key, length, value); the length is the chunk size plus the length of a start marker FFFF FFFF, and the value is the chunk.

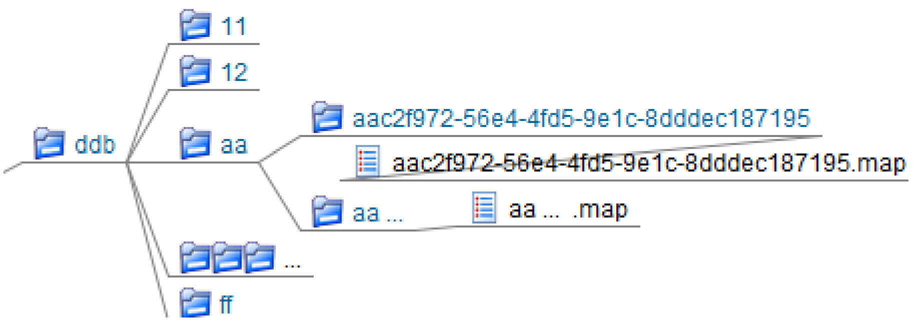

11
12
ddb
aa
aac2f972-56e4-4fd5-9e1c-8dddec187195
aac2f972-56e4-4fd5-9e1c-8dddec187195.map
aa ...
aa ... .map
...
ff
Figure 28 - SDFS ddb structure

Table 22 - Hashes sequence

Hexadecimal
0x000000 19 2A 02 00 00 00 00 00
...
0x000100 00 00 00 00 35 00 00 00
0x000108 01 2D FA E1 3F CE 15 51
0x000110 B1 9A A7 55 28 A0 E8 99
0x000118 41 00 FE 00 00 00 00 00
…
0x000160 00 35 00 00 00 01 BD D7
0x000168 C2 E3 4B C9 85 7B C0 1A
0x000170 34 CE F1 B4 28 EF 00 FE

Table 23 - Chunkstore pointer

Hexadecimal
0x162048 2D FA E1 3F CE 15 51 B1
0x162050 9A A7 55 28 A0 E8 99 41
0x162058 7D 6E 75 5F 20 A3 39 F5

Table 24 - Chunkstore

Hexadecimal	Text
0x0AD2D0 00 00 00 10 2D FA E1 3F	....-úá?
0x0AD2D0 CE 15 51 B1 9A A7 55 28	Î.Q±š§U(
0x0AD2D0 A0 E8 99 41 00 00 10 04	è™A....
0x0AD2D0 FF FF FF FF EF BB BF 54	ÿÿÿÿï»¿T
0x0AD2D0 68 65 20 50 72 6F 6A 65	he Proje
0x0AD2D0 63 74 20 47 75 74 65 6E	ct Guten
0x0AD2D0 62 67 2 65 720 45 42 6F	berg EBo

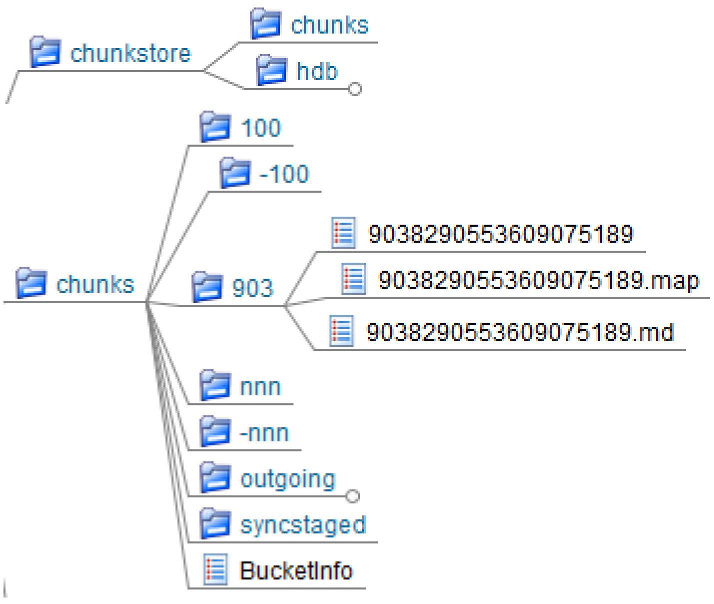


Figure 29 - SDFS chunkstore structure

Windows 2012 Deduplication

The Windows 2012 Deduplication (W2012Dedup) (PATENTSCOPE, 2012) is a feature of the file system, while OpenDedup is a file system in userspace. This implies that the analysis has to access the underlying structure of the file system. To analyze the file system, the tools used are FTK Imager and Autopsy; by means of these tools, it is possible to access all the elements of the NTFS structure.

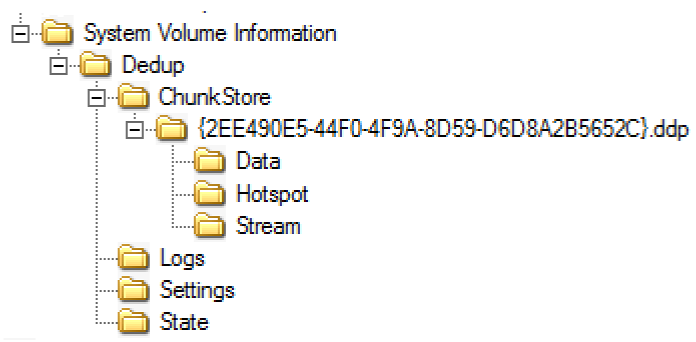

System Volume Information
Dedup
ChunkStore
{2EE490E5-44F0-4F9A-8D59-D6D8A2B5652C}.ddp
Data
Hotspot
Stream
Logs
Settings
State

Figure 30 - Windows 2012 System Volume Information

The file system stores all deduplication data under the "System Volume Information" (Figure 30 - Windows 2012 System Volume Information); this hidden folder contains the chunkstore structure. The chunkstore contains three elements, the Stream container, the Data container and the Hotspot container. The first two elements are common in deduplication. The Stream contains the hashes sequences; the Data contains the chunks, while the Hotspot is an element added by Microsoft to store most common or frequently used chunks; in this last container, there is a controlled level of redundancy.
The analysis of a Windows 2012 file system starts from the Master File Table ($MFT) and $MFT information are stored in little endian. The $MFT entry relative to a deduplicated file contains information about chunkstore. These data are saved in a "Reparse Points" attribute ($REPARSE_POINT e 0xC0). The function of the "Reparse Point" attribute is to act as a collection of user-defined data. Microsoft or third party software can use this attribute for specific applications. When a reparse point is present, the system needs a specific filter to parse this attribute.
Reparse Point (Table 25 - Reparse Point) starts with the NTFS attribute type 0xC0; in our example, the full length of this section is 0x00A0. The length of the original file is written at the relative offset 0x28 (Len 4 bytes); at offset 0x38 (Len 16) there is the unique identifier of ChunkStore ({2EE490E5-44F0-4F9A-8D59-D6D8A2B5652C}.ddp). Inside this folder, we have the "Data" and "Stream" folders. Inside the Data folder, there are .ccc files that are chunks containers, while inside the Stream folder the .ccc files contain the hashes sequences for the deduplicated files. At offset 0x78 (Len 30) there is the sequence of bytes that are the stream header; this value identifies the stream of a particular file in the stream container.
In the Stream folder, a .ccc file has three type of sections: "Cthr" called file header, "Rrtl" or redirection table, "Ckhr" or stream map element. The syntax of the file is described in Code 3 - Grammar of Stream file

Table 25 - Reparse Point

Address	Haxadecimal content
+0x00	C0 00 00 00 A0 00 00 00
0x08	00 00 00 00 00 00 03 00
0x10	84 00 00 00 18 00 00 00
0x18	13 00 00 80 7C 00 00 00
0x20	01 02 7C 00 00 00 00 00
0x28	16 8F 09 00 00 00 00 00
0x30	00 00 00 00 00 00 00 00
0x38	E5 90 E4 2E F0 44 9A 4F
0x40	8D 59 D6 D8 A2 B5 65 2C
0x48	40 00 40 00 40 00 00 00
0x50	F5 F4 B2 C1 6E B0 D1 01
0x58	01 00 00 00 00 00 01 00
0x60	00 50 00 00 01 00 00 00
0x68	01 00 00 00 08 05 00 00
0x70	C8 01 00 00 00 00 00 00
0x78	9C FC 06 75 EB 4E D1 0C
0x80	FD 13 F3 14 AA 1D B1 D3
0x88	8C BA 9C 19 E2 EF D5 12
0x90	50 58 CE B1 FB 58 05 00
0x98	C1 AD 45 7A 00 00 00 00
0xA0	

```
<Stream Container> ::= <file header> <redirection table> <stream maps>
<stream maps> ::= <stream map> <stream maps> | <stream map>
<stream map> ::= <stream header> <metadata> <hash values>
<hash values> ::= <hash value> <hash values> | <hash value>
```

Code 3 - Grammar of Stream file

The data we are looking for is in Ckhr (0x 43 6B 68 72) sections. Each Ckhr section (Table 26 - Ckhr entry in Stream container) contains the full sequence of hashes relative to a file also called “stream map”; each section reports the stream header at offset relative 0x38. Starting from global offset 0x30 and each 64 (0x40) bytes, there is a new hash section. At offset 0x70 starts the first hash (sequence 0x01), at 0x78 there is the absolute position in the chunkstore (0x5000), and at offset 0x88 the hash (len 32). The value at offset 0xA8 is the length of the chunk payload (0xA7ED).

