RAMS
Reliability, Availability,
Maintainability, and Safety

This is the Title of my Thesis

Your Name

December 2012

PROJECT THESIS

Department of Production and Quality Engineering

Norwegian University of Science and Technology

Supervisor 1: Professor Ask Burlefot

Supervisor 2: Professor Fingal Olsson

Contents

Chapter 1

Introduction

File recovery from digital data storage devices has been a hot topic among the Digital Forensics field. Traditional data storage devices make use of file systems, in order to manage contained data, their available space and to maintain location of files. When the storage device and its file system are intact, it is quite simple to recover data from them. This is mainly because file systems make use of metadata in order to track the metadata for their files. Meta-data can contain information such as creation date, data structure (e.g directory or regular file), file type, file owner, size, last modified date and more. In practice, most data can be recovered using the regular file system, but often investigators are specifically interested in the data that appears to be missing. In a real life forensic case, it is likely that a part of file systems metadata might be corrupted or deleted.

1.1 File Carving

File carving is a forensics technique that recovers files based on their content, without relying on their meta-data. File carving process involves two steps. File format validation and file reconstruction [1]. During the recovery procedure, forensics investigators must first validate the type of the file and then apply the appropriate reconstruction technique. At this point we should note that in this thesis, only the file format validation techniques are of our interest. By examining the byte-content and/or the structure of an unclassified file, file format validation techniques are used to classify its type. The Magic Number Matching technique [3] looks for magic numbers, specific byte sequences that signal the beginning and/or the end of a file(headers,footers) and try to classify them to a file type according to that information. For example jpeg files begin with the hexadecimal sequence "FFD8" and end with "FFD9" [4]. Similarly, the Data Dependency Resolving technique is used to identify fields that specify file attributes like color or size [1].

Furthermore, other file carving techniques use statistical learning algorithms, which process the complete byte set of a file, creating a representative fingerprint for every file type. A classifier compares these fingerprints with an unidentified byte sequence and produce an accuracy level for each fingerprint. Then, it classifies the unidentified byte sequence with the file type of the fingerprint that yielded the highest accuracy level. Some common statistical learning techniques are the n-Gram Analysis [5] and the Byte Frequency Analysis(BFA) and the Byte frequency cross-correlation(BFC) algorithms [**?**].

1.2 Problem Formulation

The aforementioned techniques have some profound weaknesses. The Magic Number Matching and the Data Dependency Resolving approaches make general type classification infeasible. This is due to the fact that not every file-type contain such characteristic structures [2]. Furthermore, n-Gram Analysis and both BFA and BFC were designed to be applied in a complete file or a pre-defined part of it, which retains all of its content. Hence, they depend on files overall internal structure and characteristics.

So why this is a problem? The answer lies in file systems behaviour and file fragmentation. When we delete a file from a data storage device, the data are not actually removed. The sectors in which the file was stored still contain the same data, although the file system marks them as unallocated [4]. That means the next time a new file is created, the file system is free to use these sectors, which are marked as unallocated, to store a new one. But if the new file is bigger than the old one, and the file system tries to store it starting from the same sector entry as the deleted one, it wont have enough space to store it. So the file system will allocate-overwrite all the sectors of the previous deleted file, while the remaining data which do not fit, will be stored in other unallocated sectors. This results to file fragmentation.

Although fragmentation in current file systems is small [17], the classification of the missing fragmented parts of a file are essential for its recovery. In that case, validation techniques which use the complete file content are unable to provide aid to forensic examiners.

1.3 File Fragment Classification

File fragment classification is a technique that uses only a small fragment of a file, in order to determine its type. This approach is independent from files overall structure. Although in theory, file fragment classification looks like an ideal approach, in practice it proved to be difficult to create a technique of high precision [8]. It is noteworthy that in the last Digital Forensic Research Workshop(DFRWS 2012) challenge, the winning classification tool achieved an overall classification accuracy of 36% [7], in a corpus of 38 different file types.

1.4 Objectives

The main objectives in this project are:

1. Create a classification algorithm ,for identifying document-type fragments, of higher precision that the existing similar algorithms. In particular, we focus on the classification of text, xls, doc and pdf file fragments and try to improve their classification precision.

2. Test the hypothesis that by analysing only a special ASCII byte-set of file fragments which corresponds to the printable ASCII characters, accuracy of classification algorithms can be enhanced for document-type fragments. This ASCII subset is comprised of byte values

of the range ($32 \geq b \leq 126$) among with the tab(9), new line(10) and carriage return(13) bytes. From this point, we will refer to this special ASCII subset as "plain text".

1.5 Algorithms Requirements

The design requirements for our classification algorithm are as follows:

1. Speed - Comparable in runtime performance to the current light-weight algorithms such as the N-Gram Analysis [5] [8] and the BFA algorithm [2] [8].
2. Accuracy - Improve upon the overall accuracy of the algorithms in the same runtime performance class.
3. Operate in common fragment sizes, preferably of 512-bytes size, the smallest relevant size which is also equivalent with a hard drives sector size.

1.6 Methodology

Most of the current file and fragment classification techniques use the whole byte content of a file/fragment for both the training and classification procedures. Since we intend to create an algorithm which would be able to yield better accuracy results for fragments that originate from a document file type, we want to test the hypothesis that by using only the plain text ASCII subset of a fragment, we could achieve better results regarding text fragment classification. The plain text characters are a behavioural trait of a document so we expect that their occurrence will be more frequent in a document file.

To test our hypothesis we have to use one of the current classification algorithms in order to compare their accuracy results. Additionally, since our main goal is to design a classification algorithm which will satisfy the already mentioned requirements, we should carefully choose a currently available algorithm that has the potential to be easily modified, without adding significant complexity, and to create a custom more effective version of it. Our algorithm of choice is the Byte Frequency Analysis [2]. More about the reasons of this choice can be found in chapter 3.

Our design procedure is comprised of two main phases. In phase 1 we intent to use a BFA that uses only the plain text byte set for a fast scan of the corpus, in order to isolate a big amount of document-type fragments. In phase 2 we analyse the complete ASCII byte set of BFA output and try to classify of what types these document-type fragments are.

During phase 1, we use 4 variations of BFA that analyse only the plain text of the input fragments and also test our hypothesis. We compare the results of these variations with each other and we choose the one that yields the best results regarding text fragment classification. By text fragment classification we mean the classification of document-type fragments like pdf, text, doc and xls as text. After that we compare the best BFA variation with the default BFA, which takes under account the whole ASCII byte set, that Shahi [8] used in a corpus comprised of the same file types as the one we use. Finally, we choose the variations that yields the best results. This BFA variation will be the first part of our final algorithm. Thereafter, we isolate all fragments that were classified as text, resulting in a new corpus and proceed in phase 2 of our design procedure.

During phase 2, we analyse the whole byte content of the fragments that were classified as text, trying to find patterns that could aid our algorithms design. Initially we used simple statistical metrics such as the mean, mode, median and standard deviation trying to find characteristic patterns in specific file types. This resulted to focus on some specific byte sequences, where in conjunction with the histogram analysis we did, resulted in the discovery of two new metrics. The Individual Null Byte Frequency and the Plain Text Concentration Categories. We combine these two metrics along with the Shannon entropy metric [9] and the accuracy levels that BFA produced in phase 1, to create a new custom algorithm.

