An Algorithm for Content-Based

Automated File Type Recognition

Mason B. McDaniel

A thesis submitted to the Graduate Faculty of

JAMES MADISON UNIVERSITY

in

Partial Fulfillment of the Requirements

for the degree of

Master of Science

Department of Computer Science

December 2001

Approved and recommended for acceptance as a thesis in partial fulfillment of the requirements for the degree of Master of Science.

Special committee directing the work of Mason B. McDaniel.

Thesis Advisor Date

Member Date

Member Date

Department Head Date

Received by the Graduate School Office

Date

ACKNOWLEDGEMENTS

I would like to thank the staff of the James Madison University Computer Science Information Security program for their dedication and assistance through past couple of years. I would specifically like to thank my advisor, Dr. M. Hossein Heydari, and thesis committee members, Dr. John McDermott and Dr. Mohammed Eltoweissy for their valuable input and assistance.

TABLE OF CONTENTS

TABLE OF FIGURES

This page intentionally left blank.

ABSTRACT

Identifying the true type of a computer file can be a difficult problem. Previous methods of file type recognition include fixed file extensions, fixed "magic numbers" stored with the files, and proprietary descriptive file wrappers. All of these methods have significant limitations. This paper proposes an algorithm for automatically generating "fingerprints" of file types based on a set of known input files, then using the fingerprints to recognize the true type of unknown files based on their content, rather than metadata associated with them. Recognition is performed by three independently selectable options: byte frequency analysis, byte frequency cross-correlation analysis, and file header/trailer analysis. Tests were run to identify optimal values of constants used in recognition calculations, and to measure the accuracy of the algorithm with different combinations of options. The accuracy varied greatly depending upon which options were used, from 23% accurate using only the first option, to 96% accurate using all three options. The algorithm could be used by virus scanning packages, firewalls, or any other program that needs to identify the types of files for proper operation. Finally, recommendations are made regarding additional research that could improve the flexibility and accuracy of the proposed algorithm.

CHAPTER 1: INTRODUCTION

Background

Computers use a tremendous array of file formats today. All types of files are frequently transmitted through intranets and the Internet.

Currently, operating systems, firewalls, and intrusion detection systems have very few methods for determining the true type of a file. Perhaps the most common method is to identify the type of a file by the file's extension. This is an extremely unreliable method, as any user or application can change a file's name and extension at any time.

As a result, some users are able to conceal files from system administrators simply by renaming them to a filename with a different extension. While this doesn't conceal the existence of a file, it can conceal the nature of a file and can prevent it from being opened by the operating system.

In addition, many virus-scanning packages default to only scanning executable files. These packages may miss any viruses contained within executable files that had non-executable file extensions. This could introduce vulnerabilities into a network, even if it contained virus protection.

The other common method of identifying file types is through manual definition of file recognition rules. This is an extremely time-consuming process, whereby an individual examines a file type specification, if one is available, and identifies consistent features of a file type that can be used as a unique identifier of that type. In the absence of a specification, the individual must manually examine a number of files looking for common features that can be used to identify the file type. Not only is this time-consuming, but it can require an individual with a highly technical background that is capable of doing a hexadecimal analysis of files.

Manual rule definition is the method used by many Unix-based operating systems, as well as tools used in forensic analysis of computer disks during investigations. These investigations could be part of law enforcement investigations, or part of internal corporate investigations. Regardless of the investigating authority, automated file type recognition is a critical part of this sort of computer forensic analysis.

An efficient, automated algorithm to perform this kind of file type recognition would be of tremendous benefit to organizations needing to perform forensic analyses of computer hard drives. It could also be used by virus protection software, intrusion detection systems, firewalls, and security downgrading packages to identify the true nature of programs passing through the protected systems. Finally, this kind of algorithm could be of use to the operating systems themselves to allow for correct identification and handling of files regardless of file extension.

Previous Work

To date, there have been relatively few methods for identifying the type of a file. One of the most common methods is the use of file extensions. Microsoft's operating systems use this method almost exclusively. They come preset with associations between file extensions and file types. If different associations are desired, they must be manually reconfigured by the user.[1]

As mentioned above, this approach introduces many security vulnerabilities. A user can change the extension of a file at any time, rendering the operating system unable to identify it. They can also change the file extension associations to fool the operating system in to handling files in an inappropriate manner, such as trying to execute a text file.

Another approach is that taken by many Unix-based operating systems. These make use of a "magic number" which consists of the first 16 bits of each file. Another file, such as */etc/magic* then associates magic numbers with file types.[2]

This approach has a number of drawbacks as well. The magic numbers must be predefined before the files are generated, and are then built into the files themselves. This makes it very difficult to change them over time, since a change might interfere with the proper operation of many files that were generated using the old magic number. Furthermore, not all file types use magic numbers. The scheme was initially intended to assist with the proper handling of executable and binary formats. With only 16 bits allocated, a number of extensions had to be introduced over time, such as using the "#!" magic number to identify a command to execute on the rest of the file.[3]

Another approach is to define a proprietary file format that encapsulates other files and provides information regarding their type. One example of this approach is the Standard File Format (SAF) developed by the Advanced Missile Signature Center (AMSC).[4]

There are many down sides to this approach. The specification must be written defining how to encapsulate and identify each file format. An individual or external system must identify the type of the file before it can be correctly encapsulated in the standard format in the correct manner. The most significant problem, however, is that this type of file can only be used within the small proprietary system that recognizes the "standard" format. The files cannot be exported to external systems such as the Internet without removing the encapsulation, and thus negating its benefit.

---

[1] "To Associate a File Extension With a File Type", Windows 2000 Professional Documentation, available online from: http://www.microsoft.com/WINDOWS2000/en/professional/help/win_fcab_reg_filetype.htm
[2] /etc/magic Help File, available online from: http://qdn.qnx.com/support/docs/qnx4/utils/m/magic.html
[3] Why do some scripts start with #!, Chip Rosenthal, available online from: http://baserv/uci/kun.nl/unix-faq.html
[4] The Advanced Missile Signature Center Standard File Format, available online from: http://fileformat.virtualave.net/archive/saf.zip

Although not directly related to file identification, the field of cryptanalysis has a very similar problem. In order for a software package to automate the process of attempting to break encryption, it must be able to recognize a decrypted product when it appears. This is a very difficult problem for the general case. For encrypted ASCII English text, however, it becomes easier. The letters of the English language appear with a predictable frequency in large text documents.[5] This means that a program can be written to analyze frequency distributions and detect the characteristic distribution pattern of English text when a text file has been successfully decrypted.

[5] Stallings, William, Cryptography and Network Security, Prentice Hall, Upper Saddle River, New Jersey, 1999, p. 32.

CHAPTER 2: THE ALGORITHM

Introduction

This paper describes an attempt to extend the concept of frequency analysis and apply it to the general case of generating a characteristic "fingerprint" for computer file types, and subsequently using the fingerprint to identify file types based upon their characteristic signatures. This would enable systems to solve many of the problems associated with the methods of file type recognition discussed in Chapter 1. The process could be almost entirely automated, and would not be affected if a user changed a file name or extension. In addition, if file types change, or new types were created, it would be a simple matter to update the file fingerprint library without affecting the functionality of files written under the previous file types.

The design goals for the proposed file recognition algorithm are as follows:

- Accuracy – The algorithm should be as accurate as possible at identifying file types
- Automatic generation of file type fingerprints – To generate a file type fingerprint, the user should only be required to input a batch of files of a known type. The fingerprint should then be generated automatically with no further input required from the user. (Moreover, additional files should be able to be added at any time to further refine a file type fingerprint.)
- Small fingerprint files – The fingerprint file sizes should be minimized.
- Speed – Comparisons should be as fast as possible for a given fingerprint file size
- Flexibility – The algorithm should provide a customizable tradeoff between speed and accuracy.
- Independence from file size – The size of the fingerprint files should be largely independent from the size of the input files.

These design goals can be achieved by implementing the three options described in this section, each of which could be selected independently, or used together for increased accuracy.

Option 1: Byte Frequency Analysis

Much like text documents can be broken into series of letters, all computer files can be broken into a series of bytes, which correspond to eight-bit numbers capable of

representing numeric values from 0 to 255 inclusive. By counting the number of occurrences of each byte value in a file, a frequency distribution can be obtained. Many file types have consistent patterns to their frequency distributions, providing information useful for identifying the type of unknown files.