Table 26 - Ckhr entry in Stream container

Address	Haxadecimal content
+0x00	43 6B 68 72 01 03 03 01
…	…
0x30	00 00 00 00 00 00 00 00
0x38	9C FC 06 75 EB 4E D1 0C
0x40	FD 13 F3 14 AA 1D B1 D3
0x48	8C BA 9C 19 E2 EF D5 12
0x50	50 58 CE B1 FB 58 0F 27
0x58	EB 47 3C 95 A2 30 E5 A5
0x60	77 51 A6 31 DF FF CB 71
0x68	53 6D 61 70 01 04 04 01
0x70	01 00 00 00 01 00 00 00
0x78	00 50 00 00 01 00 00 00
0x80	2E 5E 01 00 00 00 00 00 ED
0x88	DB 30 58 FA 7F 5C 19
0x90	5C 89 FD 23 FE 97 FA 43
0x98	58 B2 99 B4 FF 6B 40 6C
0xA0	0B 8A BE 27 49 BB 28 7A
0xA8	ED A7 00 00 00 00 00 00

The last file to analyze is the chunks container .ccc in the "Data" folder. The syntax of the file is described in Code 4 - Grammar of chunk container

```
<Chunk Container> ::= <file header> <redirection table> <data chunks>
<data chunks> ::= <data chunk> <data chunks> | <data chunk>
<data chunk> ::= <chunk header> <chunk data>
```

Code 4 - Grammar of chunk container

Here are stored the chunks, jumping to the position indicated in the "Ckhr entry" (0x5000) there is a the first "Ckhr" entry and after a few bytes (0x5C) starts the chunks content for the length indicated again in this file at offset 0x0C (0xA7ED) (Table 27 - Data chunk in Chunk container).
Following the sequence as reported in the stream map, all the chunks in a file can be retrieved and the rehydration of the whole file can be accomplished. If chunks are compressed, before being concatenated, they have to be deflated. When a file is deleted, the $MFT entry $REPARSE_POINT is cleared, but the chunk hashes sequence, and the chunks in the chunk container are preserved until the first "garbage collection job" runs. The chunks may be compressed, depending on the system configuration, and some types of file are excluded from compression because they already contain compressed data. The compression used (sometimes called LZNT1þ) is very similar to LZNT1 (Microsoft Corporation, 2016) (Microsoft Corporation, 2015) . LZNT1 is a Microsoft algorithm inspired to LZ77 (Lempel & Ziv, 1977). The difference between LZNT1 and this compression algorithm is that the flag bytes are 32 bits long (4 bytes) instead of 16 bits (2 bytes) used by LZNT1. The syntax is described in Code 5 - Grammar of chunk compression.

```
<compressed chunk> ::= <Flag group>
<Flag group> ::= <Flag data> <Flag group> | <Flag data>
<Flag data> ::= Flag-byte <data block>{1-32}
<data block> ::= Char | Len-displacement
```

Code 5 - Grammar of chunk compression

There is no official documentation about this element, but this compression algorithm seems an evolution of LZNT1.

Table 27 - Data chunk in Chunk container

Address	Haxadecimal content	
0x5000	43 6B 68 72 01 03 03 01	Ckhr....
	01 00 00 00 ED A7 00 00	
	01 00 28 00 08 00 00 00	
	08 00 00 00 08 00 00 00	
	02 00 00 00 00 00 00 00	
	ED DB 30 58 FA 7F 5C 19	
	5C 89 FD 23 FE 97 FA 43	
	58 B2 99 B4 FF 6B 40 6C	
	0B 8A BE 27 49 BB 28 7A	
	5D 1A 7C 25 A5 A8 E7 CF	
	32 B8 58 6B BB 92 4C 9D	
	00 00 00 00 50 72 6F 6A	....Proj
	65 63 74 20 47 75 74 65	ect Gute
	6E 62 65 72 67 27 73 20	nberg's
	4C 61 20 44 69 76 69 6E	La Divin
	61 20 43 6F 00 10 00 00	a Co....
	6D 6D 65 64 69 61 20 64	mmedia d
	69 20 44 61 6E 74 65 2C	i Dante,

W2012Dedup hash algorithm outputs values 256 (32 bytes) bits long. According to documentation (Program, 2016), many Microsoft protocols use SHA-1, for example Windows Distributed File System (DFS) replication (MS-FRS2) and Remote Differential Compression Algorithm (MS-RDC). In this case, the length of the hash value (256 bits) indicates another algorithm. To verify this hypothesis we tested some algorithms but without the knowledge of the padding strategy, it is difficult to identify the algorithm: the best hypothesis is SHA-256.

5.4.2 Forensic Analysis

Analysis carefulness

The seizure of storage devices containing a deduplicated file system is an activity that requires some carefulness. During a seizure, we need to check the presence of deduplication in storage devices and if it is present, there are a few solutions applicable. The first is to seize the whole system and not only the deduplicated storage. The second is to conduct a live forensic analysis and extract the documents of interest for investigation (if known) on-site. The third method is to write down all installation details and replicate the same configuration in laboratory, because to recover a deduplicate volume we can mount it using an operating system that runs the same configuration as the original installation.
However, if during seizure no check was done for the presence of deduplication, we have to extract information directly from storage devices. In this case, recovering of the volume requires a little more effort. The first step is to recognize the file system type; the second is to infer the configuration parameters. We must pay specific attention at data carver's results, because at the date we wrote this article, popular data carvers do not recognize the presence of a deduplicated file system, and do not know how to rehydrate original files. This article is a beginning of investigation of these file systems, to improve awareness about the problem.

OpenDedup

The previous explanation of how this file system works, gives us the way to recover it using the chunks repository and the hash sequences. Usually a deduplicated file system is used to support backup of huge quantity of data. The idea to manually rehydrating a file system is nonsensical, but a clear understanding of the process is the basis to create procedures to automate the process. The direct analysis of the storage support is reserved to recovering of corrupted volumes. The fundamental elements in recovery procedure are the chunkstore and the relative hash sequences. To see if the volume under analysis contains a working file system, we can analyze the structure of the deduplicated file system. It allows checking integrity of data; the integrity check is done using the information available in the chunkstore. The chunkstore contains a set of ordered pairs (hash, chunk), and we can use the algorithm murmurhash3 and the “hash-key” (default 0x6442 in case of default configuration) to verify integrity of chunks. To verify that all the chunks are present, we must use the hash sequences. This procedure gives a granularity of check corresponding to the chunk size.
If we have a well-functioning OpenDedup volume, we can install the same version of the file system and configure it to mount the volume under analysis. The configuration

requires parameters inferable from data present in the volume. The basic parameters are chunk type and size, the hash algorithm, the "hash-key", the position of the hash-db and of the chunkstore. To identify chunk type and size we can analyze the chunks length: if all chunks have the same size, the chunk type is "fixed-size" and the chunk size is easily computed, while if chunks have different length the chunk type is "variable-size" and "min-size" and "max-size" need to be estimated analyzing all the chunks. The hash-key is in the first byte of all the map files. The hash-type (or hash algorithm) can be detected using the length of the hash value, the hash value, the chunk content, and the hash-key. We must compute the hash value of the chunk using the possible algorithms and identify the right one; the default algorithm is murmurhash3.

Windows 2012 R2

Windows 2012 uses a post-process deduplication method. Files are first stored in the file system as regular files, and only after a configured period (parameter: fileMinimumAge) are processed for deduplication. After deduplication, the file system removes the original files. However, until the disk area is overwritten, the artifacts of deleted files remain on the volume. Since they were regular files, they can be recovered using a data carver.
W2012 does not deduplicate all the files: it filters files according to "exclude Filextensions Default" configuration parameter, that indicates which file extensions are excluded. The excluded files are saved as regular files and no deduplication is applied. Other files are deduplicated and stored in the volume as previously explained.
W2012 stores chunks in a compressed form, but compression is not applied to all files, there is a configuration parameter that excludes the compressed formats (parameter: noCompressionFileExtensions, default values: asf, mov, wma, wmv, ace, arj, bhx, bz2, cab, gz, gzip, hpk, lha, lzh, lzx, pak, pit, rar, sea, sit, tgz, z, zip, zoo). The excluded files are deduplicated, but chunks are not compressed. These files can be recovered concatenating all chunks as they are in the chunkstore, following the order specified in the stream map; no deflate process is required after chunks extraction.
The simplest method, to recover a well-functioning W2012 deduplicated volume, is to mount it on a system with the same operating system version with the deduplication engine enabled.
Tools like FTK Imager or Autopsy can analyze many different file systems, reading directly the file system data structure. However, when you try to read a deduplicated file system, it starts from $MFT of the deduplicated volume, reads all the entry, shows files and folders with their metadata and when it tries to inspect the content of the files, they result empty. This happens because these tools do not know how to use reparse point information. Therefore, in case of a damaged device, we must recover the fundamental files that are the chunk-container and the stream-container; these two elements are the bearing structure of the chunkstore. Then, following a stream map, we can concatenate chunks to compose a whole file. When a file is rehydrated, if the $MFT is available, we can recover the file name and metadata, otherwise we can analyze header and structure of the file and recognize the file type.
Considering a non-deduplicated volume, when we delete a file, a data carver can recover the whole file, until the allocated space is overwritten. A deduplicated volume instead splits files in chunks, and stores each chunk in a repository. When a file is removed from

a deduplicated volume, the entry in $MFT is immediately removed, but the stream map and the chunkstore remains unchanged. Therefore, immediately after deletion, it is possible to recover a file.
A deduplicated volume runs a optimization process regularly, but a "regular" optimization has no effects on stream maps and chunk repositories. Only when a garbage collection (GC) job runs, it removes the chunks that are part of deleted elements from the chunkstore. A "regular" GC deletes only part of the unreferenced chunks from the chunkstore, while a "full" GC eliminates all traces of deleted files in deduplicated volume structure. Nevertheless, analyzing unallocated space after a GC we can find artifacts left by this process. During GC, the system creates a new version of the stream container and the chunk container, and then deletes the previous version of the stream container and the chunk container files, but they remain for a while in the file system as artifacts. We can recover fragments of stream map and chunk container, and sometimes whole files. To recognize stream-container and chunk-container, we can use their internal structure reported in Table 28 - Stream container format and Table 29 - Data container format. The particular structure of files that support a deduplicated file system gives high confidence to the recovered file, because the stream map is a sequence of hashes, and so it is possible to verify the completeness of the chunks repository.