Chapter 2

Related Work

Karresand and Shahmehri [10] introduced a new algorithm that uses a metric called Rate-of-Change (ROC). They define the rate of change of a byte content as the difference of the ASCII values of consecutive bytes. Although this technique yields good classification rates for jpeg files (99% true positives), mainly because of their 0xFF00 metadata tags, for other files types the false positive rates are extremely high (e.g for zip and portable executable(PE) files near 70% false positives rates).

Veenman[11] used a combination of the BFA[2] with Shannon entropy and Kolmogorov complexity measures to classify fragments that were 4096 bytes in size. He used a corpus of 450mb comprised of 11 different file types. He managed to achieve high detection rates(99%) for jpeg and html files. However, results for the other file types weren't so good, achieving an overall acurracy of 45%. Additionally, the corpus that he used is not big enough to produce statistically significant results. Moreover, the big size of the fragments that Veenman used is not convenient enough for a real forensic case.

Calhoun and Coles [12] used a set of techniques like byte frequency of ASCII codes and Shannon entropy, linear discriminant analysis and prediction with longest common substrings and subsequences along with many other common statistical metrics. Their corpus was comprised of gif, pdf, jpeg and bmp files. Although they achieved a high average rate of correct prediction of 88.3%, their testing set was comprised only of 50 fragments per file type. The fragments size that were used in their experiment was of 512 and 896 bytes. Moreover, since they don't give information about the lengths of the file type representative strings that were used, we don't know how expensive longest common subsequence technique can be.

Axelsson[13] used a corpus of 28 different file types and applied the k-nearest-neighbour classification technique with Nearest Compression Distance(NCD) as the distance metric between file fragments. The results are unremarkable, achieving an average accuracy of around 34%. It was observed that this approach achieved higher accuracy for fragments with high entropy.

Li et al.[5] used the N-Gram Analysis to create representative fileprints for file types. The fileprints was based on a centroid which combined the mean and the standard deviation of byte frequencies. More specifically, they focused on 1-Gram Analysis of the ASCII byte values, representing a file as a 256-element histogram. In order to compare an unknown byte stream with a fileprint they used the Mahalanobis distance function. When they applied this technique in full files they achieved success rates of 60-90%. Morever by using only the first 20 bytes o files they managed to achieve an accuracy of 99%, but this was due to the fact that these 20 bytes mainly contained header data("magic numbers").

Fitzgerald, Mathews, Morris and Zhulyn[15] investigated whether techniques from natural language processing could be applied successfully to file fragment classification. They used the macro-averaged F1 metric in a set of 24 file types. They managed to achieve an average prediction accuracy of 49.1% on 24 file types out performing Axelssons (34% for 28 file types) and Veenmans (45% for 11 file types) results.

Lastly, Shahi[8] tested 4 different classification algorithms in the same corpus, in order to compare their performance. His corpus was comprised of 10 different file types. The algorithms used were the BFA[2], the N-Gram Analysis[5], the Rate of Change[10] and the algorithm of Conti et al. [16]. The results show that the average overall accuracy of the aforementioned techniques is around 30%. Moreover he benchmarked their performance in terms of execution time and found out that the N-Gram Analysis is the fastest among them, with BFA coming second, third the Rate of Change and fourth the algorithm of Conti et al.

Chapter 3

Experimental Setup

3.1 Byte Frequency Analysis(BFA) Algorithm

BFA[2] is a statistical learning algorithm that was initially developed to analyse and classify whole files. It was not meant to be used for file fragment classification. By counting the number of instances of each byte in a file of a certain type, BFA creates a representative average value for each byte instance, among with its respective correlation strength. This results in a fingerprint of a particular file-type. Thereafter, during the classification procedure, the input file is compared with every fingerprint and an accuracy level is created for each of them. BFA classifies the file to the file type of the fingerpint that corresponds to the highest accuracy level.

Shahi[8] trained and tested the BFA with file fragments of 512-byte size. His results show that although the algorithm is pretty bad for broad fragment classification, it is quite good in classifying fragments that belong to document-type files, as text. He tested the performance of BFA, along with the Byte Frequency Correlation algorithm, n-Gram Analysis and Conti et al. algorithm. The results show that BFA has the highest precision in classifying document-type fragments as text.

In contrast to the default technique, we use a BFA that trains our fingerprints with byte values that correspond only to the plain text ASCII characters, instead of the complete byte-set of the fragments. We also use fragments of 512-bytes size. This BFA will be the first half of our final algorithm and after this point we intend to use additional metrics to create a custom classifier. Taking under account speed requirements, BFA seems as a good candidate since it is a lightweight technique, compared to similar statistical learning algorithms[8] or heavier machine learning techniques.

3.2 Data Set

The data set we used for our training, experimentation, analysis and testing procedures is derived from Garfinkels[17][18] coprus, Wikipedia and Academic Earth [19] and is a subset of the coprus that Shahi used in [8]. Our corpus is comprised of 10 different file types with a total size of about 20GB. We divided the corpus in half, resulting in two subsets of 10GB each. The experimental and the final testing set.

We use the experimental set to do all of our experimentations, analysis and training, and the final testing set for testing the performance of our final algorithm. At this point we should note that the 10GB that corresponds to the final testing set wont undergone any type of analysis that will affect the design of our algorithm, since we only want to use it for testing our final algorithm. We fully designed our algorithm based only on the experimental set.

In the experimental set, we split these 10GB in two subsets of 9-1 ratio. 90% of the experimental set is used as our training set and the other 10% as our experimental testing set. Additionally, we transformed all of our files content, in both the experimentation and the final testing set, into 512-byte blocks, which we refer to them as fragments. Since our algorithm would be able to classify only fragments that contain at least one plain text character, fragments with no plain text were discarded. The percentage of discarded fragments per file type can be found in Table 3.1. As we can see, the percentage of fragments with no plain text for most of the file types is around 5%. Surprisingly enough, this percentage is significantly higher(13-17%) for the doc file type. This is quite interesting since files of the doc type are documents, which are mainly comprised of plain text characters.

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
Training Set										
num.of files	1,642	1	954	1,697	1	373	193	1,781	464	4,395
size in megabytes	869.3	860.6	831.2	867.6	813.6	869.5	866.9	870.5	863.4	868.9
expected num. of fragments	1,780,326	1,762,508	1,702,297	1,776,844	1,666,252	1,780,736	1,775,411	1,782,784	1,768,243	1,779,507
output num.of fragments	1,694,034	1,680,771	1,622,534	1,467,314	1,588,908	1,684,374	1,683,444	1,698,877	1,685,954	1,692,813
percentage of fragments with no plain text	4.8	4.6	4.7	17.4	4.6	5.4	5.2	4.7	4.7	4.9
Testing Set										
num.of files	217	1	367	257	1	81	35	214	101	555
size in megabytes	100	104.9	97.4	100.2	104.9	100.2	100.6	100.2	100.2	101.5
expected num. of fragments	204,800	214,835	199,475	205,209	214,835	205,209	206,028	205,209	205,209	207,872
output num.of fragments	189,732	204,795	190,055	177,887	204,728	193,352	195,289	195,608	195,656	195,653
percentage of fragments with no plain text	7.4	4.7	4.7	13.3	4.7	5.8	5.2	4.7	4.7	5.9

Table 3.1: Experimental Data Set

Furthermore, we used our training set to train our fingerprints and the experimental testing set to test all 4 variations of our BFA algorithm. Both of the aforementioned sets undergone statistical analysis in order to discover useful patterns. More detailed information about our experimental data set can be found in Table 3.1.