Figure 2-1 shows the frequency distributions for two different RichText (RTF) files. Although there are noticeable differences, they are quite similar. Contrast those to the frequency distributions for two different Graphics Interchange Format (GIF) files, shown in Figure 2-2. Many file types likewise have characteristic patterns that can be used to differentiate them from other file formats.

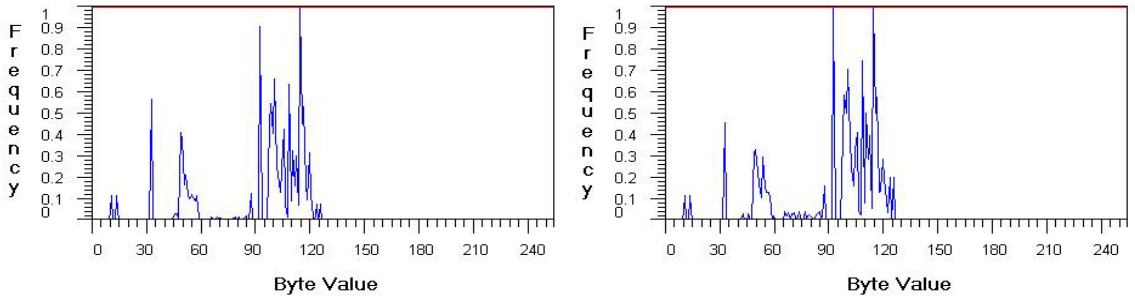

Figure 2-1 - Byte frequency distributions for two RTF files.

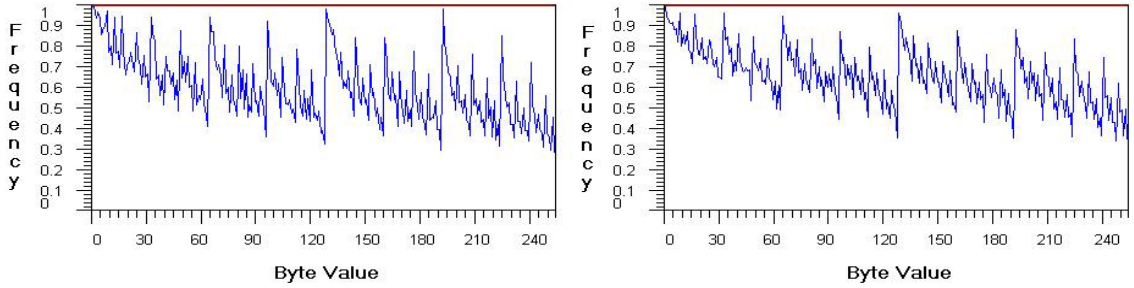

Figure 2-2 - Byte frequency distributions for two GIF files.

This section describes the methods used to build the byte frequency distribution of individual files, to combine the ratings from multiple files into a fingerprint representative of the file type, and to compare an unknown file to a file type fingerprint, obtaining a numeric score.

Building the Byte Frequency Distribution

The first step in building a byte frequency fingerprint is to count the number of occurrences of each byte value for a single input file. This is done by generating an array with elements from 0 to 255, and initializing all values to zero. Each byte in the input file is then looped through. For each byte, the value is extracted and the appropriate element

of the array is incremented by one. For example, if the next byte in the file contained the ASCII value 32, then array element 32 would be incremented by one.

Once the number of occurrences of each byte value is obtained, each element in the array is divided by the number of occurrences of the most frequent byte value. This normalizes the array to frequencies in the range of 0 to 1, inclusive. This normalization step prevents one very large file from skewing the file type fingerprint. Rather, each input file is provided equal weight regardless of size.

Some file types have one byte value that occurs much more frequently than any other. If this happens, the normalized frequency distribution may show a large spike at the common value, with almost nothing elsewhere. Figure 2-3 shows the frequency distribution for an executable file that demonstrates this. The file has large regions filled with the byte value zero. The resulting graph has a large spike at byte value zero, with insufficient detail to determine patterns in the remaining byte value ranges.

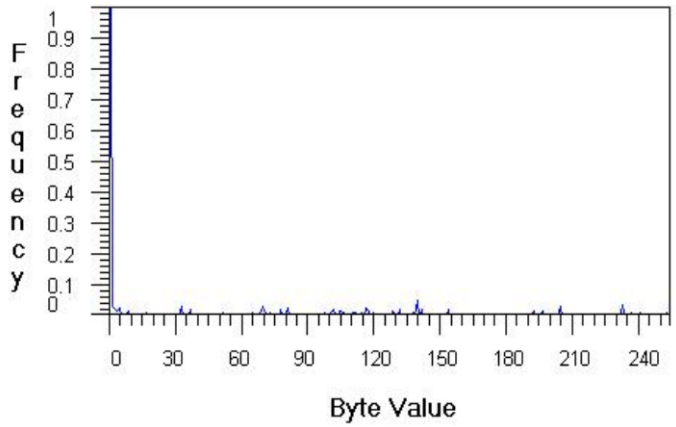

Figure 2-3 - Frequency distribution for a sample executable file.

A way to solve this problem would be to pass the frequency distribution through a companding function to emphasize the lower values. Common companding functions, such as the A-law and μ-law companding functions used in telecommunications,[6] can be roughly approximated by the following function, which can be very rapidly computed:

$$y = x^{\left(\frac{1}{\beta}\right)}$$

Figure 2-4 shows the graph of this function for $\beta = 2$.

[6] Bellamy, John, Digital Telephony, Second Edition, John Wiley & Sons, Inc., New York, New York, 1991, pp 110-119.

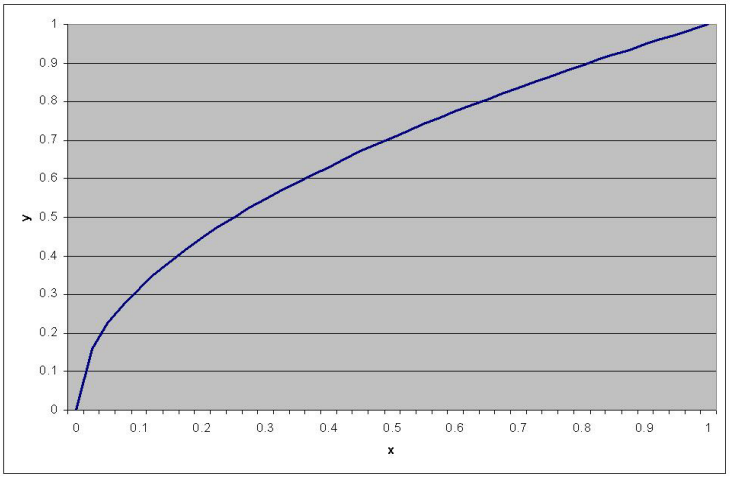

Figure 2-4 - Graph for companding function $y = x^{\left(\frac{1}{\beta}\right)}$ for $\beta = 2$.

The same file shown in Figure 2-3, after being passed through this equation, produces the frequency distribution shown in Figure 2-5. This graph shows more of the detail across all byte frequencies, and therefore may allow for more accurate comparisons. Different values of β produce different levels of companding. Tests were run, discussed in Chapters 3 and 4, to determine the optimal value of β for the most accurate file type recognition.

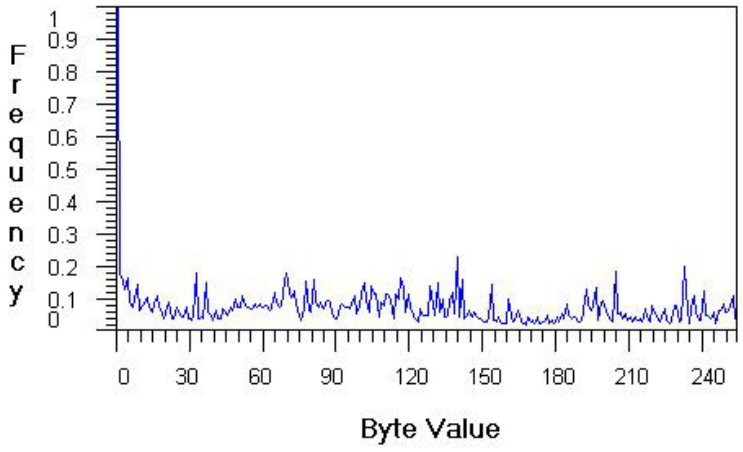

Figure 2-5 - Frequency distribution for a sample executable file after passing through the companding function.