The importance of Hash-db

Suppose you recover a chunk container, without the hash sequences, and the chunks are not compressed. Without the knowledge of chunks concatenation sequence, it is impossible to do an accurate reconstruction because of the deduplication algorithms used to create chunks. An efficient chunking strategy uses a system based on the Rabin algorithm (Rabin, 1981). This algorithm locates the point where to break a file creating chunks; the localization of cut points happens where the original file shows a predefined "fingerprint". When systems use this algorithm to process files containing documents like contracts or invoices, the resulting chunks are very similar, because these documents are usually based on models, and their chunks can be concatenated to generate files never existed in the original file system. The existence of a flawless hash sequences container is the only way to be sure of the accuracy of file reconstruction.

Table 28 - Stream container format

Address	Haxadecimal content	
0x000000	43 74 68 72 01 04 04 01	Cthr....
0x000020	last stream entry	
	43 74 68 72 01 04 04 01	Cthr....
0x001000	52 72 74 6C 01 03 03 01	Rrtl....
0x002000	52 72 74 6C 01 03 03 01	Rrtl....
0x003000	52 72 74 6C 01 03 03 01	
	. . .	

Table 29 - Data container format

Address	Haxadecimal content	
0x000000	43 74 68 72 01 04 04 01	Cthr....
0x000020	last stream entry	
…	43 74 68 72 01 04 04 01	Cthr....
0x001000	52 72 74 6C 01 03 03 01	Rrtl....
0x002000	52 72 74 6C 01 03 03 01	Rrtl....
0x003000	52 72 74 6C 01 03 03 01	
end of	. . .	
file	FF FF FF FF FF FF FF FF	

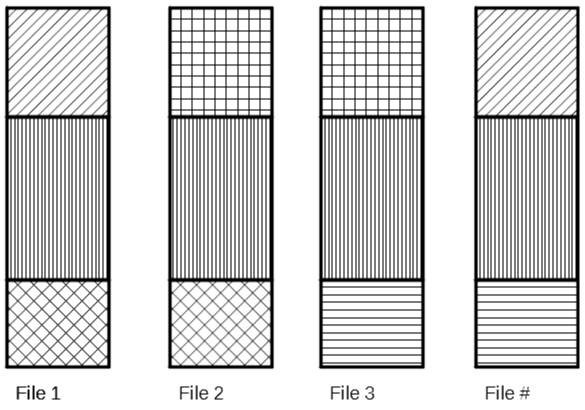


Figure 31 - Recovering using chunks without hashes sequence

To test this hypothesis, we used the first twelve pages of “Alice's Adventures in Wonderland”. The first file in Figure 32 - Recovering using chunks without hashes sequence-File 1 contains the first twelve original pages. This file is the smallest file that W2012Dedup splits in three chunks. We modified the starting line of the text and created a second file Fig. 23-File 2. To create the third file we modified the ending line of the second file Fig. 23-File 3. After creation of these three files, we converted them in PDF format, and to simplify the test we changed the extension from “.pdf” to “.ace” to avoid chunks compression. Then we copied them in the deduplicated volume. The system broke the three files generating chunks has depicted in Fig. 23-File 1 2 3 (same pattern indicates identical chunk). The file # in Fig. 23 can be composed using the chunks of “file 1” “file 2” “file 3”, and obtaining a new valid file. The chunking based on Rabin algorithm uses the output of a polynomial function, and cuts the files where a fixed trend is present. This generates accurate cuts; the same hash in the central part of the three files proves the precision of these cuts. Exploiting this property, you can concatenate chunks, to create a new well-formatted file, but this file was not present in the original volume.
The procedure we used to generate files to demonstrate this hypothesis is very similar to the procedure used to create documents in a company, where employees start always from the same document model to create new documents, and then they modify only part of header and part of the body (examples are invoices and contracts).
The problem exposed, enforces the rule that we must have the hash sequences to rehydrate files present in the chunkstore. The hash sequences are a crucial element of a forensics acceptable reconstruction of a deduplicated file system.

5.5 Conclusion

New storage technologies need to be investigated from a forensic point of view, because manufacturers rarely give detailed documentation about low-level implementation. Storage technologies knowledge is central in digital forensic analysis and this work gives a first look inside deduplication implementations.
This paper addresses deduplication technologies, it analyses a post process deduplication (windows-server-2012, 2016) and an inline implementation (Silverberg, 2016). The knowledge of deduplication implementations helps to identify the presence of this kind of file system on devices, and to recover files and folders content from them. In case of

damaged devices or file systems, this work proves that, without the structures containing the hash sequences, the file reconstruction can generate never-existed files.
The hash sequences are crucial for a “forensically sound” recovery. Future works will analyze from a forensic point of view other implementations of storage deduplication and storage technology.
I presented this work at DFRWS/IMF 2017 EU in Germany and the relative article is printed on Digital Investigation 20 (Lanterna & Barili, 2017).

6. The Future of Fragments: Virtual Desktop Infrastructures

The growing usage of virtual desktop infrastructures requires a validation of traditional model for investigation in digital forensics (Freiling, Glanzmann, & Reiser, 2017). The fragments are under different layers of virtualization, to reach them it is required the analysis of a whole infrastructure.
The investigation of a virtual desktop infrastructure yields a virtual crime scene using all traces left from virtual desktop usage.
The virtual desktop delivered to a user is assembled when the user log in. The data that belongs to a user is stored in a shared virtual storage. The users with the same profile share the operating system and the applications. The temporary files and other elements of the virtual desktop are not permanent; the system deletes these elements when a new virtual desktop sessions starts. This chapter analyses a virtual desktop infrastructure in order to identify traces left from users. Many elements used to create VDI[47] services are well known, but key elements such as provisioning services or delivery services need to be investigated. The delivery of a virtual desktop leaves many traces on the servers that compose the infrastructure; these traces need to be correlated.
The virtual disks delivered to the desktops are composed of multiple parts, each part has proper volatility level, investigation has to focus on two elements called differential disk and personal virtual disk, and these elements contain only data pertaining to user session.
In this work, you will find following topics: description of a virtual desktop architecture layer; analysis of the elements from a forensics point of view; analysis of different configuration impact, volatility concern in investigations, organization of global and user storage space.
These infrastructures can be used to commit crimes, or can be target of a crime, and the knowledge of how they work if essential for investigations. Some configuration of VDI, are targeted to deliver a service with minimal resources and to obtain this results all temporary data and logs are deleted after usage, and the storage space is allocated to new users. This implies that is very difficult to identify traces in case of investigations (Nirbhay & Narayanan, 2011).
Thanks to University of Pavia, I can operate on an in-house virtual desktop infrastructure. The infrastructure is based on VMWare vSphere as hypervisor, and on Citrix product XenDesktop and XenApp for Virtual desktops delivery.

In the 1960's IBM developed a wide range of computer systems, each generation was different from the previous, and so customers need to change their software to satisfy new system requirements. Computer was not capable to support more than one user at a time, and task were scheduled using batch script. In response to this situation, IBM built a new mainframe the CP-40 (only for research purpose inside IBM laboratory) and CP-67 with operating system CP[48]/CMS[49]. CP created the virtual machine environment, explicitly

[47] VDI stands for Virtual Server Infrastructure
[48] CP stands for Control Program
[49] CMS stands for Cambridge or Conversational Monitor System

dividing memory and other resource among the users. CMS was the virtual machine operating system. Another step in computer virtualisation was done in 1987, Insignia Solutions developed SoftPC, and it was an emulator of a PC, capable of running Dos applications under UNIX workstation. In 1998, a company called VMWare began to sell a program called VMWare workstation. This company is now the market leader in virtualization. In 2001, VMWare released a product called ESX Server that run on bare metal and allow user to deploy their virtual machine, this technology is referred as Type -1 Hypervisor, and do not require an operating system to run. It runs Virtual Machines using server hardware directly.
Today's Virtual Desktop Infrastructure (VDI) permits to run a desktop operating system in a centralized virtual infrastructure. This idea is similar to original IBM's idea. Giving users a separated virtual operating environment to work in. The desktop virtualization delivery requires a whole network infrastructure; it is complex than server virtualization.

The VDI give company a series of advantages, virtual desktops can run in Cloud Services. We refer to "Cloud Service" or "Cloud Computing" when the resource is delivered over the Internet, instead of being delivered in a local server farm[50]. Using cloud computing also little company and individuals can use any type of resources without the need of expensive and complex infrastructure. Typical type of cloud services are: data storage space; webmail; social networks; online business applications. Cloud computing is a model for convenient, on-demand access to computing resources, the name of this model is IaaS[51]. Digital investigation about resources in cloud depends on the level of visibility that we obtain. The analysis presented here requires the access to the whole infrastructure.

The server virtualization (VSI[52]) is the software partitioning of a physical server into many small virtual server to maximize usage of physical server resources. The desktop virtualization is a technology that runs operating system and applications of an end user in a centralized environment, separating the desktop from the user client device.
The workload of VSI depends on the service provided by the servers, while on VDI depends on the users and their applications. The storage for VSI is huge and requires high speed SAN, VDI requires rather limited storage quantity for each user, but there are many users, this requires a different storage organization. There are three type of storage spaces required for each user. The operating system requires a read-only fast-access storage; user's documents resides on network-shared space, write-cache requires limited space but very fast access.
A VSI deliver services on the network, internal users of virtual server are system administrator. VDI users require interaction with many remote peripherals of different type. The network bandwidth is high predictable for VSI, while is unpredictable for VDI. VSI access requires virtual console or remote desktop and use administrative rights, VDI access is made using different device with a specific protocol by many different users with limited ICT knowledge (a sophisticated access control is a prerequisite for a secure usage).