Information regarding the final testing set that we used to test the performance of our final algorithm can be found in Table [final testing set].

Chapter 4

Algorithm Development

4.1 Approach Description

Our algorithms development procedure is comprised of two main phases. In the first phase, we use 4 different variations of the BFA using only the plain text ASCII byte set of the fragments. We compare the 4 BFA variations and we choose the one that yields the best results regarding text fragment classification. Additionally, we compare the performance of the best BFA variation with the results of the default BFA, which takes under account the complete ASCII byte set, that Shahi[8] used in a similar corpus. In the end of phase 1 we choose the best technique regarding text fragment classification and proceed to phase 2.

In phase 2, we isolate all the fragments that were classified as text from the ooptimal BFA variation, in order to analyse them. This analysis resulted in the discovery of 2 new lightweight classification metrics, where in conjunction with the Shannon entropy[9] metric and the file type accuracy levels of the BFA aid the design of our classification algorithm.

4.2 BFA Variations

4.2.1 Variation 1 - Plain Text ASCII Subset

In this variation we created 10 fingerprints which were trained with fragments from the training set, one for each file type. We used only the printable ASCII characters (32 ≥ b ≤ 126) along with the tab(9), new line (10) and the carriage return(13) characters. The results can be found in Table 4.1.

This variation of BFA classifies 589,758 fragments as text which corresponds to the 30.4% of the initial corpus. 501,012 of them are fragments that originate from pdf, xls, doc and text files and 88,746 fragments originate from the other 6 file types. This means that in the set that is classified as text we have an 85% of true positives in identifying document-type fragments as text with 15% false positives. This 85% of true positives corresponds to the 66.7% of the total pdf, xls, doc and text files of our corpus.

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
num.of fragments	189,732	204,795	190,055	177,887	204,728	193,352	195,608	195,608	195,656	195,653
pdf	27.9	52.3	0	20.3	48.1	0.2	35.3	40.7	46.5	44.1
zip	20.2	26.6	0	13.3	28.0	0.1	24.9	29.2	24.7	28.2
text	21.3	4.9	98.0	50.4	4.4	95.5	14.1	6.0	7.1	7.2
doc	14.4	4.2	0.5	7.1	5.2	0.2	9.7	7.9	8.7	5.8
mp4	1.7	0.6	0	0.2	0.8	0	0.4	0.5	0.4	0.5
xls	1.2	0	1.4	0.8	0.1	3.9	1.0	0.2	0	0.1
ppt	3.2	2.2	0	1.8	2.7	0	3.3	3.3	2.7	2.9
jpg	0.5	0.1	0	0.1	0.0	0	0.1	0.1	0	0.1
ogg	2.8	2.2	0	1.4	3.0	0	2.8	3.0	2.7	2.7
png	6.8	6.9	0	4.6	7.7	0	8.3	9.1	7.2	8.5
Unclassified	0	0	0	0	0	0	0	0	0	0

Table 4.1: BFA Results - Fingerprints trained with plain text ASCII subset

4.2.2 Variation 2 - Plain Text Concentration Categories

During our research we thought that it would be interesting to analyse the distribution of byte value sthat correspond to the plain text ASCII subset. Depending on the concentration of plain text of a fragment, the fragment was assigned to one of 4 plain text concentration categories. 0-25%, 25-50%, 50-75% and 75-100%. The results of this analysis can be found in Table 4.2.

As it seems fragments from certain file types are more likely to belong to certain concentration categories. For example almost all text fragments(99.95%) contain more than 75% of our special ASCII subset and almost all xls fragments less than 50%. Undoubtedly this is completely reasonable. Text files are mostly comprised of plain text and Excel sheets, due to their cell-like structure, contain less printable characters. And this analogy is more obvious in a 512-byte fragment. That finding can be used as a metric to improve current classification techniques and we are going to elaborate more on this later in this chapter.

Based on the analysis results, we thought that it would be interesting to divide the fragments of our training set in 4 such categories. Then for each category and for each file type we created their respective fingerprints. So we ended up with 40 fingerprints, 4 for every file type. The algorithm first checks the plain text concentration of the input fragment and according to its value, it compares the fragment with the fingerprint of the respective category. The results of this BFA variation can be found in Tables 9.1, 9.2, 9.3 and 9.4.

ratio	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
0 - 25%	9,327	347	235	528,661	3,130	1,054,503	114,968	7,842	785	11,875
25 - 50%	1,332,849	1,680,052	436	768,686	1,585,760	576,755	1,547,585	1,685,320	1,684,877	1,674,301
50 - 75%	86,583	370	181	8,834	18	31,595	10,106	1,305	287	1,787
75 - 100%	265,275	2	1,621,682	161,133	0	21,521	10,785	4,410	5	4,850
Total:	1,694,034	1,680,771	1,622,534	1,467,314	1,588,908	1,684,374	1,683,444	1,698,877	1,685,954	1,692,813
0 - 25%	0.55	0.02	0.01	36.03	0.20	62.61	6.83	0.46	0.05	0.70
25 - 50%	78.68	99.96	0.03	52.39	99.80	34.24	91.93	99.20	99.94	98.91
50 - 75%	5.11	0.02	0.01	0.60	0	1.88	0.60	0.08	0.02	0.11
75 - 100%	15.66	0	99.95	10.98	0	1.28	0.64	0.26	0	0.29

Table 4.2: Training Set Ratio of special ASCII subset Analysis

The accuracy for both the actual classification and the text classification are really bad. This variation classified 366,969 fragments as text which corresponds to the 18.9% of the initial corpus. 87,837 of them are fragments that come from pdf, xls, doc and text files and 279,132 fragments originate from the other 6 file types. This means that in the set that is classified as text we have an 31.5% of true positives in identifying document-type fragments as text with 68.5% false positives. This percentage of true positives corresponds to the 11.7% of the total pdf, xls, doc and text files of our corpus.

The bad results are probably due to the fact that some of the fingerprints were trained with a tiny amount of fragments, so there are not representative at all, for the category they were build for. For example it is obvious that in the 0-25% category the xls fingerprint was trained with the 62.83% of the total xls fragments and the ogg fingerprint, for this particular category, was trained only with the 0.02% of the total ogg fragments. Probably this is the reason why in the 0-25% category most of the fragments were classified as xls since most of the other fingerprints, with the only exception of xls, were under-trained. This observation led as to the formulation of the next variation.

4.2.3 Variation 3 - Dominant Plain Text Concentration Categories

If we look at Table 4.2 it is obvious that most fragments of a certain file type are expected to belong to one of the 4 categories that we discussed in the previous variation. We hypothesized that for every file type the category which contains the majority of files fragments is more representative for the respective file type than the others. So from the 4 fingerprints that we created for every file type for the previous BFA variation, we chose the one that was trained with the highest plain text concentration category of this particular file type. We call this category the dominant concentration category of the file type. For example the dominant plain text category of the text file type is the 75-100%, for the pdf is the 25-50%, for the xls is the 0-25% etc.