The companding function results in a frequency distribution that is still normalized to 1. This is true since the most frequent byte value was normalized to 1, and the companding function with an input value of 1 results in an output value of 1.

Combining Frequency Distributions into a Fingerprint

A fingerprint is generated by averaging the results of multiple files of a common file type into a single fingerprint file that is representative of the file type as a whole. (See APPENDIX A for a description of the file format for the fingerprint file.)

To add a new file's frequency distribution to a fingerprint, the fingerprint is loaded, and the frequency distribution for the new file is generated. Next, each element of the new file's frequency distribution array is added to the corresponding element of the fingerprint score using the following equation:

$$New\,FP\,Score = \frac{\left(Old\,FP\,Score \times previous\ \ number\,of\ \ files\right) + New\,file\ \ score}{previous\ \ number\,of\ \ files + 1}$$

This results in a simple average, where the previous fingerprint score is weighted by the number of files already loaded into the fingerprint.

Aside from the byte frequency distributions, there is another related piece of information that can be used to refine the comparisons. The frequencies of some byte values are very consistent between files of some file types, while other byte values vary widely in frequency. For example, note that almost all of the data in the files shown in Figure 2-1 lie between byte values 32 and 126, corresponding to printable characters in the lower ASCII range. This is characteristic of the RichText format. On the other hand, the data within the byte value range corresponding to the ASCII English alphanumeric characters varies widely from file to file, depending upon the contents of the file.

This suggests that a "correlation strength" between the same byte values in different files can be measured, and used as part of the fingerprint for the byte frequency analysis. In other words, if a byte value always occurs with a regular frequency for a given file type, then this is an important feature of the file type, and is useful in identification.

A correlation factor can be calculated by comparing each file to the frequency scores in the fingerprint. The correlation factors can then be combined into an overall correlation strength score for each byte value of the frequency distribution.

The correlation factor of each byte value for an input file is calculated by taking the difference between that byte value's frequency score from the input file and the frequency score from the fingerprint. If the difference between the two frequency scores is very small, then the correlation strength should increase toward 1. If the difference is large, then the correlation strength should decrease toward 0. Therefore, if a byte value always occurs with exactly the same frequency, the correlation strength should be 1. If

the byte value occurs with widely varying frequencies in the input files, then the correlation strength should be nearly 0.

One simple function that would accomplish this is the following linear equation:

$$F(x) = 1 - |x|$$

This function produces the graph shown in Figure 2-6.

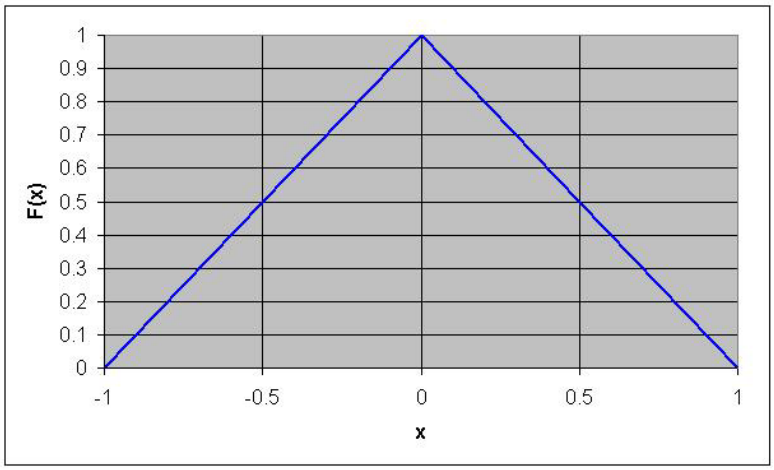

Figure 2-6 - Graph of a linear correlation strength function

Another function that would provide more tolerance for small variations and less tolerance for larger variations is a bell curve with a peak magnitude of 1 and the peak located at 0 on the horizontal axis. The general equation of a bell curve is:

$$F(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)}$$

This gives a curve for which the area under the curve equals 1. To get a curve with a peak magnitude of 1, the equation reduces to:

$$F(x) = e^{\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)}$$

Since the center of the curve should be located at 0, the equation further reduces to:

$$F(x) = e^{\left(\frac{-x^2}{2\sigma^2}\right)}$$

where *F(x)* is the correlation factor between the new byte difference and the average byte difference in the fingerprint, and *x* is the difference between the new byte value frequency and the average byte value frequency in the fingerprint.

Figure 2-7 shows the plot of this equation for $\sigma = 0.125$. This curve provides a rapid drop-off to a low correlation factor as the absolute value of *x* increases.

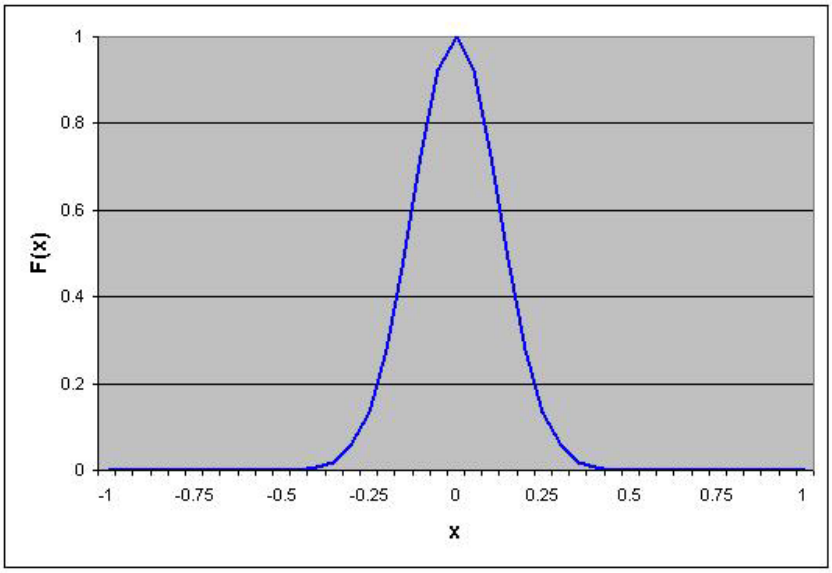

Figure 2-7 - Bell curve for $\sigma = 0.125$.

Tests were run, discussed in Chapters 3 and 4, to compare the linear equation to the bell curve, as well as to determine the bell curve's optimal value of σ for the most accurate file type recognition.

Once the input file's correlation factor for each byte value is obtained, these values need to be combined with the correlation strengths in the fingerprint. This is accomplished by using the following equation, which directly parallels the method used to calculate the frequency distribution scores:

$$NewCorr.Strength = \frac{(Old\ Corr.Strength \times previous\ number\ of\ files) + NewCorr.Factor}{previous\ number\ of\ files + 1}$$

As with the frequency distribution scores, this results in a simple average, where the previous correlation strength is weighted by the number of files already loaded into the fingerprint.

Comparing a Single File to a Fingerprint

When identifying a file using the byte frequency analysis option, the unknown file's byte frequency distribution must be compared to the byte frequency scores and the associated correlation strengths stored in each file type fingerprint.

First, a score must be generated for each fingerprint identifying how closely the unknown file matches the frequency distribution in the fingerprint. Another important step is to generate an "assurance level" for each fingerprint. The assurance level indicates how much confidence can be placed on the score. This is intended to differentiate between file types that have characteristic byte frequency distribution and those that do not.

The score is generated by comparing each byte value frequency from the unknown file with the corresponding byte value frequency from the fingerprint. As the difference between these values decreases, the score should increase toward 1. As the difference increases, the score should decrease toward 0.

Both the linear and bell curve equations discussed in Option 1 would satisfy this requirement. Each was used to generate the score for the byte frequency component of the fingerprint, and tests were run to identify the optimal equation as described in Chapters 3 and 4.