[50] The model that runs application on a local farm is called "on-premise".
[51] IaaS stands for Infrastructure as a service
[52] VSI stands for Virtual Server Infrastructure

The VSI as well as physical server needs rarely to reboot, while virtual desktop reboots each time a session is closed.
The VSI infrastructure needs backup and archive policy to restore data and service without data loss, the VDI requires only user's documents preservation. Each virtual server is different and highly customized, while all virtual desktops are clones of a single instance (the template).

6.1 Virtual Desktop Infrastructures

VDI centralizes delivery of end-user working environment in the datacenter. In this way, VDI increases security, control, reliability, efficiency and scalability. VDI is composed of a number of services that run on virtual servers. The VDI addresses the need to work on mobile device as phone or tablet, granting security and performance. Centralizing computing power, storage, and applications deploy, it creates a more controlled environment and simplifies management.
The provisioning of the services to the end user consists of the following phases:

1. The user connects to a portal;
2. The user make identification and authentication;
3. The system lists available desktop/services for user profile;
4. The user selects the desktop and the connection continues using a connection client;
5. The delivery controller starts the virtual desktop according to the selected profile, after the desktop boots, it connects the desktop to the user session.
6. The provisioning service provide the disk image;
7. The DHCP sets the IP address;
8. The desktop during login procedure arrange the working environment, according to defined user policy;

Each step sends log entries in service log repository. The infrastructure extracts users' configuration from a database, each transaction is traced in transactions log. To reach the infrastructure many network devices are crossed, and each device logs connection events. Correlating all the events, we can compose the whole scene. VDI can be analysed according to the schema in" Figure 32 - VDI Layer".

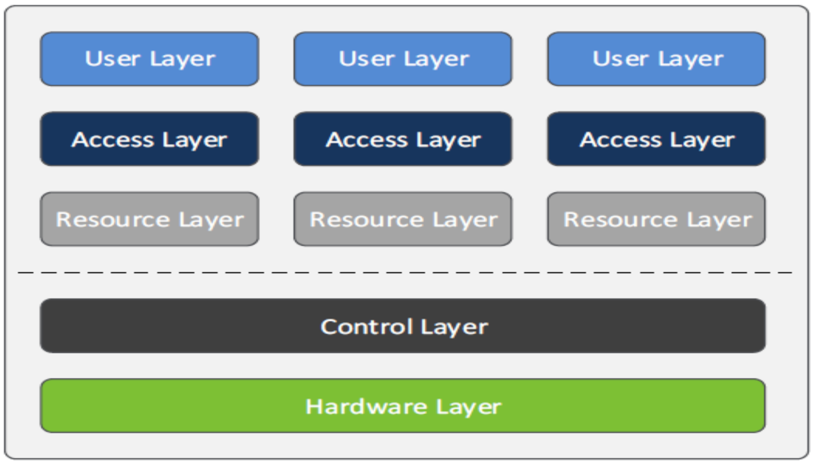


Figure 32 - VDI Layer

User layer defines the procedures to get access to VDI according to the user physical device. Devices used to connect to the infrastructure can be a corporate device or personal device, and typically, people use thin client, personal computer and mobile device (tablet or smartphone).
The communication protocols used to establish connections are SSL (SSL, Freier, Karlton, & Kocher, 2011) and Citrix ICA (CITRIX, 2016). The ICA protocol uses TCP/IP and RTD/UDP. ICA works in different network topology environment, for example mobile device network. Low network bandwidth and low computational workload are needed thanks to intelligent redirection, adaptive compression and data deduplication. The adaptive compression uses HDX[53] protocol; it defines which codecs are used in different network traffic conditions, as well as it acts an intelligent use of CPU and GPU resources. A forensic network analysis is complex due to all this technologies combined together. Ica[54] is a full protocol used from Citrix to deliver XenApp[55] and XenDesktop[56] services. This protocol sends input from the client (keyboard, mouse, mic, webcam) to the remote server, and receive output (video, sound, printer directives). All traffic flows over common TCP/IP network connections. Citrix Receiver[57] handles client side protocol and sets proper handling for encoding, display attributes, audio attributes, compression, and encryption.
The user layer manages client device peripherals. By the use of HDX, these devices terminate locally, in this manner webcams, printers and scanner can interact using native USB speed.
Analysing user devices, we can find traces of the usage of a VDI. The configuration of installed application gives information about the target infrastructure. When an application runs, generate log and temporary file.

The access layer provides access to the environment and connects central resources to the user session. The level of access to the resource respects the security policy applied to the user and to its location. The access includes identification and authentication, all data transfers use protection protocol based on encryption and encapsulation. These layers acquires data for service access delivery, and is responsible for authentication and identification, their logs are the most important in forensic analysis of the infrastructure.

The resource layer controls access to the resources. The resources are desktop and applications, each resource allocates to a defined user group; there are multiple images of operating system deliverable to the users. Resources can be delivered using different technology (XD, 2016) (Desktop-Transformation, 2016): Machine Creation Services, or Provisioning Services (PVS, 2016) (CITRIX-XD7, et al., 2016) (CITRIX, 2016) (VMWARE, 2016).
There are two types of applications delivering techniques (Broker-Concepts, 2016): installed, streamed. This layer is defines the level of personalization of a VD and the need of a personal vDisk (vDisk, 2016). This layer is important in forensics analysis, because it

[53] HDX stands for High Definition eXperience.
[54] ICA stands for Independent Computing Architecture.
[55] XenApp delivers applications, user can use applications that run in a centralized environment using its own devices.
[56] XenDesktop refers to the Citrix VDI.
[57] Citrix Receiver is the client application used to connect to the virtual infrastructure.

defines user's folders management policy, profiles handling, and basic settings. According to its profile, the user receives a write cache and personal virtual disk. Write cache or personal virtual disk save all data modified during a user session. Write cache deletes at each reboot of virtual desktop, personal virtual disk is persistent and delivered to a specific user each time he uses the infrastructure. The user profile defines folder redirection used to provide the user a personal environment for documents archiving.
VDI provides resources as follow:

1. The operating system is on a common image, each virtual desktop uses the same image provided by MCS. If provisioning is done in pooled-random, no temporary user activity artifact is preserved. The user receives a virtual desktop at logon and the virtual desktop returns in the pool after user log-off. Any changes made to the desktop lasts until the session is active.
2. Using policy setting of Active Directory, user folders redirect to shared folders in a file server. Trace of user activities are in network share.
3. In case of application delivered via XenApp, applications artifacts are preserved in the virtual application server.

To capture data during an active user session, VDI provides snapshot function. The snapshot of a virtual desktop fixes data for further analysis, a snapshot preserves disk content and memory contents.

The control layer is organized in access controller, delivery controller and infrastructure controller:

The Access Controller provides the list of users and their enabled resources. After a successful user authentication, it contacts the delivery controller to receive the list of available resources, the list of available resource delivered to the user.
The access controller consists of two elements: netscaler gateway, storefront and Load balancer. The netscaler gateway encapsulates all SSL traffic; storefront authenticates remote users using the Active Directory service; the load balancer distributes connections on the infrastructures. (CITRIX, 2016)

Delivery controller receives information about the user and it creates a list of resources granted to the user. The delivery process brokers the connection between user and resources. The delivery process start-ups, shutdowns, and registers virtual desktops. The controller always update the DB server about status of resources and user activities.

The infrastructure controller integrate the remaining elements of the infrastructure, these elements are DNS, DHCP (DHCP, 2016), Active directory, SQL servers, File server and Licensing server.
The Control Layer logs evidence about the infrastructure, each element has its own log:

1. DHCP: each virtual desktop when starts, requires an IP address, each request is logged using the time date, IP delivered, and request parameters;
2. Active directory: when users login, the authentication process logs username, time, and date. It is possible to configure log verbosity to obtain more information;
3. Desktop delivery: it delivers resources to the users, creates the working set and logs all events to SqlServer.

The hardware layer contains physical implementation of virtual desktops. The storage is the most important element, and its performances impacts all the services. Therefore, performance and storage size must be appropriate to deliver a good user experience. Physical servers supports computational workload, local storage, CPU power and network. To speed-up virtual desktops RAM is used as storage for temporary data such as write cache (Write-cache, 2016), the volatility of this element is extreme, only a snapshot fixes this data for further analysis.

6.2 Analysis path

VDI is composed of multiple servers (Figure 33- VDI infrastructure); the traces left from users are multiple. As in a traditional infrastructure, we have to search for file, temporary files, and event logs.
The Domain Controller (Active directory, DHCP, DNS) keeps logs accessible through Event Viewer as in windows operating systems. The Domain Controller gives many information; the most important for forensic analysis is the folder redirection configuration, using this information we can discover where user saves permanent data. A path for analysis about a known user, starts from Event Viewer (TECHNET-MS, 2016) of Domain Controller (Windows Log, Security), here are stored users profile. Information about user connecting device and originating IP can be found correlating information given by access controller (StoreFront or Netscaler gateway). The logs of the delivery controller give information about the assigned virtual desktop, and the shared folders (this information is stored in domain controller, but it is used also from delivery controller). The virtual desktops can be Pooled Desktop or Dedicated Desktop. In case of Pooled Desktops, the traces about temporary activities are discarded at reboot, the reboot starts immediately after user log-off from a virtual desktop. In case of Dedicated Desktops all traces left from user activities are preserved, the analysis of a virtual desktop in this configuration is similar to analysis of a physical desktop.
XenDesktop controller logs configuration changes and administrative activities (operations and activities such as PowerShell operations, session control, and messaging) initiated by an administrator from Studio Director or from a PowerShell. Events Log contains configuration changes such as creation, editing, deleting, assigning of machine-catalogues or host resources, as configured in VDI policies through Studio Director (CITRIX-XD7, et al., 2016). File server contains logs about users handling of files and folders. The file servers keeps all traces about files in their file systems, the analysis can be accomplished using traditional carving techniques.
Database server is a core element of the infrastructure; it stores the running configuration of all desktops.
Storefront and NetScaler contains information about connection from client to the internal VDI resource. Storefront uses Microsoft IIS[58] web server, log are in IIS format.
Virtual machine are allocated in a datastore , using function of VMware hypervisor can be created snapshot or clone of running machine, so running situation can be fixed for further analysis (VMWARE, 2016). Snapshot function can save also the memory contents, and allows an analysis of running process, without alter the desktop under analysis.