Consequently, we ended up with 10 fingerprints witch corresponds to the dominant categories of every file type. This variation is identical with the first one, with the only difference that we use the fragments of the dominant categories of every file type to train our fingerprints instead

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
num.of fragments	189,732	204,795	190,055	177,887	204,728	193,352	195,289	195,608	195,656	195,653
pdf	5.0	3.9	0	2.9	4.9	0	5.3	5.4	4.8	5.1
zip	20.4	26.8	0	13.4	28.2	0.1	25.1	29.5	24.9	28.4
text	27.9	6.8	98.4	51.9	6.4	81.7	17.3	8.6	10.6	9.0
doc	31.4	51.8	0.1	22.1	47.4	0.2	37.5	42.0	47.8	44.6
mp4	3.0	1.9	0	0.9	2.8	0	1.6	1.7	1.4	1.9
xls	1.8	0.3	1.5	2.6	0.4	17.8	1.8	0.4	0.4	0.3
ppt	6.7	6.5	0	4.7	7.2	0	8.5	9.2	7.5	8.1
jpg	1	0.3	0	0.3	0.3	0	0.5	0.6	0.3	0.4
ogg	2.2	1.5	0	1	2.1	0	1.9	2.1	1.9	1.9
png	0.7	0.3	0	0.2	0.4	0	0.5	0.5	0.4	0.4
Unclassified	0	0	0	0	0	0	0	0	0	0

Table 4.3: BFA Results - Dominant Fingerprints

of the whole fragment set. The results of this BFA variation can be found in Table 4.3.

This BFA variation classified 589,402 fragments as text which corresponds to the 30.3% of the initial corpus. 490,267 of them are fragments that come from pdf, xls, doc and text files and 99,135 fragments originate from the other 6 file types. This means that in the set that is classified as text we have an 83.2% of true positives in identifying document-type fragments as text with 16.8% false positives. This percentage of true positives corresponds to the 65.3% of the total pdf, xls, doc and text files of our corpus.

4.2.4 Variation 4 - Fragments above 75% Plain Text Concentration classified as text

According to the results of Table 4.2 almost all text fragments(99.5%) contain more than 75% of plain text. In the same concentration category, fragments of pdf, doc and xls correspond to 15.66%, 10.98% and 1.28%, of the total amount of fragments of their particular file type, respectively. All other file types have less than 1% of their total fragments in this plain text concentration category. We thought that it would be interesting to apply the BFA of variation 1 only to the fragments which contain less than 75% plain text and every fragment above this percentage would be classified as text. We should note that we decided to use the fingerprints of variation 1

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
num.of fragments	165,840	204,795	1,491	157,196	204,728	192,044	192,236	194,582	195,656	195,651
pdf	31.5	52.3	3.5	22.9	48.1	0.2	35.9	40.9	46.5	44.1
zip	21.6	26.6	2.7	15.0	28.0	0.1	25.2	29.4	24.7	28.2
text	15.2	4.9	26.4	44.1	4.4	95.5	13.1	5.5	7.1	7.2
doc	16.0	4.2	59.6	7.9	5.2	0.2	9.7	7.9	8.7	5.8
mp4	0.6	0.6	0.1	0.3	0.8	0	0.4	0.5	0.4	0.5
xls	1.2	0	5.0	0.8	0.1	3.9	0.8	0.2	0	0.1
ppt	3.5	2.2	1.1	2.1	2.7	0	3.4	3.3	2.7	2.9
jpg	0.1	0.1	0.1	0.1	0	0	0.1	0.1	0	0.1
ogg	2.8	2.2	0.7	1.6	3.0	0	2.8	3.0	2.7	2.7
png	7.5	6.9	0.8	5.2	7.7	0	8.5	9.2	7.2	8.5
Unclassified	0	0	0	0	0	0	0	0	0	0

Table 4.4: BFA - Fingerprints Trained in 0-75% and tested in 0-75%

instead of the dominant fingerprints of variation 2, because overall percentage of text fragment classification is better for variation 1. The results of this variation of BFA can be found in Table 4.4. This BFA variation classified 590,834 fragments as text which corresponds to the 30.4% of the initial corpus. 512,855 of them are fragments that come from pdf, xls, doc and text files and 77,979 fragments originate from the other 6 file types. This means that in the set that is classified as text we have an 86.8% of true positives in identifying document-type fragments as text with 13.2% false positives. This percentage of true positives corresponds to the 68.3% of the total pdf, xls, doc and text files of our corpus.

4.2.5 Optimal Variation for Text Fragment Classification

It is obvious that the second variation is by far the worst and cannot aid the design process of our classification algorithm. Among the other three variation, variation 4 yields the best results. Both coverage and accuracy of variation 4 is undoubtedly the highest among the other two.

However, taking under account that these are results from a controlled corpus and not from a real life scenario, the fact that variation 4 classifies every fragment with more than 75% plain text concentration as text is a major weakness.

In a real life scenario, the ratio between the amount of fragments of every file type it's highly unlikely to be 1:1, as it is in our corpus. Therefore in a scenario where the corpus does not contain any text fragments, every fragment with a plain text concentration higher than 75% would be falsely classified as text. Furthermore, our corpus is comprised only of 10 file types. Considering the fact that the number of file types that a forensic practitioner is likely to encounter in real life cases is bigger, renders variation 4 unscalable. We should conduct similar research for all file types first, in order to be able to say if variation 4 can be used in actual forensic cases. Among the remaining variations, variation 1 is slightly better in both coverage and accuracy than variation 3. We judge that this is the optimal variation of BFA for text fragment classification among the 4 that we tested.

4.2.6 BFA Training - Complete ASCII Set VS. Plain Text

Although BFA variation 1 yielded the best results regarding text fragment classification among the other 3 variation, a comparison with a BFA that uses the complete ASCII byte set is essential, in order to choose which approach is the best for the design of our algorithm.

Shahi[8] tested a BFA for fragment classification using the exact same file types as we did. The only exception is that he used the whole ASCII byte set for his fingerprints training. The corpus that he used is almost 10 times bigger than the one we used for training. Conveniently enough, he trained his fingerprints with 10%, 20%, 50% and 100% of his training data set and provided the accuracy results. Our training set, around 800mb for each file type, is approximately the 10% of Ashims training set. In order to have a more objective comparison, we are going to compare the results that Shahi got by using fingerprints that were trained with the 10% of his training set, with our BFA variation 1. That way, fingerprints from both approaches have the same amount of training. The results can be found in Table 4.5.

For broad fragment classification, fingerprints that use the whole byte set seems to be way more effective than variation 1. Only the accuracies for pdf and ppt are higher in variation 1, simply because Shahis BFA achieved 0% of true positives for these file types.