The file type's byte frequency correlation strengths can be used to generate a numeric rating for the assurance level. This is because a file type with a characteristic byte frequency distribution will have high correlation strengths for many byte values. On the other hand, a file type that does not have a characteristic byte frequency distribution will have much more variety, resulting in very low correlation strengths for most byte values.

The assurance level can be computed by a simple average of the correlation strengths for each byte value. Figure 2-8 shows the byte frequency distribution for the HTML fingerprint, with the frequency scores in blue and the correlation strengths in red. Figure 2-9 shows the byte frequency distribution scores and correlation strengths for the ZIP fingerprint.

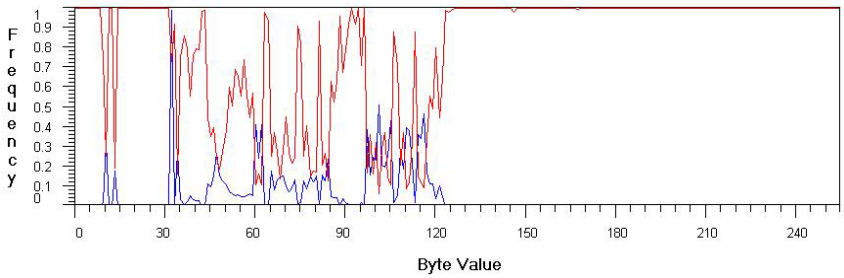

Figure 2-8 - Byte frequency distribution with correlation strength for HTML fingerprint.

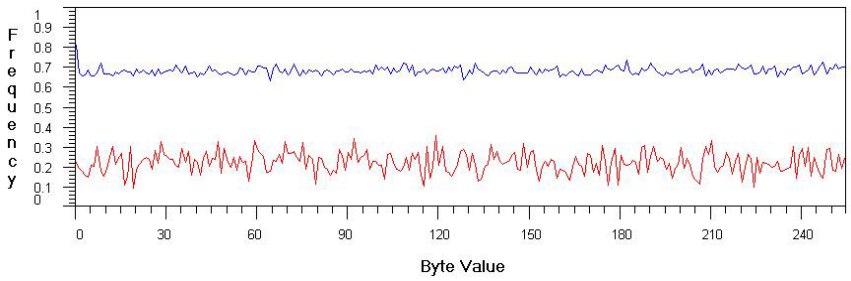

Figure 2-9 - Byte frequency distribution with correlation strength for ZIP fingerprint.

Using this scheme, the HTML file format would have a high assurance level for the byte frequency, since many byte values have high correlation strengths, whereas the ZIP file format would have a low assurance level for the byte frequency, suggesting that perhaps other options should be used to improve accuracy for this type.

Option 2: Byte Frequency Cross-Correlation

While Option 1 compares overall byte frequency distributions, other characteristics of the frequency distributions are not addressed. One example can be seen in Figure 2-8. There are two equal-sized spikes in the blue frequency scores at byte values 60 and 62, which correspond to the ASCII characters "<" and ">" respectively. Since these two characters are used as a matched set to denote HTML tags within the files, they normally occur with nearly identical frequencies.

This relationship, or cross-correlation, between byte value frequencies can be measured and scored as well, strengthening the identification process. This section describes the methods used to build the byte frequency cross-correlations of individual files, to combine the ratings from multiple files into a fingerprint representative of the file type, and to compare an unknown file to a file type fingerprint, obtaining a numeric score.

Building the Byte Frequency Cross-Correlation

In order to characterize the relationships between byte value frequencies, a two-dimensional cross-correlation array is built. Since each byte value can have a value between 0 and 255, a 256×256 array is used, with indices ranging from 0 to 255 in each dimension.

There are two key pieces of information that need to be calculated concerning the byte frequency cross-correlation analysis. First, the average difference in frequency between all byte pairs must be calculated. Second, as with the byte frequency analysis described in Option 1, a correlation strength can be calculated identifying how consistent

the frequency difference is across multiple files. Byte value pairs that have very consistent frequency relationships across files, such as byte values 60 and 62 in HTML files as mentioned above, will have a high correlation strength score. Byte value pairs that have little or no relationship will have a low correlation strength score.

Note that half of the array contains redundant information. If byte value i is being compared to byte value j, then array entry (i, j) contains the frequency difference between byte values i and j while array entry (j, i) contains the difference between byte values j and i. Since these two numbers will simply be negatives of each other, storing both of them is unnecessary. This frees half of the array to store other information.

The available space can be used to store the correlation strengths of each byte value pair. So now if byte value i is being compared to byte value j, then array entry (i, j) contains the frequency difference between byte values i and j while array entry (j, i) contains the correlation strength for the byte pair.

Furthermore, a byte value will always have an average frequency difference of 0 and a correlation strength of 1 with itself, so the main diagonal of the array does not need to be used. These positions can be used to store any other information that is needed for the comparisons. The only additional number needed is the total number of files that have been added into the fingerprint. The first entry of the main diagonal (0, 0) is therefore used to store the number of files that have been added.

Using the single array to store the different forms of data saves on memory space, as well as processing time, since redundant calculations don't have to be performed. The final array structure is shown in Figure 2-10.

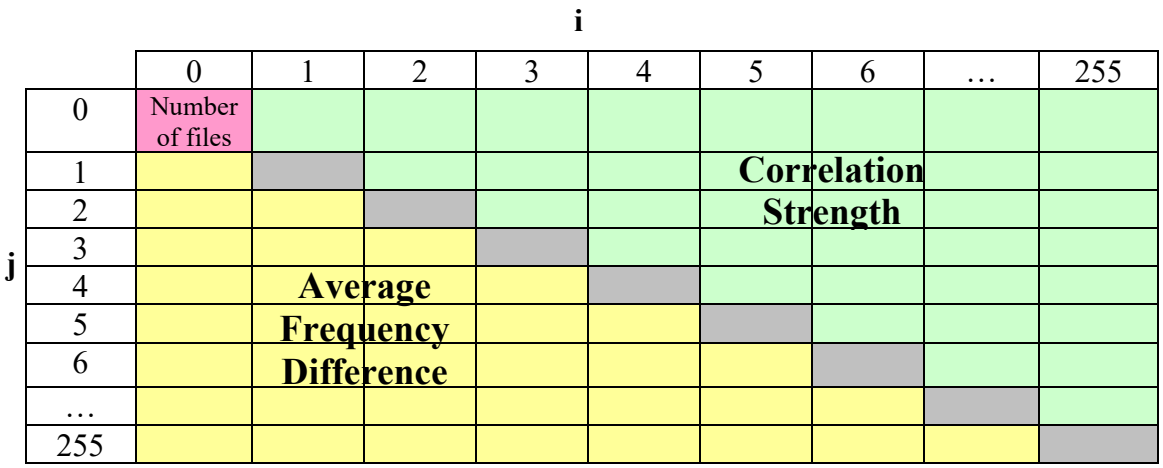

Figure 2-10 - Byte cross-correlation array structure

Calculating the difference between the frequencies of two bytes with values i and j involves simply subtracting the frequency score of byte value i from the frequency of byte value j. Since byte value frequencies were normalized, with a range of 0 to 1, this results in a number with a possible range of –1 to 1. A score of –1 indicates that the frequency of byte value i was much greater than the frequency of byte value j. A score of 1 indicates that the frequency of byte value i was much less than the frequency of byte value j. A score of 0 indicates that there was no difference between the frequencies of the two byte values.

Combining Cross-Correlations into a Fingerprint

Once the frequency differences between all byte-value pairs for an input file have been calculated, they can be added into a fingerprint. To accomplish this, the fingerprint is loaded, and the frequency differences for each byte value pair in the new file's array are combined with the corresponding element of the fingerprint's array using the following equation:

$$New\,FP\,difference = \frac{\left(Old\,FP\,difference \times previous\ \ number\,of\ \ files\right) + New\,freq.\,difference}{previous\ \ number\,of\ \ files + 1}$$

This is the same equation that was used to combine frequency distributions in Option 1, and it results in a simple average, where the previous byte value frequency difference is weighted by the number of files already loaded into the fingerprint.