[58] IIS stands for Internet Information Server

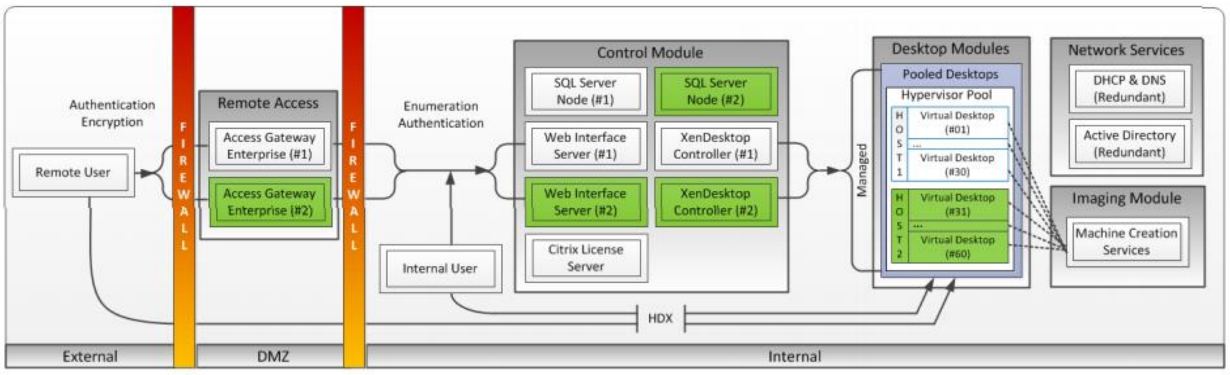


Figure 33- VDI infrastructure

The storage of a virtual desktop (Figure 35 - Structure of a virtual desktop disk (Image source VMware site)) is composed of a base disk (Base), a personal virtual disk (PvD) and a differential disk (Diff). The user activities can be located in PvD and in the differential disk.
The base disk (Base) contains the original image of the operating system; this image is delivered to all the Virtual Desktop. The base disk has usually no forensic interest if investigation affects the user of a virtual desktop. The base disk is usually a snapshot taken from reference virtual disk. The only cases that requires analysis of this part of the infrastructure is when the crime is committed by a system administrator, or the user has used Advanced Evasion Techniques to overtake limitation of a virtual desktop.
The personal virtual disk (PvD) contains all the information that the user can modify and that are persistent through the sessions. The PvD contains all the files created or modified by the user. The PvD in the past was organised on a block base, while in the recent infrastructures is organised on file base. The analysis of this disk is simpler than the analysis of a traditional hard disk, because it has a limited size, and all the content is related to the owner user. The removal of a file lives artifacts; carving the PvD, we can retrieve these artifacts.
The differential disk (Diff) contains all the information that are not persistent through the sessions. The delivery process delete this information at each reboot. The differential disk is stored in an area of a shared storage dedicated to this type of content. The analysis of storage area dedicated to differential disk, gives results that are difficult to correlate to VDI users, furthermore the content of this area has an high order of volatility, because this area is used by all the powered on virtual desktop instance. The analysis of differential disk area gives many fragments of temporary files, they arise from activity of operating system and applications, but they are difficult to correlate to the generating users.
The only secure way to capture content of differential area is when a virtual desktop is running. The VDI has tools that can provide a snapshot or a clone of the virtual machine delivered to the user. These tools are not usually used because a crime is prosecuted time longer after a session closes.
The storage used by virtual infrastructure is a shared storage infrastructure. The direct analysis of the storage area using traditional tool is rather complex (Figure 35 - Structure of a virtual desktop storage infrastructure). The core element of this infrastructure are

the hosts, this are physical computer, that have lots of computing power (server multi-socket and multi-core) and a very huge quantity of RAM memory. These servers use very little internal disks. A host server runs a program called hypervisor that can execute multiple virtual machines; the virtual machines can be servers or desktops.
The virtual machines are stored in files of type vmdk[59]. Since each host can run any virtual machine, these files are stored in repository shared between hosts.
The host are connected to the datastore by a network infrastructure called SAN[60], each SAN has multiple separated path to connect the hosts to storage. The devices that creates SAN paths from hosts to storage compose the fabric.

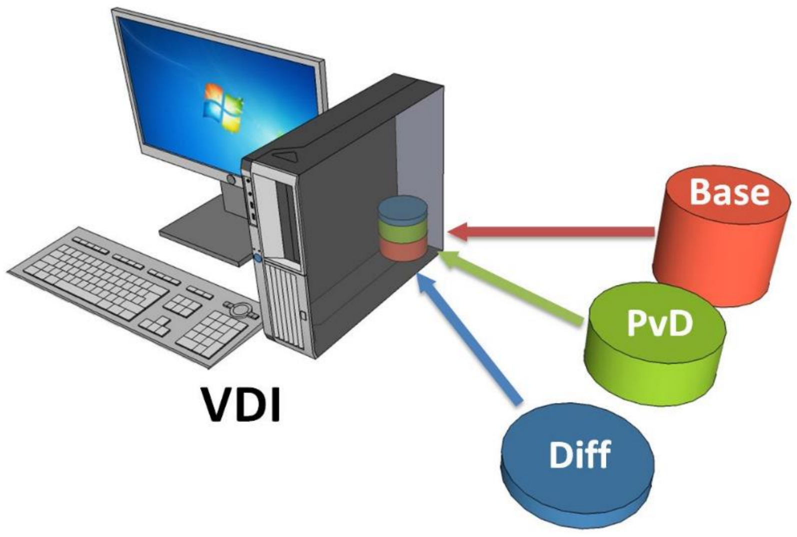


Figure 34 - Structure of a virtual desktop disk (Image source VMware site)

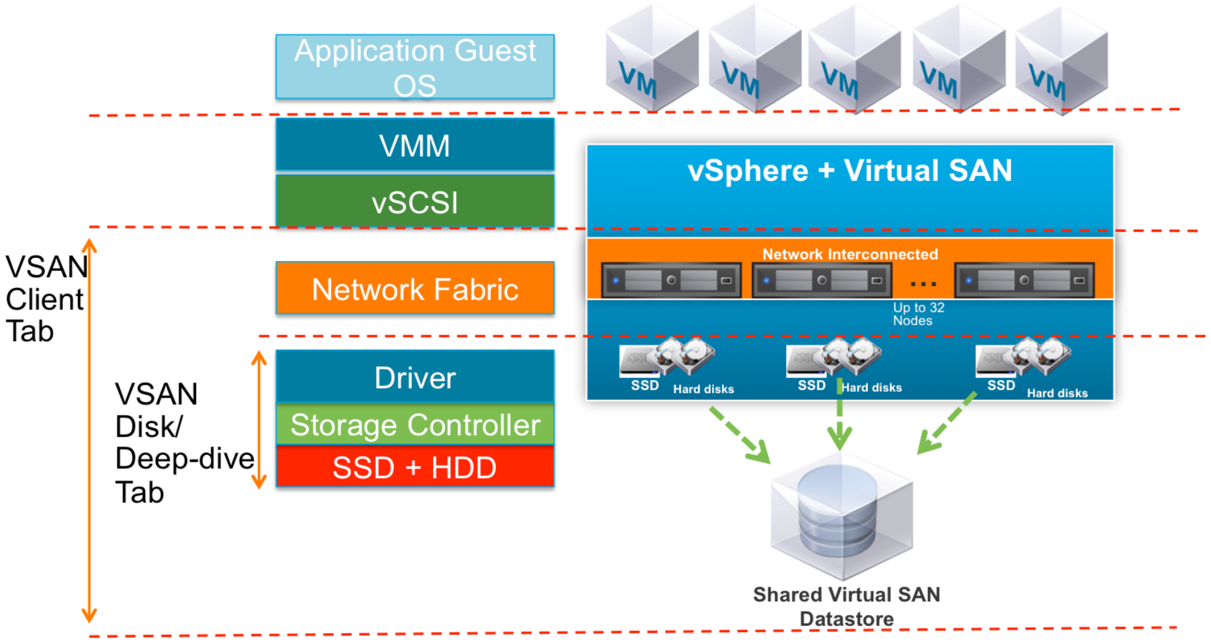


Figure 35 - Structure of a virtual desktop storage infrastructure (Image source VMware site)

[59] VMDK stands for Virtual Machine DisK, if a format used from VMware products.
[60] SAN stands for Storage Area Network.

The repository refers as a datastore; each datastore can contain multiple virtual machines. The datastore allocates in a LUN of a storage disk array. To guarantee data security disk array use a RAID technology. LUN are software defined by the storage array. The way to analyse a structure like this is using function delivered by the element of infrastructure: flashcopy, snapshot, and clone. Trying to analyse directly physical disk give no usable results, the structure is so complex that data extracted cannot be linked to the user that generate it.