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
pdf	0	0	0	0	0	0	0	0	0	0
zip	33.6	86.0	1.9	17.9	22.0	0.0	48.1	33.5	6.7	62.8
text	15.7	0.1	96.2	47.7	4.7	43	5.5	1.1	10.4	2.3
doc	2.1	0	0	0.5	0.6	0	0.4	0.1	8.2	0.3
mp4	10.1	4.5	0.4	4.1	27.2	0	12.3	25.2	18.2	11.4
xls	11.4	0.3	0.3	17.9	0.2	56.8	10.9	4.4	6.4	1.8
ppt	0	0	0	0	0	0	0	0	0	0
jpg	2.6	1.3	0.2	2	0.2	0	4.6	9.7	3.4	1.9
ogg	20.6	3	0.2	6.5	39.7	0	10.9	16.3	40.2	6.4
png	4.1	4.5	0.4	2.8	5	0	6.8	9.4	6.2	12.8
Unclassified	0	0	0	0	0	0	0	0	0	0

Table 4.5: BFA Results - Complete ASCII Byte Set Training

Regarding text fragment classification the accuracy results are pretty close. We took the accuracy percentages that correspond to text fragment classification from Table 4.5 and calculated the amount of fragments that would have been classified as text from this technique. We should mention that since Shahis BFA is not limited to classify fragments that do not contain plain text, the amount of fragments that his BFA could process is bigger (Table 3.1).

According to this, that BFA would have classified 462,345 fragments as text which corresponds to the 22.3% of the initial corpus. 410,173 of them are fragments that come from pdf, xls, doc and text files and 52,172 fragments originate from the other 6 file types. This means that in the set that is classified as text we have an 88.7% of true positives in identifying document-type fragments as text, with 11.3% false positives. This percentage of true positives corresponds to the 50.3% of the total pdf, xls, doc and text files of our testing set(fragments with no plain text included).

Although the accuracy of Shahis BFA is slightly higher(88.7%) from variation 1(85%), the amount of document-type fragments that is classified as text is significantly lower. Variation 1 classified as text 501,012 of the total pdf, xls, doc and text fragments, in comparison to Shahis BFA that would have classified 410,173. By using BFA as the first phase of our algorithm, we aim to retrieve us much pdf, xls, doc and text fragments as possible and minimize false positives. In that case, this is a trade-off between accuracy and the amount of document-type fragment retrieval. Accuracy levels are pretty close. However, variation 1 classifies significantly more(22%) pdf, xls, doc and text fragments of the total corpus as text. For that reason, we chose to use variation 1 over a BFA that uses the complete ASCII byte set for its fingerprints training. Therefore, our final algorithm will make use of BFAs variation 1.

4.3 BFAs Text Output Analysis

4.3.1 BFA Variation 1 Output

After the run of variation 1 BFA, we isolated all fragments which were classified as text. Initially, we expected that BFA falsely classifies fragments from non-text files as text, due to their high plain text concentration. We conducted a plain text concentration analysis on the BFAs output and it seems that BFA classified as text fragments with diverse plain text concentration. This analysis can be found in Table 4.6.

Although the 85% of BFAs output originates from document-type files, our algorithms considers all these fragments to be of xls, pdf, doc and text type. By doing this, we expect that the amount of fragments that were falsely classified as text without belonging to a document-type file, will be evenly distributed among the false positive classification results for xls, pdf, doc and text fragments. Our algorithms goal is to be able to correctly identify and distinguish between xls, pdf, doc and text fragments. For that purpose we conducted statistical analysis in BFAs output trying to find patterns that will help us distinguish of what file type are the resulted document-type fragments. We introduce two new metrics, the Individual Null Byte Frequency and the Plain Text Concentration Categories. The Individual Null Byte Frequency in conjunction with Shannon entropy[9] can be used to effectively distinguish between pdf from xls and doc fragments. Additionally, the Plain Text Concentration Categories metric can be used to eliminate

ratio	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
0 - 25%	5,606	75	3	51,462	2,606	145,106	9,920	1,178	198	7,781
25 - 50%	14,008	9,901	127	16,474	6,395	35,315	13,867	9,328	13,715	6,352
50 - 75%	5,646	11	263	1,416	0	2,974	1,396	110	53	13
75 - 100%	15,115	0	185,952	20,247	0	1,298	2,373	1,025	0	0
Total:	40,375	9,987	186,345	89,599	9,001	184,693	27,556	11,641	13,966	14,146
0 - 25%	13.9	0.8	0	57.4	29	78.6	36	10.1	1.4	55
25 - 50%	34.7	99.1	0.1	18.4	71	19.1	50.3	80.1	98.2	44.9
50 - 75%	14	0.1	0.1	1.6	0	1.6	5.1	0.9	0.4	0.1
75 - 100%	37.4	0	99.8	22.6	0	0.7	8.6	8.8	0	0

Table 4.6: BFAs Output Plain Text Concentration Analysis

the chances of a fragment, that belongs to a certain plain text concentration category, to be falsely classified.

4.3.2 Plain Text Concentration Categories

As we already mentioned, file fragments of certain types are expected to have a characteristic plain text concentration. We use 4 concentration categories of equal size. 0-25%, 25-50%, 50-75% and 75-100%. Our metric assumes that fragments are of 512-bytes size. As we have already seen in Table 4.2, 75% or more of text fragments is plain text and the majority of xls fragments(97%) are 0-50% plain text. Moreover more than 90% of the total mp4, zip, ppt, jpg, png and ogg fragments are 25-50% plain text. Additionally, we run an extra analysis specifically for the text fragments and we found that 98% of them are fully comprised of plain text.

We are positive that this light weight metric can be combined with current techniques and increase their accuracy. For example if a fragment is classified as text and it contains at least one non-plain text byte, then probably it's not a text fragment. So a classification algorithm could make this simple check and substitute its first classification "guess" with the one that had the second highest accuracy level. Similarly, if a fragment is more than 75% plain text then probably it's not a mp4, zip or ogg fragment etc.

4.3.3 Individual Null Byte Frequency

We applied several statistical metrics such as median, mean, mode, standard deviation, minimum and maximum frequency byte values in the BFAs output fragments. However, we couldn't find something that could significantly aid our algorithms design. Thereafter, we manually inspected several fragments from all the file types, and we noticed that the amount of null bytes in xls fragments was significantly high. However, although slightly less, the frequencies of null bytes was also similar for doc and pdf fragments. We noticed that there were many long sequences of null bytes in most of the pdf and doc fragments but in the xls fragments these sequences were fewer. Additionally, the majority of the total null bytes in xls fragments were individual. Therefore, we analysed the distribution of individual null bytes for all the document-type fragments. As you can see in figure 4.1 the number of individual null bytes in xls fragments is obviously higher compared to the other file types. For text fragments, the amount of individual null bytes is 0 and for pdf and doc fragments the frequency mainly ranges from 0 to 25. Since the majority of text fragments are fully comprised of plain text, it's natural that they do not contain null values.