A correlation factor can be calculated for each byte value pair, by comparing the frequency differences in the input file to the frequency differences in the fingerprint. The

correlation factors can then be combined with the scores already in the fingerprint to form an updated correlation strength score for each byte value pair.

If no files have previously been added into a fingerprint, then the correlation factor for each byte value pair is set to 1. This makes sense, because correlation strength is, in essence, a measure of the consistency between files of the same type. If only one file exists in the fingerprint, there is no variation at all, and all byte value pairs show 100% consistency. Of course, with only one file, this is a trivial fingerprint that is not an adequate representation of the file type as a whole. As additional files are added, the correlation strengths are then revised to more accurately reflect the file type.

If at least one file has been previously added into a fingerprint, then the correlation factor for each byte value pair is calculated by subtracting the pair's frequency difference from the new file and the same pair's average frequency difference from the fingerprint. This results in a new overall difference between the new file and the fingerprint. If this overall difference is very small, then the correlation strength should increase toward 1. If the difference is large, then the correlation strength should decrease toward 0.

These are the same conditions that were present in the calculations for Option 1, and the same two equations can be used to generate the new correlation strengths. Namely, the following simple linear equation can be used:

$$F(x) = 1 - |x|$$

where F(x) is the correlation factor of the byte value pair and |x| is the absolute value of the frequency difference. (See Figure 2-6 for the graph of this function.) Alternatively, the following bell-curve equation can be used to provide a more rapid drop-off:

$$F(x) = e^{\left(\frac{-x^2}{2\sigma^2}\right)}$$

where F(x) is the correlation factor of the byte value pair. (See Figure 2-7 for the graph of this function for $\sigma = 0.125$.) Chapters 3 and 4 describe tests that were run comparing the linear and bell curve methods, as well as testing for the optimal value of σ in the bell curve equation.

Once the input file's correlation factor for each byte value pair is obtained, these values need to be added into the correlation strengths in the fingerprint. This is accomplished by using the same equation that was used to calculate new correlation strengths from Option 1:

$$New Corr.Strength = \frac{\left(Old\ Corr.Strength \times previous\ number\ of\ files\right) + New Corr.Factor}{previous\ number\ of\ files + 1}$$

As with the correlation strengths from Option 1, this results in a simple average, where the previous correlation strength is weighted by the number of files already loaded into the fingerprint.

After the average frequency differences and correlation strengths for each byte value pair of the new input file have been updated in the fingerprint, the Number of Files field is incremented by 1 to indicate the addition of the new file.

It is interesting to compare the frequency distribution graphs from Option 1 to the byte frequency cross-correlation plots generated from Option 2. Figure 2-8 shows the frequency distribution for the HTML file format, and Figure 2-11 shows a graphical plot of the HTML fingerprint cross-correlation array. Note that there are ranges of byte values in the frequency distribution that never occurred in any files (they appear with 0 frequency.) These regions appear in the cross-correlation plot as white regions of 0 frequency difference, and dark green regions with a correlation strength of 1. Furthermore, the intersection of 60 on the vertical axis with 62 on the horizontal axis (corresponding to the ASCII values for the "<" and ">" characters as mentioned above) shows a dark green dot representing a correlation strength of 1, as expected.

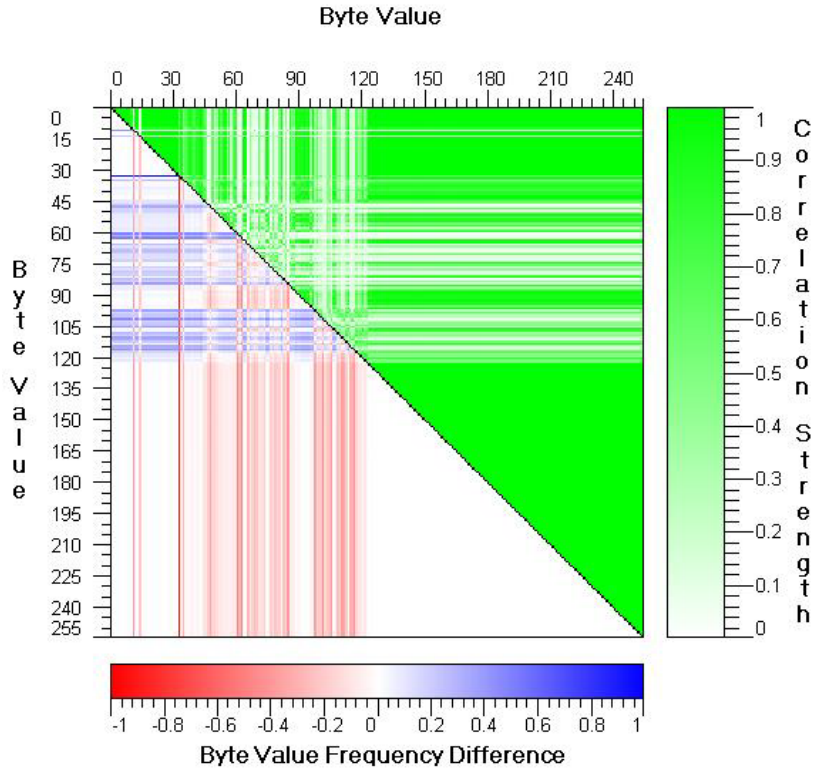

Figure 2-11 - Byte frequency cross-correlation plot for the HTML file type

Similarly, Figure 2-2 shows the frequency distributions of two GIF files, and Figure 2-12 shows a graphical plot of the GIF fingerprint cross-correlation array. The sawtooth pattern shown in the frequency distributions are a characteristic feature of the GIF file type, and it manifests in the cross-correlation plot as a subtle grid pattern in the frequency difference region. It is also clear that the regions of high and low correlation strength are much more evenly distributed in the GIF file type than the HTML file type.

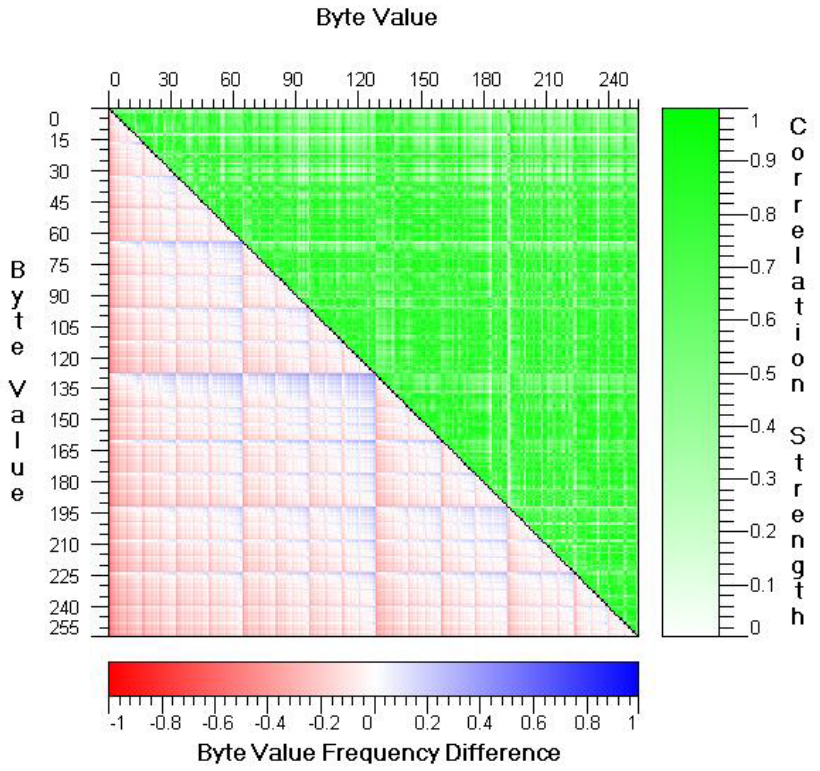

Figure 2-12 - Byte frequency cross-correlation plot for the GIF file type.

Comparing a Single File to a Fingerprint

When identifying a file using the byte frequency cross-correlation option, the unknown file's cross-correlation array must be generated and compared to the cross-correlation scores and correlation strengths stored in each file type fingerprint.