6.3 VDI Considerations

This analysis of a virtual desktop infrastructure requires access to all the elements. The availability of the whole infrastructure made it simple to understand the role of each elements. I had access to all the component of the structure, I have analysed configuration repository and logs repository, to see traces left by users activity in different configurations. This is not always possible when resources are delivered in a cloud service instead of in an on premise service. The importance of this study is that it depicts the big picture of virtual desktop infrastructure.
I showed the path of information inside the infrastructure, and the traces left from user usage. I discussed about the resources delivery, paying special attention to storage delivery.
Post crime digital forensic analysis, focuses attention on the analysis of the physical storage supports. This analysis in virtual environments is more complex, and is possible only if the desktop is delivered as “Dedicated desktop”. The virtual disk is composed by multiple part, which resides in different storage space. The knowledge of the architecture configuration makes it possible to identify all the part that compose a user virtual disk.
Virtual desktop infrastructure has to take care of investigation, especially when delivered by cloud services. Without the access to detailed logs is very hard to make a forensic analysis of a crime committed using this type of infrastructure.

7. Final Remarks

The track I followed in Ph.D. researches comes from my experience. I started with the analysis of fragments extracted from a magnetic device using a carver. In the same period, I collaborated with the police for investigation about a crime committed in a utility company, inside their structure, data was stored in a centralized file server, and their backups was in a deduplicated storage.
I work as a technician in the server farm of the University of Pavia and three year ago, we installed a VDI infrastructure, and the question was automatic: how can I analyse the disk of a virtual desktop?
These experiences pave the way for Ph.D. researches. The evolution in storage technologies requires in depth studies. Fragments are present in new storage technologies, but fragments coming from new modern devices requires new methods for their analysis.
The research topics for my Ph.D. are fragments and their analysis, fragments in deduplication technologies, fragmented traces in virtual desktops infrastructure.

This work shows that fragments analysis need to be continuously improved. Fragments already exist in new storage technology and new computing technology delivery, but they generate fragments in a different way. To handle effectively these fragments we need to know how they are generated. Only the knowledge of this technology gives the possibility to make good analysis.
The study of previous research shows that there is an open problem in the classification of fragments, if they contains mixed format or mixed languages. My work proposed the usage of grammar analysis. I investigated on the usage of different lexical analyser and on the analysis of the resulting tokens distribution. Then I focused on the usage of grammar induction method and the analysis of the resulting rules.
The result of a grammar analysis is set of complex attributes. That can be analysed using token frequency distribution, and clustering fragments using grammar attributes.
I focused on grammar attributes extraction applying different strategies. The first was the use of a detailed lexical analyser; the high number of tokens recognizable produced some problems of lexical ambiguity. The second strategy was based on the restriction of the context, this choice produced on grammar construct applicable to specific formats, but this determinist approach entailed the knowledge of the context or the test of different contexts. The last strategy was based on a restricted set of tokens and grammar constructs, the tokens selected were common to many different formats. The last strategy produce effective results in clustering analysis.
I used grammar induction on the results produced by a restricted tokenizer. The grammar induction extracts sets of grammar rules from each fragments.

The second research topic was deduplication. This infrastructure splits files in fragments to save space. But these fragments do not follow the well-known logic of blocks storage allocation, these fragments are cut using algorithms that recognise particular sequence (fingerprint), these fragments create problem to traditional carvers. When a carver operates on a storage that contains a deduplicated file system, it cannot extract a whole file, even if the file system is intact. Some storage technologies use special architecture, to extract data from these file system we need to know how it operate. My research highlights the real internal organization of two common elements, the chunkstore, and the hash sequence. It identifies how the elements are organised in two real cases

(OpenDedup and Windows Deduplication). I made a reverse engineering of the Microsoft implementation of deduplication, displaying the file organization.
The Windows Deduplication feature works off-line, and file are stored in regular file system before deduplication post process operates, this leaves artifact that can be analysed by carvers. The deleting process requires some action to remove all the traces of a deleted file, and a file can be recovered until a complete garbage collection is done in the deduplication repository.
I demonstrated that is possible to use fragments (chunks) generated in deduplication process to create file formally correct, but never existed in the file system. This result is important to fix a rule in deduplicated file system analysis, "It is impossible to assert that a recovered file really existed in a deduplicated file system, without a coherent a hash sequence".

The last research topic was dedicated to virtual desktop infrastructure (VDI). The VDI infrastructure deliver a virtual desktop to users, application and user data are stored in a virtual disk. The fragments analysis in these structures need a good knowledge of their working principle. A virtual desktop stores in separated repository the base operating system, applications, data and temporary files. The disk delivered to the user as a single disk is composed by the infrastructure managing different parts, and each part has its own storage policy. The forensic analysis has to detail, how each part can be correlated to a specific user. The study of the infrastructure discloses traces left by users, the usage of a virtual desktop leaves traces in the network devices, in the access control services, in the delivery servers, not only in the virtual disk.

References

Axelsson, S. (2010). The Normalised Compression Distance as a file fragment classifier. *Tenth Annual DFRWS Conference - Digital Investigation.* Portland, Oregon: Digital Investigation.

Kamiya, T., Kusumoto, S., & Inoue, K. (2002). CCFinder: a multilinguistic token-based code clone detection system for large scale source code. *Software Engineering, IEEE Transactions* (p. vol.28, no.7, pp.654-670). IEEE doi: 10.1109/TSE.2002.1019480.

AccessData. (2016). *Forensic Tool Kit (FTK) Imager.* Tratto da http://accessdata.com/product-download/ digital-forensics/ ftk-imager- version-3.4.2

Ahmed El-Shimi, R. K. (2012). Primary Data Deduplication - Large Scale Study and System Design. *Microsoft Corporation, 2012 USENIX Annual Technical Conference.*

APPLEBY, A. (2016). *Murmur Hash 3.* Tratto da https://github.com/aappleby/smhasher/blob/master/src/MurmurHash3.cpp

Appleby, A. (s.d.). *MurmurHash3.* Tratto da https://github.com/aappleby/smhasher /blob/master/src/MurmurHash3.cpp

Autopsy. (2016). *Autopsy by Basis Technology.* Tratto da http://www.sleuthkit.org/autopsy/

Beazley, D. (2016). *PLY (Python Lex-Yacc).* Tratto da dave@dabeaz.com: http://www.dabeaz.com/ply/ply.html

Broker-Concepts. (2016). *Citrix Broker - Concepts* . Tratto da http://support.citrix.com/proddocs/topic/citrix-broker-admin-v2-xd7/about_broker_concepts-xd7.html

Brown, R. (2012). Finding and identifying text in 900+ languages. *The Proceedings of the Twelfth Annual DFRWS Conference - Digital Investigation 9*, S34–S43.

C.C.P. (2016). *Italian Criminal Procedure Code.* Tratto da http://www.brocardi.it/codice-di-procedura-penale/

Carlton, G. H. (2011). A survey of contemporary enterprise storage technologies from a digital forensics perspective. *The Journal of Digital Forensics, Security and Law: JDFSL 6*, JDFSL 6, no. 3 - 63.

Carlton, G., & Matsumoto, J. (2011). A survey of contemporary enterprise storage technologies from a digital forensics perspective. *The Journal of Digital Forensics, Security and Law*, JDFSL 6, no. 3 (2011): 63.

Carlton, G., & Matsumoto, J. (2011). A survey of contemporary enterprise storage technologies from a digital forensics perspective. *The Journal of Digital Forensics, Security and Law: JDFSL 6, no. 3*, 63.

Carrier, B. (2005). *File System Forensic Analysis.* Addison-Wesley Professional.

Casey, E. (2000). *Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet.* Orlando, FL, USA: Academic Press, Inc.

Center, N. -N. (September 2013). *Crime Scene Investigation - A Guide for Law Enforcement.* U.S.Department of Justice.

Chung, H., Park, J., Lee, S., & Kang, C. (November 2012). Digital forensic investigation of cloud storage services. *Digital Investigation, Volume 9, Issue 2*, Pages 81-95, ISSN 1742-2876, http://dx.doi.org/10.1016/j.diin.2012.05.015.

Chung, T.-S., Park, D.-J., Park, S., Lee, D.-H., Lee, S.-W., & Song, H.-J. (2009). A survey of Flash Translation Layer. *Journal of Systems Architecture*, Volume 55, Issues 5–6, May–June 2009, Pages 332-343, ISSN 1383-7621, http://dx.doi.org/10.1016/j.sysarc.2009.03.005.

CITRIX. (2016). *ICA Settings Reference*. Tratto da http://support.citrix.com/proddocs/topic/ica-settings/ica-settings-wrapper.html

CITRIX. (2016). *specification programmers guide*. Tratto da citrix: https://www.citrix.com/static/cdn/archivedsdks/simulsdk/9.x/sim_api_specification_programmers_guide.pdf

CITRIX-XD7, Feller, D., Baker, A., Berger, T., Meesters, R., Brooks, M., . . . Ben-Chanoch, A. (2016). *Citrix XenDesktop 7 Blueprint, Citrix XenDesktop Handbook, An architect's guide to desktop virtualization.* Citrix Enterprise Consultant.

Cohen, W., Ravikumar, P., & Fienberg, S. (2003). A comparison of string distance metrics for name-matching tasks.

Commerce, U. D. (2012). *NIST FIPS PUB 180-4, Secure Hash Standard (SHS).* NIST FIPS .

Corpora, D. (2015). *Corpora*. Retrieved from Web Site: Digital Corpora: http://digitalcorpora.org/corpora/

D. Harnik, B. P.-P. (2010). Side Channels in Cloud Services: Deduplication in Cloud Storage. *IEEE Security & Privacy*, vol. 8, no. 6, pp. 40-47, Nov.-Dec. 2010. doi: 10.1109/MSP.2010.187.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Commun. ACM 7*, 171-176 doi=http://dx.doi.org/10.1145/363958.363994 .

David E. Bernstein, & D. Jackson. (2004). The Daubert Trilogy in the States. *Jurimetrics*, 44.

Debnath, B. K., Sengupta, S., & Li, J. (2010). ChunkStash: Speeding Up Inline Storage Deduplication Using Flash Memory. *USENIX annual technical conference.*

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., & Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science (1986-1998) Sep 1990; 41, 6; ABI/INFORM Global*, 391.