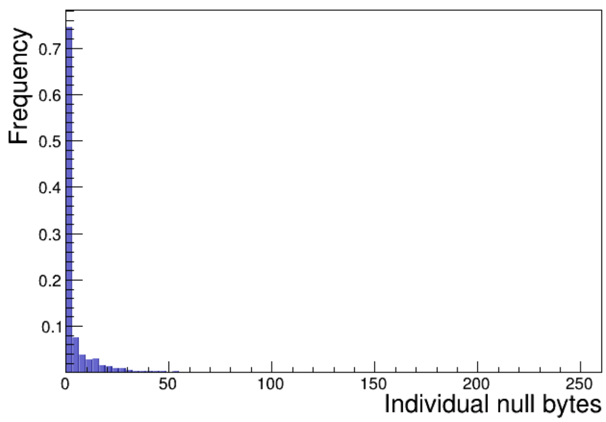

Figure 4.1.a: Pdf distribution

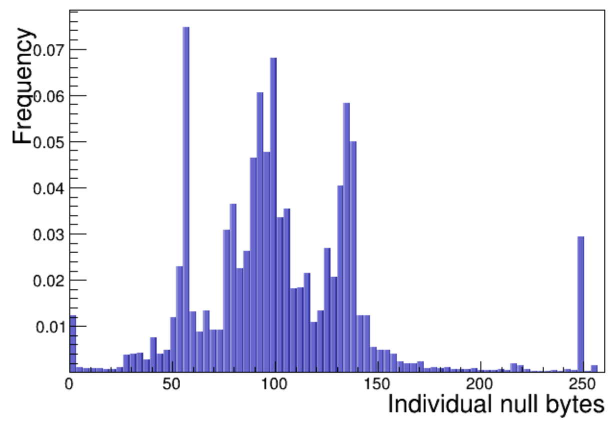

Figure 4.1.b: Xls distribution

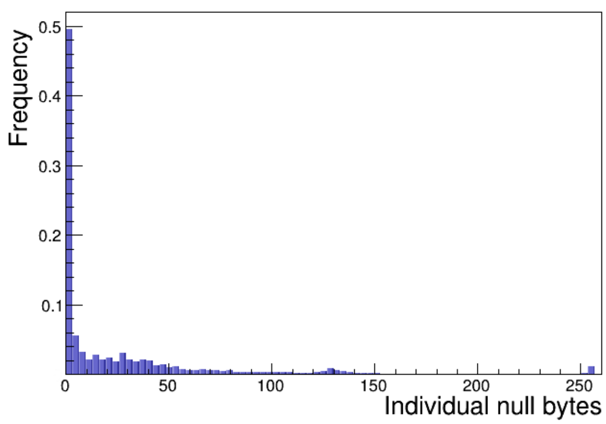

Figure 4.1.c: Doc distribution

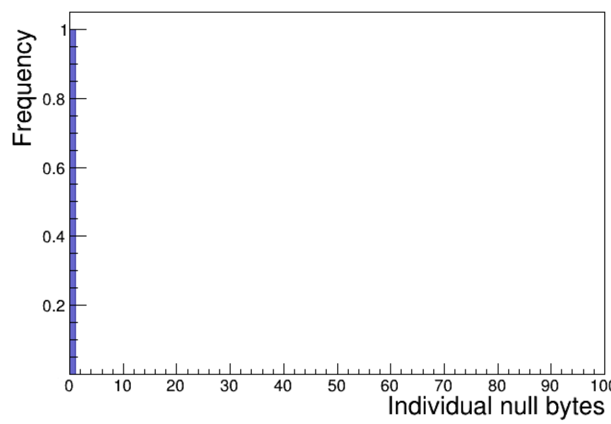

Figure 4.1.d: Text distribution

Figure 4.1: Individual Null Byte Distribution

4.3.4 Shannon Entropy

There is a widespread use of the Shannon entropy[9] metric in file carving techniques. Entropy measures how much information a sequence of symbols contains. Entropy is defined as:

$$H(X_i..X_n) = -\sum_{i=0}^{n} p(x_i)\log_2 p(x_i)$$

In our case, $X = X_i..X_n$ is the byte-content of a fragment, where $n = 511$ and $p(x_i)$ is the frequency of x_i in X. To calculate $p(x_i)$, we simply divide the number of occurrences of x_i in a fragment with the fragments size. It is known that usually compressed files have high entropy in contrast with text files that have low entropy[12][20]. In figure 4.2 we can see the entropy distribution among these file fragments. As expected, by beign a compressed file format pdf has

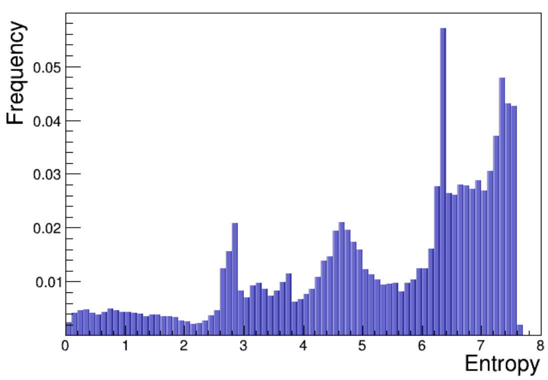

Figure 4.2.a: Pdf distribution

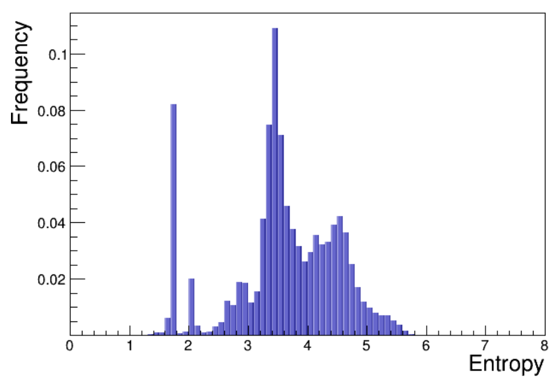

Figure 4.2.b: Xls distribution

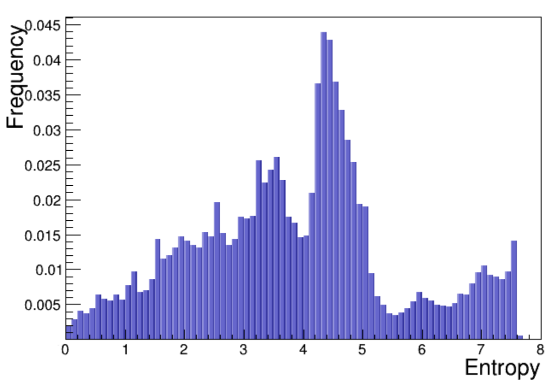

Figure 4.2.c: Doc distribution

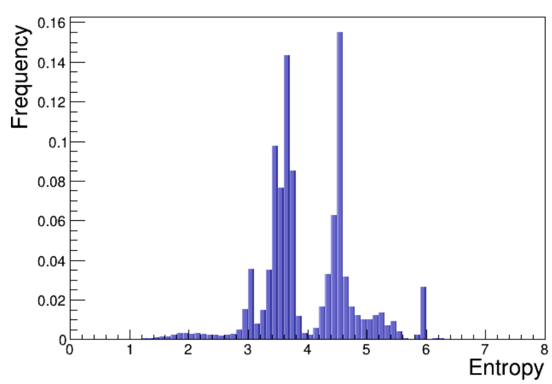

Figure 4.2.d: Text distribution

Figure 4.2: Entropy Distribution

significantly higher entropy than doc, xls and text fragments. Themajority of pdf fragments have an entropy value of 6 or more, in contrast with the other file types where the majority of their fragments has an entropy of lesser value.