As with Option 1, a score is generated for each fingerprint identifying how closely the unknown file matches the fingerprint. An assurance level is also generated to indicate how much confidence can be placed on the score. File types that have characteristic cross-correlation patterns should have high assurance levels, while those that do not have characteristic cross-correlation patterns should have low assurance levels.

The score is generated by comparing the frequency difference for each byte value pair from the unknown file with the average frequency difference for the corresponding byte value pair from the fingerprint. As the difference between these values decreases, the score should increase toward 1. As the difference increases, the score should decrease toward 0.

Both the linear and bell curve equations discussed in Option 1 would satisfy this requirement. Each was used to generate the score for the byte frequency component of the fingerprint, and tests were run to identify the optimal equation. These tests are described in Chapters 3, and their results are discussed in Chapter 4.

As with Option 1, the correlation strengths are used to generate a numeric rating for the assurance level. The assurance level is computed as a simple average of the correlation strengths of each byte value pair. The higher the assurance level, the more weight can be placed on the score for that fingerprint.

Option 3: File Header/Trailer Analysis

Options 1 and 2 make use of byte value frequencies to characterize and identify file types. While these characteristics can effectively identify many file types, some do not have easily identifiable patterns. To address this, the file headers and file trailers can be analyzed and used to strengthen the recognition of many file types. The file headers and trailers are patterns of bytes that appear in a fixed location at the beginning and end of a file respectively. These can be used to dramatically increase the recognition ability on file types that do not have strong byte frequency characteristics.

This section describes the methods used to build the header and trailer profiles of individual files, to combine the ratings from multiple files into a fingerprint for the file type, and to compare an unknown file to a file type fingerprint, obtaining a numeric score.

Building the Header and Trailer Profiles

The first step in building header and trailer profiles is to decide how many bytes from the beginning and end of the file will be analyzed. If H is the number of file header bytes to analyze, and T is the number of trailer bytes to analyze, then two two-dimensional arrays are built, one of dimensions $H \times 256$, the other of dimensions $T \times 256$. Figure 2-13 shows the structure of the array used for analysis of the file header. For each byte position in the file header, all 256 byte values can be independently scored based upon the frequency with which the byte value occurs at the corresponding byte position.

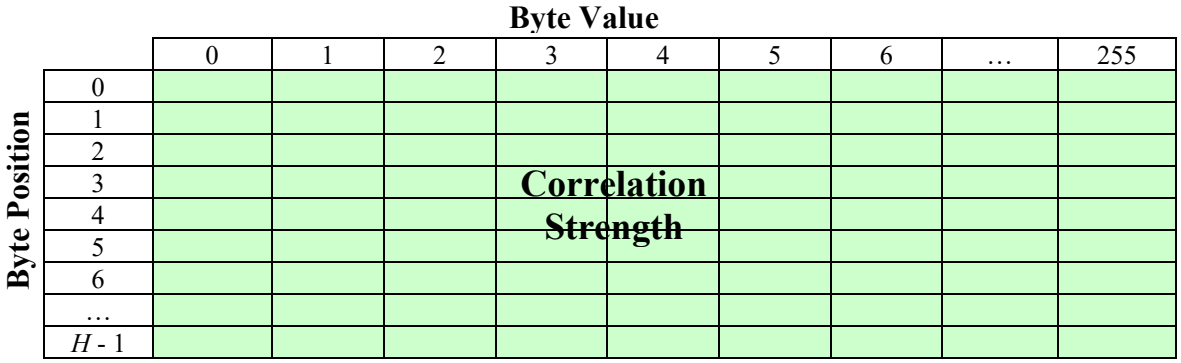

Figure 2-13 - Array structure used for header analysis

The array structure used for analysis of the file trailer would have the same basic structure. The only difference is that the byte position dimension would range from 0 to T – 1 rather than H – 1.

An individual file's header array is filled by looping through each byte in the header, from byte 0 (the first byte in the file) to byte H – 1. For each byte position, the array entry corresponding to the value of the byte is filled with a correlation strength of 1. All other byte value entries in that row are set to 0. After the entire array is filled, each byte position row is filled with a single 1 corresponding to the byte value at that byte position.

The only exception occurs when an input file is shorter than the header or trailer lengths. In this case, the fields in the missing byte position rows will be filled with the value -1 to signify no data. (Note that if a file length is greater than the header and trailer lengths, but less then the sum of the two lengths, then the header and trailer regions will overlap.)

The file trailer array is similarly filled by looping through the last T bytes of the file. Again, for each byte position the array entry corresponding to the value of the byte is filled with a value of 1. All other byte value entries in that row are set to 0.

Combining Header and Trailer Profiles into a Fingerprint

The two dimensional array is an extremely inefficient way to store the data for a single file, but it becomes useful when many files are combined into a single fingerprint. This can be accomplished by simply averaging the correlation strength values from each file into the fingerprint using the following equation, which is similar to the ones used for Options 1 and 2:

$$New\,FP\,array\,entry = \frac{\left(Old\,FP\,array\,entry \times previous\;\; number\,of\;\; files\right) + New\,array\,entry}{previous\;\; number\,of\;\; files + 1}$$

This provides a simple average, where the previous fingerprint correlation strength is weighted by the number of files already loaded into the fingerprint. A sample graphical plot of the file header array is shown in Figure 2-14 for the GIF file type.

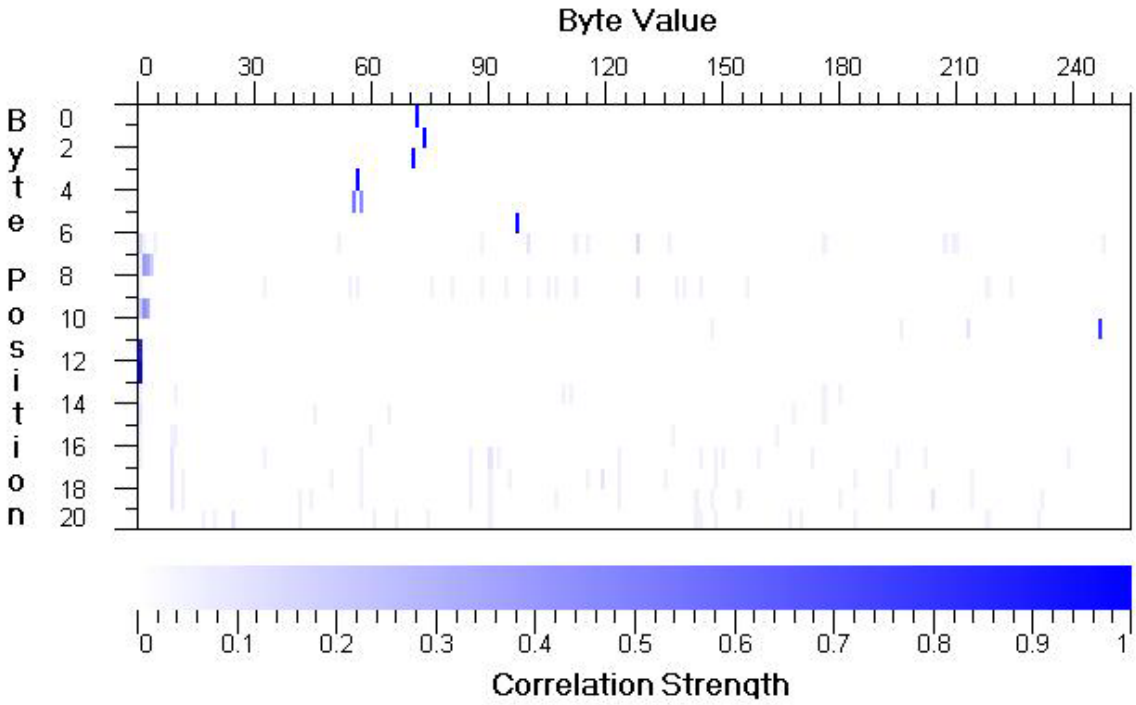

Figure 2-14 - File header plot for the GIF file fingerprint

The first few bytes of the GIF header show high correlation strengths (represented by dark blue marks,) indicating that this type has a strongly characteristic file header. The specification for the GIF format states that the files shall all begin with the text string "GIF87a" for an earlier version of the format, or "GIF89a" for a later version.