Desktop-Transformation. (2016). *Desktop Transformation - Pooled Desktops - Reference Architecture*. Tratto da http://support.citrix.com/servlet/KbServlet/download/28856-102-665173/Desktop%20Transformation%20-%20Pooled%20Desktops%20-%20Reference%20Architecture.pdf

DHCP. (2016). *Analyze DHCP Server Log Files*. Tratto da http://technet.microsoft.com/en-us/library/dd183591(v=ws.10).aspx

Dumais, S. T. (2004). Latent semantic analysis. *Ann. Rev. Info. Sci. Tech., 38*, 188–230. doi:10.1002/aris.1440380105.

EEG-COE. (18 March 2013). *Electronic evidence guide - A basic guide for police officers, prosecutors and judges.* Strasbourg, France: CyberCrime@IPA, EU/COE Joint

Project on Regional Cooperation against Cybercrime, Data Protection and Cybercrime Division, Council of Europe.

El-Shimi, A., Kalach, R., Kumar, A., Oltean, A., Li, J., & Sengupta, S. (2012). Primary Data Deduplication - Large Scale Study and System Design. *2012 USENIX Annual Technical Conference.*

Eoghan Casey, & Gerasimos J. Stellatos. (2008). The Impact of Full Disk Encryption on Digital Forensics. *SIGOPS Oper. Syst. Rev. 42* (p. 93-98). DOI=http://dx.doi.org/10.1145/1368506.1368519 .

Fitzgerald, S., Mathews, G., Morris, C., & Zhulyn, O. (2012). Using NLP techniques for file fragment classification. *Digital Investigation 9* , S44–S49.

FOREMOST. (2016). *Foremost*. Tratto da Jesse Kornblum and Kris Kendall: http://foremost.sourceforge.net/

Freiling, F., Glanzmann, T., & Reiser, H. (2017). Characterizing loss of digital evidence due to abstraction layers. *Digital Investigation, Volume 20, Supplement, March 2017*, Pages S107-S115, ISSN 1742-2876, https://doi.org/10.1016/j.diin.2017.01.012.

FTK. (2016). *Forensic Tool Kit (FTK) Imager by AccessData*. Tratto da http://accessdata.com/product-download/digital-forensics/ftk-imager-version-3.4.2

Garfinkel, S. (2007). Carving contiguous and fragmented files with fast object validation. *Digital Investigation*, 2007 DFRWS, doi:10.1016/j.diin.2007.06.017.

Garfinkel, S. P. (2009). Bringing science to digital forensics with standardized forensic corpora. *Digital investigation*, S2-S11.

Garfinkel, S., Nelson, A., White, D., & Roussev, V. (2010). Using purpose-built functions and block hashes to enable small block and sub-file forensics. *Digital Forensic Research Workshop 2010.* Portland, Oregon: Digital Investigation - doi:10.1016/j.diin.2010.05.003.

GARFINKEL-METS. (2016). *Carver taxonomy*. Tratto da http://forensicswiki.org/wiki/File_Carving

Hall, G. A., & Davis, W. (2006). Sliding Window Measurement for File Type Identification. *IEEE information assurance workshop 2006.* IEEE.

Harnik, D., Pinkas, B., & Shulman-Peleg, A. (2010). Side Channels in Cloud Services: Deduplication in Cloud Storage. *IEEE Security & Privacy*, vol. 8, no. 6, pp. 40-47, Nov.-Dec. 2010. doi: 10.1109/MSP.2010.187.

J. Min, D. Y. (2011). Efficient Deduplication Techniques for Modern Backup Operation. *IEEE Transactions on Computers*, Vol. 60, no. 6, pp. 824-840 doi: 10.1109/TC.2010.263.

Karresand, M., & Shahmehri, N. (2006). File Type Identification of Data Fragments by Their Binary Structure. *Information Assurance Workshop, 2006 IEEE* (p. vol., no., pp.140,147, 21-23 June 2006, doi: 10.1109/IAW.2006.1652088). West Point, NY: IEEE.

Kiran, R., Shang, H., Toyoda, M., & Kitsuregawa, M. (2015). Discovering Recurring Patterns in Time Series. *EDBT 2015, 18th International Conference on Extending Database Technology.* Tokyo, Japan: EDBT .

Kullback, S., & Leibler, R. (1951). On Information and Sufficiency. *Ann. Math. Statist. 22*, (p. no. 1, 79--86).

Lakhotia, A., Dalla Preda, M., & Giacobazzi, R. (2013). Fast location of similar code fragments using semantic 'juice'. *Proceedings of the 2nd ACM SIGPLAN Program Protection and Reverse Engineering Workshop (PPREW '13)* (p. Article 5 , 6 pages. DOI=http://dx.doi.org/10.1145/2430553.2430558). ACM, New York, NY, USA,.

Lämmel, R. (2008). Google's MapReduce programming model — Revisited. *Science of Computer Programming*, Volume 70 Pages 1-30, ISSN 0167-6423, http://dx.doi.org/10.1016/j.scico.2007.07.001.

Lanterna, D., & Barili, A. (2017). Forensic analysis of deduplicated file systems. *DFRWS 2017 Europe d Proceedings of the Fourth Annual DFRWS Europe* (p. Digital Investigation 20 - S99-S106). Lake Constance, Germany, 21 - 23 March 2017: Elsevier.

Lavoie, T., & Merlo, E. (2012). An accurate estimation of the Levenshtein distance using metric trees and Manhattan distance. *Software Clones (IWSC), 2012 6th International Workshop* (p. pp.1-7). IWSC doi: 10.1109/IWSC.2012.6227861.

Lempel,]. J. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory* (p. vol. 23, no. 3, pp. 337-343). IEEE.

Lempel, A., & Ziv, J. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337-343 - doi: 10.1109 /TIT.1977.1055714.

Levenshtein, V. (1996). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, vol. 10, pp. 707–10.

Li , W.-J., Wang, K., Stolfo, S., & Herzog, B. (2005). Identifying file types by n-gram analysis. *Proceedings from the 6th Annual IEEE System, Man and Cybernetics Information Assurance Workshop.* SMC.

Li, Y., Lin, J., & Oates, T. (2012). Visualizing Variable-Length Time Series Motifs. *SIAM Conference on Data Mining* (p. pages 895-906). SDM.

Linux-NTFS. (2016). *Linux NTFS*. Tratto da https://flatcap.org/linux-ntfs/ntfs/index.html

Mark Reith, Clint Carr, & Gregg Gunsch. (2002). An Examination of Digital Forensic Models. *International Journal of Digital Evidence - Department of Electrical and Computer Engineering Graduate School of Engineering and Management Air Force Institute of Technology Wright-Patterson AFB, OH 45433-7765*, Volume 1, Issue 3.

McDaniel, M., & Heydari, M. (2003). Content based file type detection algorithms. *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference* (p. pp.10). doi: 10.1109/HICSS.2003.1174905.

MFT. (2016). *Mater File Table*. Tratto da https://msdn.microsoft.com/en-us/library/bb470206(v=vs.85).aspx

Michael MATES NATO Parliamentary Assembly. (April 2001). *Technology and terrorism.* United Kingdom: SUB-COMMITTEE ON THE PROLIFERATION OF MILITARY TECHNOLOGY - http://www.nato-pa.int.

Microsoft. (2016). *Mater File Table.* Tratto da https://msdn.microsoft.com/en-us/library/bb470206(v=vs.85).aspx

Microsoft Corporation. (2015). *MS-XCA - Xpress Compression Algorithm - v20151016.* Microsoft Corporation.

Microsoft Corporation. (2016). Tratto da Technet: https://blogs.technet.microsoft.com/filecab/2012/05/20/ introduction-to-data-deduplication- in- windows-server-2012/

Microsoft Corporation. (2016). *Introduction to data deduplication.* Tratto da https://blogs.technet.microsoft.com/filecab/2012/05/20/introduction-to-data-deduplication-in-windows-server-2012/

Microsoft Corporation. (2016). *MS-XCA - v20151016. Xpress Compression Algorithm.* Microsoft Corporation.

Min, J., Yoon, D., & Won, Y. (2011). Efficient Deduplication Techniques for Modern Backup Operation. *IEEE Transactions on Computers, vol. 60, no. 6*, 824-840 doi: 10.1109/TC.2010.263.

Min, J., Yoon, D., & Won, Y. (2011). Efficient Deduplication Techniques for Modern Backup Operation. *IEEE Transactions on Computers*, vol. 60, no. 6, pp. 824-840 - doi: 10.1109/TC.2010.263.

Moffat, A., & N. Jesper Larsson. (1999). Offline dictionary-based compression. *Data Compression Conference, Proceedings. DCC '99 , 29-31 Mar 1999, doi: 10.1109/DCC.1999.755679*, (p. pp.296-305).

Moody, S., & Erbacher, R. (2008). SÁDI - Statistical Analysis for Data Type Identification. *Proceedings of the 3rd IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering* (p. pp. 41-54). Oakland, CA: IEEE .

Mukasey, N. -M., Sedgwick, J. L., & Hagy, D. W. (Apr. 2008). *Electronic Crime Scene Investigation: A Guide for First Responders, Second Edition.* National Institute of Justice NCJ 219941.

Muthitacharoen, A. B. (2001). A low-bandwidth network file system. *ACM SIGOPS Operating Systems Review*, (p. Vol. 35, no. 5, pp. 174-187).

Muthitacharoen, A., Chen, B., & Mazieres, D. (2001). A low-bandwidth network file system. *ACM SIGOPS Operating Systems Review* (p. vol. 35, no. 5, pp. 174-187). ACM.