4.4 Longest Common Subsequence

While trying to find a way to reduce false positives of the doc and xls fragment classification, we thought to test the performance and accuracy of the longest common subsequence technique. Calhoun[] used this technique to distinguish between fragments of two different file types. He achieved high accuracy results(90%) using the standard dynamic programming version of the algorithm. Even though the dynamic version is faster than the naive approach of the algorithm, with runtime complexity mxn, where mn the length of the input strings, it still seems like an "expensive" technique to be used in file carving. He extracted the longest common subsequences of every file fragment in his training set and concatenated them in a big string. This string is used as a representative of the respective file type. Due to the fact that the speed of this technique depends on the length of the input strings, it is essential to know about how long the file type representative string should be in order to be effective. Since he does not provide information about the length of the strings that he used as file type representatives , we want to find out strings of what length can be used as file type representatives and yield similar results. If the lengths are not too long then the computation of the longest common subsequence between two strings could be fast enough to be used in file carving techniques.

Instead of concatenating every longest common subsequence between fragments of the same file type, we tried a different approach. We used 500 fragments of the doc and 500 fragments of the xls type for our representative string creation. This resulted to $500x500 - 500 = 249,500$ comparisons for each file type in order to extract their longest common subsequences. We gathered all longest common subsequences from these comparisons and putted them in a map data structure. Then we sorted the map and took the first 100, 500, 1000 and 1500 most frequent longest common subsequences. We concatenated these subsequences in 4 long representative strings for each of the doc and xls file type. Thereafter, we used a set of 10,000 fragments, 5000 of xls and 5000 of doc type, to test their accuracy. At this point we should note that Calhoun used only 50 fragments per file type to test this technique. This was an additional reason to want to try its performance since we consider testing sets of this size extremely insufficient. However, we also consider our testing set significantly small, but since our goal was to test the correlation between the speed and the accuracy of that technique, that size is acceptable. The results can be found in Table 7.1.

n most frequent lcs	$n = 100$	$n = 500$	$n = 1000$	$n = 1500$
doc vs. xls precision	83	89.5	90.06	91.63
doc lcs representative string length	1,007	5,763	15,225	27,070
xls lcs representative string length	859	4,679	9,482	14,609

Table 4.7: Longest Common Subsequence comparison for doc vs. xls

As someone would expect, by using longer strings as file type representatives, the classification precision is enhanced. The precision gradually increases while using longer strings. However, Although the precision of this metric proved to be in pair with the results Calhoun presented in [12], its speed is way to slow to be used in real life cases. Even by using the shortest file type representative strings, which corresponds to the first 100 most frequent longest common subsequences of a file type, the runtime complexity remains extremely high. We compared the speed of this technique with our unoptimized algorithms speed, and although our benchmarking is not completely accurate, the longest common subsequent technique takes 56% longer to compute, than our complete algorithm(BFA included). Taking under account that our algorithm was designed to be able to handle 10 different file types and that the LCS technique that we tested only 2, it is obvious that the difference in speed is quite significant. Moreover, since our algorithm yielded few false positives for xls fragment classification, we don't think that we could improve our overall accuracy by using the LCS technique. In conclusion, we strongly believe that such an expensive technique is not appropriate for broad fragment classification and researchers should first invest time in searching for light weight techniques before trying brute force approaches.

Chapter 5

Algorithm Description

Chapter 6

Results

Chapter 7

Longest Common Subsequence

While trying to find a way to reduce false positives of the doc and xls fragment classification, we thought to test the performance and accuracy of the longest common subsequence technique. Calhoun[] used this technique to distinguish between fragments of two different file types. He achieved high accuracy results(90%) using the standard dynamic programming version of the algorithm. Even though the dynamic version is faster than the naive approach of the algorithm, with runtime complexity mxn, where mn the length of the input strings, it still seems like an "expensive" technique to be used in file carving. He extracted the longest common subsequences of every file fragment in his training set and concatenated them in a big string. This string is used as a representative of the respective file type. Due to the fact that the speed of this technique depends on the length of the input strings, it is essential to know about how long the file type representative string should be in order to be effective. Since he does not provide information about the length of the strings that he used as file type representatives , we want to find out strings of what length can be used as file type representatives and yield similar results. If the lengths are not too long then the computation of the longest common subsequence between two strings could be fast enough to be used in file carving techniques.

Instead of concatenating every longest common subsequence between fragments of the same file type, we tried a different approach. We used 500 fragments of the doc and xls type for our representative string creation. This resulted to $500x500-500 = 249,500$ comparisons for each file type. We gathered all longest common subsequences from these comparisons and putted them in a map data structure. Then we sorted the map and took the first 100, 500, 1000 and 1500 most frequent longest common subsequences. We concatenated these subsequences in 4 long representative strings for each of the doc and xls file type. Thereafter, we used a set of 10,000 fragments, 5000 of xls and 5000 of doc type, to test their accuracy. At this point we should note that Calhoun used only 50 fragments per file type to test this technique. This was an additional reason to want to try its performance since we consider testing sets of this size extremely insufficient. However, we also consider our testing set significantly small, but since our goal was to test the correlation between the speed and the accuracy of that technique, that size is acceptable. The results can be found in Table 7.1.

As someone would expect, using longer strings as file type representatives result in higher

n most frequent lcs	$n = 100$	$n = 500$	$n = 1000$	$n = 1500$
doc vs. xls precision	83	89.5	90.06	91.63
doc lcs representative string length	1,007	5,763	15,225	27,070
xls lcs representative string length	859	4,679	9,482	14,609

Table 7.1: Longest Common Subsequence comparison for doc vs. xls

classification precision. The precision gradually increases while using longer strings. However, Although the precision of this metric proved to be in pair with the results Calhoun presented [], its speed is way to slow to be used in real life cases. Even by using the shortest file type representative strings, which corresponds to the first 100 most frequent longest common subsequences of a file type, the runtime complexity remains extremely high. We compared the speed of this technique with our unoptimized algorithms speed, and although our benchmarking is not completely accurate, the longest common subsequent technique takes 56% more time to compute than our complete algorithm(BFA included). Taking under account that our algorithm was designed to be able to handle 10 different file types and that the LCS technique that we tried only 2, it is obvious that the difference in speed is quite significant. Moreover, since our algorithm yielded few false positives for xls fragment classification, we don't think that we could improve our overall accuracy by using the LCS technique. In conclusion, we strongly believe that such an expensive technique is not appropriate for broad fragment classification and researchers should first invest time in searching for light weight techniques before trying brute force approaches.