Further inspection of Figure 2-14 that byte position 0 has a correlation strength of 1 for byte value 71, which corresponds to the ASCII character "G". The second byte position has a correlation strength of 1 for byte value 72, which corresponds to the ASCII character "I". This continues through byte values 70 (ASCII "F") and 56 (ASCII "8"). At byte position five, though, byte values 55 and 57 both show correlation strengths roughly balanced. This indicates that these byte values, which correspond to the ASCII values for "7" and "9" respectively, occurred with almost equal frequency in the input files. (This further indicates that approximately equal numbers of files of each version of the GIF format were loaded into the fingerprint.) Finally, byte position six shows a correlation strength of 1 for byte value 97, corresponding to the ASCII character "a".

Beyond byte position six, there is a much broader distribution of byte values, resulting in lower correlation strengths and lighter marks on the plot.

Figure 2-15 shows a very similar plot of the file trailer for the MPEG file type fingerprint, where the end of the file is represented by byte position 0 at the bottom of the plot. This plot shows a broad distribution of byte values (resulting in extremely faint marks) up until four bytes from the end of the file. These final four bytes show a characteristic pattern similar to the pattern described above for the GIF file header.

Note that for file types that do not have a characteristic file header or trailer, the corresponding plots would appear essentially empty, with many scattered dots with very low correlation strengths (therefore producing almost white dots.)

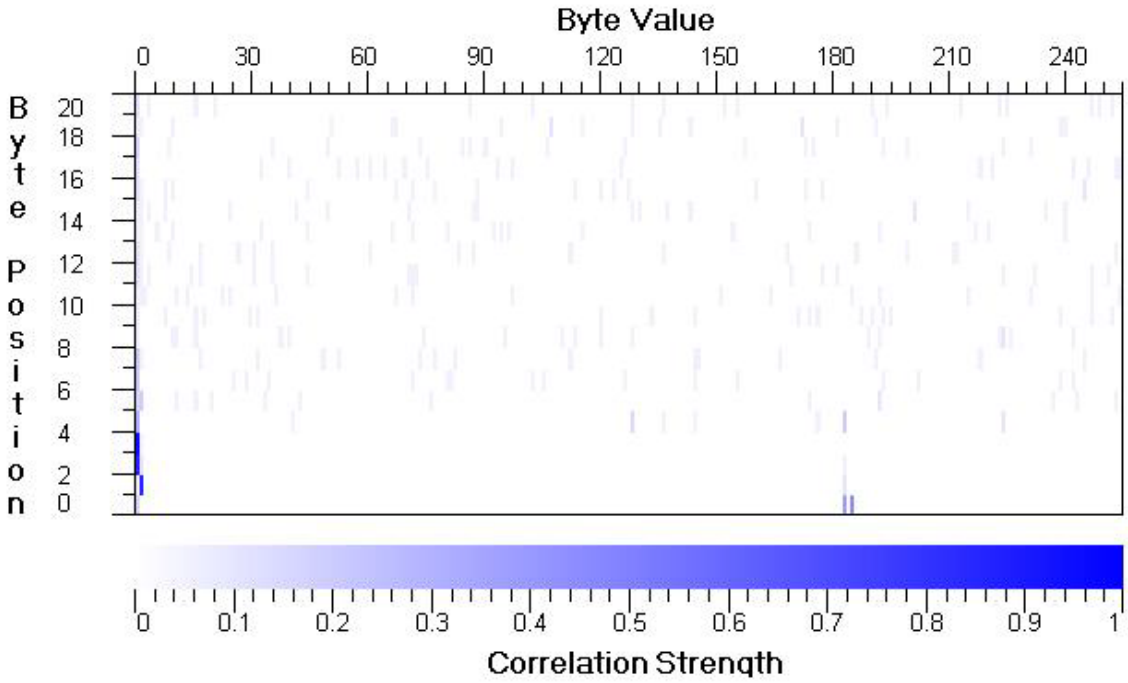

Figure 2-15 - File trailer plot for the MPEG file type.

Comparing a Single File to a Fingerprint

The first step in using file headers and trailers to identify a file is to fill header and trailer arrays for the unknown file as described above. This will result in an array for which each byte position row has all fields filled with 0, except for a single field filled with 1 corresponding to the value of the byte at that position in the file.

For each byte position in the header and trailer, the byte value of the field set to 1 is extracted. The following equation is then used to generate the score for the file header or trailer:

$$\overline{S} = \frac{C_1 G_1 + C_2 G_2 + \ldots + C_n G_n}{G_1 + G_2 + \ldots + G_n}$$

where C is the correlation strength for the byte value extracted from the input file for each byte position, and G is the correlation strength of the byte value in the fingerprint array with the highest correlation strength for the corresponding byte position.

This equation produces an average of the correlation strengths of each byte value from the input file, weighted by the greatest correlation strength at each byte position. The result of this is to place greatest weight on those byte positions with a strong correlation, indicating that they are part of a characteristic file header or trailer, and to place much less weight (ideally no weight) on values where the file type does not have consistent values.

The GIF file header plot in Figure 2-14 provides a clear example. The first four byte positions each have a correlation strength of 1 for a single byte. This indicates that all input files of the GIF file type had the same byte values for these positions. If an unknown file has different bytes in these positions, it is a very strong indicator that it is not a GIF file. On the other hand, if the unknown file has a differing byte value at position 20, which shows a very low correlation strength, this offers no real information about whether the unknown file is a GIF file or not since there are no bytes in this position with a high correlation strength.

The assurance level for the file header and file trailer is simply set equal to the overall maximum correlation strength in the header and trailer arrays, respectively. This is different from the approach used in Options 1 and 2, where the average of all correlation strengths was used.

Setting the assurance level equal to the maximum correlation strength allows even a few bytes with very high correlation strength, such as those in the GIF file format to provide a strong indication of file type. Therefore even a few bytes can produce a strong influence in recognition. On the other hand, if a file type has no consistent file header or trailer, the maximum correlation strength will be very low. This means little weight will be placed on the header or trailer with the low assurance level.

Combining Scores from Multiple Options

If more than one option is used, then the option scores must be combined into a single overall rating for each fingerprint. The fingerprint with the highest overall rating is then selected as the most likely match of the file type.

The following equation is used to combine the scores from each option:

$$O = \frac{S_{freq} A_{freq} + S_{corr} A_{corr} + S_{header} A_{header} + S_{trailer} A_{trailer}}{A_{freq} + A_{corr} + A_{header} + A_{trailer}}$$

where O is the overall fingerprint score, S is the score for each comparison mode, and A is the assurance level of each assurance mode.

This produces an average of each score, weighted by the score's assurance level. Therefore, if a file type has a strongly characteristic frequency distribution and cross-correlation, but no characteristic file header or trailer, the frequency distribution and cross-correlation will be counted toward the overall score far more than the header and trailer data. This allows each fingerprint to produce a combination of byte frequency distribution, byte frequency cross-correlation, file header profile, and file trailer profile that is customized to provide the optimal recognition for its specific file type.

CHAPTER 3: TESTING

The following sections describe the tests that were run to identify optimal values for equation constants, as well as quantifying the accuracy of the algorithm across a range of file types.

Test 1 – Beta Sweep

The goal of the β sweep test is to determine the optimal value of the constant β in the following equation, used in the Option 1, byte frequency analysis:

$$y = x^{\left(\frac{1}{\beta}\right)}$$

The optimal value of β is defined as the value that produces the greatest difference between the fingerprint with the highest frequency score and the fingerprint with the second-highest frequency score.

Different values of β between 0.5 and 2.5 in increments of 0.5 are used to find the optimum β. For each value of β, a set of fingerprint files are generated for 15 representative file types using a constant library of 20 input files for each file type. (See APPENDIX B for a list of the file types used.) A test file is then chosen at random from a different set of files, and an identification report is run for the test file against the collection of fingerprints generated for each value of β. (See

APPENDIX F for an example of an identification report.) Since the equation is a mathematical function that affects each frequency calculation in a uniform manner, regardless of file type, the test file should adequately represent the effects of the various values of β in general.

For the test file, a grid is filled with the frequency score for each fingerprint for each value of β. The difference between the highest and second-highest scoring fingerprints are then calculated and plotted for each sample file. The peak value is identified as the optimal value of β.