Neuner, S. M. (2016). Effectiveness of file-based deduplication in digital forensics. *Security and Communication Networks.*

Neuner, S., Mulazzani, M., Schrittwieser, S., & Weippl, E. (2015). Gradually Improving the Forensic Process. *Availability, Reliability and Security (ARES) - 10th International Conference* (p. pp. 404-410). Toulouse: ARES - doi: 10.1109/ARES.2015.32.

Neuner, S., Schmiedecker, M., & Weippl, E. (2016). Effectiveness of file-based deduplication in digital forensics. *Security and Communication Networks.*

Nevill-Manning, C., & Witten, I. (1997). Identifying Hierarchical Structure in Sequences:A linear-time algorithm. In H. N. Department of Computer Science - University of Waikato, *Journal of Artificial Intelligence Research 7* (p. 67–82). Hamilton, New Zealand: AI Access Foundation and Morgan Kaufmann Publishers.

Ng, C.-H., Ma, M., Wong, T.-Y., Lee, P., & Lui, J. (2011). Live Deduplication Storage of Virtual Machine Images in an Open-Source Cloud. *ACM/IFIP/USENIX 12th International Middleware Conference.* Lisbon, Portugal, December 2011.

Nguyen, K., Tran, D., Wanli Ma, & Sharma, D. (2014). The impact of data fragment sizes on file type recognition. *Natural Computation (ICNC), 2014 10th International Conference* (p. pp.748,752). ICNC.

Nirbhay, J., & Narayanan, A. (2011). Organisational preparedness for hosted virtual desktops in the context of digital forensics. *Australian Digital Forensics Conference - 2011.*

Nishi, F. Y. (2013). Hardware-based hash functions for network applications,. *19th IEEE International Conference on Networks (ICON)*, (p. pp.1-6). Singapore.

NIST. (2012). *FIPS PUB 180-4 - Secure Hash Standard (SHS).* U.S. Department of Commerce.

NIST. (2016). *Digital Evidence and Forensics*. Tratto da http://www.nij.gov/topics/forensics/evidence/digital/Pages/welcome.aspx

NIST. (2016). *Forensic Sciences: Types of Evidence*. Tratto da http://www.nij.gov/topics/forensics/evidence/Pages/welcome.aspx

Park, J., Hyunji, C., & Sangjin, L. (2012). Forensic analysis techniques for fragmented flash memory pages in smartphones. *Digital Investigation*, Digital Investigation 9, no. 2 (2012): 109-118.

PATENTSCOPE. (2012, 05 24). *164. (WO2012067805) SCALABLE CHUNK STORE FOR DATA DEDUPLICATION*. Tratto da PATENTSCOPE: https://patentscope.wipo.int/search/en/detail.jsf;jsessionid=27950A5A2339C6A87EAB6BA0F2829DC4.wapp2nA?docId=WO2012067805&recNum=164&maxRec=4648&office=&prevFilter=&sortOption=&queryString=%28PA%2Fmicrosoft%29+&tab=PCTDescription

Penrose, P., Macfarlane, R., & Buchanan, W. (2013). Approaches to the classification of high entropy file fragments. *Digital Investigation.*

PHOTOREC. (2016). *TestDisk*. Tratto da Christophe Grenier: http://www.cgsecurity.org/wiki/TestDisk_Download

Poisel, R., & Tjoa, S. (2013). A Comprehensive Literature Review of File Carving. *Proceedings of the 8th International Conference on Availability, Reliability and Security*, 475-484.

Powers, D. M. (2007). *Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation.* Flinders University • Adelaide • Australia: School of Informatics and Engineering.

Program, M. O. (2016). *Microsoft Open Specifications Program*. Tratto da https://msdn.microsoft.com/en-us/library/dd208104.aspx

PVS. (2016). *PVS Write Cache Sizing & Considerations*. Tratto da http://blogs.citrix.com/2011/10/06/pvs-write-cache-sizing-considerations/

Rabin, M. O. (1981). *Fingerprinting by Random Polynomials.* Center for Research in Computing Technology, Harvard University. Tech Report TR-CSE-03-01. Retrieved 2007-03-22.

Rabin, M. O. (1981). *Fingerprinting by Random Polynomials.* Tech Report TR-CSE-03-01. Retrieved 2007-03-22: Center for Research in Computing Technology, Harvard University.

Roussev, V., & Quates, C. (2013). File fragment encoding classification - An empirical approach. *Digital Investigation 10*, http://dx.doi.org/10.1016/j.diin.2013.06.008.

S. Neuner, M. M. (2015). Gradually Improving the Forensic Process. *Availability, Reliability and Security (ARES), 2015 10th International Conference*, (p. pp. 404-410.). Toulouse.

S., K. (1959). *Information Theory and Statistics.* John Wiley & Sons.

Sallis, P., Aakjaer, A., & MacDonell, S. (1996). Software forensics: old methods for a new science. *Computer and Information Science* (p. pp.481-485, doi: 10.1109/SEEP.1996.534037). Computer and Information Science - University of Otago, Dunedin, New Zealand: IEEE Computer Society Press.

Sam, S. (s.d.). *http://opendedup.org/ - https://github.com/ /opendedup /sdfs.* Tratto da OpenDedup: http://opendedup.org/ - https://github.com/ /opendedup /sdfs

SCALPEL. (2016). *Scalpel is an open source data carving tool.* Tratto da Golden G. Richard III; Lodovico Marziale: https://github.com/sleuthkit/scalpel

Sengupta, S., & Benton, J. (2012). Primary Data Deduplication in Windows Server 2012. *Microsoft Corporation Redmond, Storage Developer Conference - Santa Clara 2012.*

Senin, P., Lin, J., Wang, X., Oates, T., Gandhi, S., Boedihardjo, A., . . . Lerner, M. (2014). GrammarViz 2.0 - A Tool for Grammar-Based Pattern Discovery in Time Series. In *Machine Learning and Knowledge Discovery in Databases - Volume 8726 of the series Lecture Notes in Computer Science* (p. pp 468-472). Berlin Heidelberg: Springer Berlin Heidelberg - http://dx.doi.org/10.1007/978-3-662-44845-8_37.

Silverberg, S. (2016). *OpenDedup.* Tratto da OpenDedup: http://opendedup.org/ - https://github.com/opendedup/sdfs

Spitkovsky, V., Alshawi, H., & Jurafsky, D. (2012). Bootstrapping Dependency Grammar Inducers from Incomplete Sentence Fragments via Austere Models. *JMLR: Workshop and Conference Proceedings 2012, 11th International Conference on Grammatical Inference.* JMLR.

Sportiello, L., & Zanero, S. (2011). File Block Classification by Support Vector Machine. *Availability, Reliability and Security (ARES), 2011 Sixth International Conference* (p. pp.307,312). doi: 10.1109/ARES.2011.52.

SSL, Freier, A., Karlton, P., & Kocher, P. (2011). *The secure sockets layer (SSL) protocol version 3.0.* Tratto da RFC SSL-3.0: (http://www.rfc-base.org/rfc-6101.html)

TECHNET-MS. (2016). *Active Directory Diagnostic Logging.* Tratto da TECHNET: http://technet.microsoft.com/en-us/library/cc961809.aspx

Technology, B. (2016). *Autopsy.* Tratto da Autopsy: http://www.sleuthkit.org/ autopsy/

The Apache Software Foundation. (2016). *Apache Hadoop.* Tratto da Apache.org: http://hadoop.apache.org/

Tor, P. (s.d.). *The Onion Routing.* Tratto da The Onion Routing: https://www.torproject.org/

Ullman, J. D., Aho, A., Sethi, R., & Lam, M. (2006). *Compilers: Principles, Techniques, and Tools, Second Edition, "Purple Dragon Book".* Pearson Education, Inc.

Underwood, W., & Laib, S. (2012). *Attribute Grammars and Parsers for Chunk - Based Binary File Formats - ICL/ITDSD Technical Report 12-01.* Georgia Tech Research Institute - Atlanta, Georgia: Information Technology and Decision Support Division, Information and Communications Laboratory, .

vDisk. (2016). *Personal vDisk XenDesktop*. Tratto da http://support.citrix.com/proddocs/topic/personal-vdisk-7x/c_1-personal-vdisk-best-practices.html

Veenman, C. (2007). Statistical Disk Cluster Classification for File Carving. *Third International Symposium on Information Assurance and Security 2007 - IAS 2007* (p. pp.393-398). IAS.

VMWARE. (2016). *Understanding virtual machine snapshots in VMware ESXi and ESX*. Tratto da http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1015180

windows-server-2012. (2016). *Introduction to data deduplication in introduction to data deduplication in windows server 2012*. Tratto da https://blogs.technet.microsoft.com/filecab/2012/05/20/introduction-to-data-deduplication-in-windows-server-2012/

Winkler, W. E. (1990). String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. *Proceedings of the Section on Survey Research Methods* (p. 354–359). American Statistical Association.

Write-cache. (2016). *Understanding Write-Cache in Provisioning Services Server*. Tratto da http://support.citrix.com/article/CTX119469

XD. (2016). *XD - Modular Reference Architecture*. Tratto da http://support.citrix.com/servlet/KbServlet/download/30706-102-697507/XD%20-%20Modular%20Reference%20Architecture.pdf

Yamaguchi, F., & Nishi, H. (2013). Hardware-based hash functions for network applications. *19th IEEE International Conference on Networks (ICON)* (p. pp. 1-6). Singapore: doi: 10.1109/ICON.2013.6781990.

Zhu, B., Kai , L., & Patterson, H. (2008). Avoiding the Disk Bottleneck in the Data Domain Deduplication File System. *Fast*, vol. 8, pp. 1-14.

Zoubek, C., Seufert, S., & Dewald, A. (March 2016). Generic RAID reassembly using block-level entropy. *Digit. Investig. 16*, S44-S54. DOI=http://dx.doi.org/10.1016/j.diin.2016.01.007.

Figure Index

Equation Index

Table Index

Code snippets Index