Chapter 8

Conclusion

Chapter 9

Appendix

9.1 test

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
num.of fragments	5,714	90	3	52,264	2,854	147,873	11,027	1,332	222	7,874
pdf	0	0	0	0	0	0.3	0	0.1	0	0
zip	0	0	0	0	0	0	0	0	0.5	0
text	0	0	0	0.1	0	0.7	0	0	0	0
doc	0	0	0	0	0	0	0	0.1	0	0
mp4	0	0	0	0	0.1	0.1	0	0	0	0
xls	99.6	95.6	100	99.6	99.9	97.3	98.3	95.3	96.3	99.9
ppt	0	0	0	0	0	0	0	0	0	0
jpg	0.3	4.4	0	0.2	0	0.9	1.6	4.5	2.7	0.1
ogg	0	0	0	0	0	0.2	0	0.1	0	0
png	0	0	0	0	0	0.4	0		0.5	0
Unclassified	0	0	0	0	0	0	0	0	0	0

Table 9.1: BFA - Fingerprints Trained in 0-25% and tested in 0-25%

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
num.of fragments	147,705	204,662	285	102,831	201,859	41,013	178,816	193,103	195,368	187,688
pdf	6.9	4	4.9	5.5	5.1	0.1	6	5.8	5.2	5.3
zip	25.2	26.7	14	23.7	28.4	0.6	27.6	30	25.3	29.5
text	32.6	40.1	14.7	30.9	38.8	0.8	33.4	34.7	36.5	37.3
doc	16.3	17.7	4.9	15.6	15.8	0.4	14.8	14.4	19.4	15.1
mp4	2	0.8	2.1	1.1	1.7	0	1.2	1.1	1.1	1.1
xls	3.9	1.7	49.1	11.5	0	97.9	4.2	0.8	1.3	0.4
ppt	9.2	6.7	7.7	8.8	7.4	0.2	9.5	9.8	8.1	8.6
jpg	0.7	0.3	0.7	0.5	0.2	0	0.6	0.6	0.4	0.4
ogg	2.6	1.5	1.8	2	2.2	0.1	2.2	2.2	2.2	2
png	0.6	0.3	0	0.5	0.4	0	0.6	0.5	0.5	0.5
Unclassified	0	0	0	0	0	0	0	0	0	0

Table 9.2: BFA - Fingerprints Trained in 25-50% and tested in 25-50%

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
num.of fragments	12,421	43	1,203	2,101	15	3,158	2,393	147	66	89
pdf	39.1	23.3	6.2	1.8	0	1.6	1.6	2	3	1.1
zip	4.8	16.3	6.7	10.4	0	0.4	3.1	5.4	1.5	14.6
text	0.6	2.3	1.6	5.9	0	0	2.3	2	0	9
doc	6.2	7	40.9	7.5	0	2.4	18.2	4.1	3	1.1
mp4	12.2	27.9	1.2	37.6	100	27.2	42.6	40.8	36.4	12.4
xls	13.5	0	1.4	19.6	0	65.5	18.7	35.4	15.2	1.1
ppt	16.0	0	17.5	1.4	0	1.5	0.6	0.7	1.5	0
jpg	5.3	0	15.1	1.2	0	1.2	7	1.4	3	3.4
ogg	0.6	0	8.8	3.7	0	0.2	4.9	0	36.4	0
png	1.5	23.3	0.5	10.9	0	0	0.9	8.2	0	57.3
Unclassified	0	0	0	0	0	0	0	0	0	0

Table 9.3: BFA - Fingerprints Trained in 50-75% and tested in 50-75%

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
num.of fragments	23,892	0	188,564	20,691	0	1,308	3,053	1,026	0	2
pdf	7.6	0	0.3	0.3	0	0	0.5	0	0	0
zip	0.7	0	0.4	0.5	0	3.7	5.9	1.2	0	0
text	11.8	0	1.4	3.4	0	6.2	2.3	1.8	0	0
doc	2	0	8.2	43.2	0	17.7	5.3	1.4	0	0
mp4	49.3	0	86.5	48.6	0	68.3	78.0	74.4	0	0
xls	7.9	0	0.6	1.2	0	0.9	2.9	0.1	0	0
ppt	0.8	0	0.7	0.1	0	0	0.3	0	0	100
jpg	4.2	0	1.4	1.7	0	1.3	0.8	20.6	0	0
ogg	4.3	0	0.4	0.9	0	1.8	3.7	0.7	0	0
png	11.4	0	0	0	0	0	0.4	0	0	0
Unclassified	0	0	0	0	0	0	0	0	0	0

Table 9.4: BFA - Fingerprints Trained in 75-100% and tested in 75-100%

Chapter 10

Bibliography

[1] Aronson, L., & Bos, J. Van Den. (2011). Towards an Engineering Approach to File Carver Construction. 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops, pp . 368–373.

[2] M. McDaniel and M. Heydari , “ Content based file type detection algorithms ,” in Proc. 36th Annu. Hawaii Int. Conf. System Sciences (HICSS’03)—Track 9 , IEEE Computer Society , Washington , D.C. , 2003 , p. 332.1

[3] G. G. Richard , III and V. Roussev , “ Scalpel: A frugal, high performance file carver ,” in Proc. 2005 Digital Forensics Research Workshop (DFRWS) , New Orleans , LA , Aug. 2005 .

[4] Pal, A., & Memon, N. (2009). The evolution of file carving. Signal Processing Magazine, IEEE, (March),pp. 59–71.

[5] Li W, Wang K, Stolfo S, Herzog B. Fileprints: identifying file types by n-gram analysis. In: IEEE information assurance workshop, 2005.

[6] Garfinkel, S. L. (2007). Carving contiguous and fragmented files with fast object validation. Digital Investigation, 4, pp. 2–12

[7] Maddox, L., & Beebe, N. (2012). Systematic Classification Engine and Data Analysis Overview by Dr . Nicole Beebe

[8] Shahi, A. Classifying the classifiers for file fragment classification, August 2012.

[9] Shannon CE. The mathematical theory of communication. Bell System Tech J 1948;27:379–423, 623–56.

[10] M. Karresand and N. Shahmehri, "File Type Identification of Data Fragments by their Binary Structure," in Proceedings of the 7th Annual IEEE Information Assurance Workshop, 2006.

[11] C. Veenman, "Statistical Disk Cluster Classification for File Carving," in Proceedings of the First Internation Workshop on Computational Forensics, 2007.

[12] Calhoun, W. C., & Coles, D. (2008). Predicting the types of file fragments. Digital Investigation, 5, S14–S20.

[13] Axelsson, S. (2010). The Normalised Compression Distance as a file fragment classifier. Digital Investigation, 7, S24–S31. doi:10.1016/j.diin.2010.05.004

[14] Gopal S, Yang Y, Salomatin K, Carbonell J. Statistical learning for file-type identification. In: 2011 10th International Conference on Machine Learning and Applications and Workshops (ICMLA), Vol. 1; 2011. p. 68–73.

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

[16] Conti G, Bratus S, Sangster B, Ragsdale R, Supan M, Lichtenberg A, et al. Automated mapping of large binary objects using primitive fragment type classification. In: Proceedings of the 2010 Digital Forensics Research Conference (DFRWS); 2010.

[17] S. Garfinkel, P. Farrell, V. Roussev and G. Dinolt, "Bringing science to digital forensic with standardized forensic corpora," in DFRWS, 2009.

[18] "Digital Corpora," [Online]. Available: http://digitalcorpora.org/.

[19] "Academic Earth," [Online]. Available: http://www.academicearth.org/.

[20] J. van den Bos and T. van der Storm, "Bringing Domain- Specific Languages to Digital Forensics," in Proceedings of the 33rd ACM/IEEE International Conference on Software Engineering (ICSE'11), vol. 2. ACM, 2011.