Test 2 – Sigma Sweep

The goal of this test is to determine the optimal value of the constant σ in the following equation, used in the Option 2, byte frequency cross-correlation analysis:

$$F(x) = e^{\left(\frac{-x^2}{2\sigma^2}\right)}$$

Similar to the β sweep test, the optimal value of σ is defined as the value that produces the greatest difference between the fingerprint with the highest cross-correlation score and the second-highest cross-correlation score.

The value of σ is varied exponentially between 0.0015625 and 0.8, doubling the value at each increment. For each value, a set of fingerprint files is generated for the same 15 representative file types used in the β sweep test, using the same constant library of 20 input files for each type. (See APPENDIX B for a list of the file types used.) A test file is then chosen at random, and an identification report is run against the collection of fingerprints generated for each value of σ. As with the β sweep, the equation tested is a mathematical function that affects each calculation in a uniform manner, regardless of file type, so the test file should adequately represent the effects of the various values of σ in general.

For the test file, a grid is filled with the cross-correlation score for each fingerprint for each value of σ. The difference between the highest and second-highest scoring fingerprints is then calculated and plotted for each sample file.

Because such a broad range of values is covered by the exponential variance of σ, it does not provide adequate resolution around each data point. To address this, a smaller linear range of σ values is selected that brackets the peak value identified in the first test. The sweep test is then rerun using the new smaller linear range of σ in place of the initial exponential range.

The difference results from the second running of the test provides higher resolution that will more accurately identify the optimal value of σ .

Test 3 – Linear Correlation vs. Bell

The goal of this test is to compare the results of the bell curve equation tested in the σ sweep test above with the results obtained by using the following linear equation instead:

$$F(x) = 1 - |x|$$

To accomplish this comparison, all bell curve equations are replaced with this linear equation, and a set of fingerprint files are generated using the same file types and input file that were used in the σ sweep test. (See APPENDIX B for a list of the file types used.) An identification report is then run against the same test file used in the σ sweep test.

The difference between the highest and second-highest scoring fingerprints is then calculated and compared to the results of the σ sweep test to determine which method offers the strongest recognition.

Test 4 – Header Length Sweep

The goal of this test is to find an optimal value of header length. Unlike the preceding three tests, the header length is not likely to affect all files or file types in the same manner. This is because some file types have very short file headers, while others have longer file header and some have no file headers at all. Each file type behaves differently in the analysis as the number of header bytes changes. Because of this, a much more comprehensive set of data needs to be developed and analyzed than was necessary for the preceding three tests.

The same 15 representative file types as the previous tests are used to perform the header length sweep test. (See APPENDIX B for a list of the file types used.) For each file type, four test files are selected at random, resulting in a library of 60 test files, equally representing each file type.

The value of the header length variable is varied from 5 to 50 in steps of 5. For each header length, a set of fingerprints is generated using the same library of 20 input files per file type used in each of the previous tests. Then, for each header length, an identification report is generated for each of the 60 test files.

This test results in a tremendous number of data points that must be analyzed to determine if there is an overall optimal header length. First, the results are broken out by file type. For the first file type, a table is generated for each header length. Each table is filled with the header scores generated by each fingerprint for each of the four input files of the file type. In each table, the scores from each file are averaged together into a single score for each fingerprint.

Next, another table is built that stores the average header score for each fingerprint for each header length. Finally, for each header length, the difference is calculated between the highest fingerprint header score and the second-highest fingerprint header score. These differences are plotted, and the header length that shows the highest difference (highest differentiation between file types) is selected as the optimal length for that file type.

This set of tables are generated for each of the 15 file types tested. Finally, the difference values from each file type are combined into a single table. The average of all file types are calculated, showing the overall optimal header length.

Test 5 – Trailer Length Sweep

The goal of this test is to find an optimal value of trailer length. As with file headers, trailer length is not likely to affect all files or file types in the same manner, because some file types have long trailers while others have short trailers or no trailer at all. Therefore, each file type is likely to behave differently in the analysis as the number of trailer bytes changes.

Because of this, essentially the same set of tests are run for file trailers as was run for the file headers. The same 15 representative file types are used to perform the trailer length sweep test. (See APPENDIX B for a list of the file types used.) The same set of 60 test files are used, representing four files of each file type.

The value of the trailer length variable is varied from 5 to 50 in steps of 5. For each trailer length, a set of fingerprints is generated using the same library of 20 input files per file type used in each of the previous tests. Then, for each trailer length, an identification report is generated for each of the 60 test files.

As with the header length test, the resulting data is initially separated by file type. For each file type, a table is generated for each trailer length. Each table is filled with the trailer scores generated by each fingerprint for each of the four input files of the file type. In each table, the scores from each file are averaged together into a single score for each fingerprint.

Another table is built that stores the average trailer score for each fingerprint for each header length. Finally, for each trailer length, the difference is calculated between the highest fingerprint trailer score and the second-highest fingerprint trailer score. These differences are plotted, and the trailer length that shows the highest difference is selected as the optimal for that file type.

This set of tables is generated for each of the 15 file types tested. Finally, the difference values from each file type is combined into a single table and the average of all file types are calculated, showing the overall optimal trailer length.

Test 6 – Accuracy Test (All Options)

The goal of this test is to quantify what percentage of unknown input files are correctly identified with all Options enabled.

Twenty-five file type fingerprints are be used for this test. (See APPENDIX C for a list of file types.) Four test files are selected for each file type and are combined into a single test directory, resulting in a total library of 100 files.

An identification report is generated for each of the 100 files, with all three Options enabled. The identified type for each file is recorded and compared to the actual type of the file. A percentage is calculated, representing the percentage of input files that are correctly identified.

Test 7 – Accuracy Test (Option 1)

The goal of this test is to quantify what percentage of unknown input files are correctly identified when only Option 1 is enabled.

Twenty-five file type fingerprints are used for this test. (See APPENDIX C for a list of file types.) Four test files are selected for each file type and are combined into a single test directory, resulting in a total library of 100 files.

An identification report is generated for each of the 100 files, with only Option 1 enabled. The identified type for each file is recorded and compared to the actual type of the file. A percentage is calculated, representing the percentage of input files that are correctly identified.

Test 8 – Accuracy Test (Option 2)

The goal of this test is to quantify what percentage of unknown input files are correctly identified when only Option 2 is enabled.

Twenty-five file type fingerprints are used for this test. (See APPENDIX C for a list of file types.) Four test files are selected for each file type and are combined into a single test directory, resulting in a total library of 100 files.

An identification report is generated for each of the 100 files, with only Option 2 enabled. The identified type for each file is recorded and compared to the actual type of the file. A percentage is calculated, representing the percentage of input files that are correctly identified.

Test 9 – Accuracy Test (Option 3)

The goal of this test is to quantify what percentage of unknown input files are correctly identified when only Option 3 is enabled.

Twenty-five file type fingerprints are used for this test. (See APPENDIX C for a list of file types.) Four test files are selected for each file type and are combined into a single test directory, resulting in a total library of 100 files.

An identification report is generated for each of the 100 files, with only Option 3 enabled. The identified type for each file is recorded and compared to the actual type of the file. A percentage is calculated, representing the percentage of input files that are correctly identified.

Test 10 – Extended Accuracy Test (All Options)

The accuracy tests performed in Tests 6 through 9 tested the accuracy of the algorithm in identifying well-known file formats. In addition to the common file formats, there are many formats created by developers to interface only with one or a small number of programs.

For this test, five additional file types are added to those used in Tests 6 through 9. Each of the new file formats are proprietary types created for specific applications. (See APPENDIX D for a list of file types used for this test.)

The primary goal of this test is to quantify what percentage of unknown input files are correctly identified with all Options enabled when the proprietary file formats are added to the common formats. A secondary goal is to observe the effect of adding additional fingerprints on accuracy.

A total of thirty file type fingerprints are used for this test. Four test files are selected for each file type and are combined into a single test directory, resulting in a total library of 120 files.

An identification report is generated for each of the 120 files, with all Options enabled. The identified type for each file is recorded and compared to the actual type of the file. A percentage is calculated, representing the percentage of input files that are correctly identified.

Test 11 – Extended Accuracy Test (Option 1)

The goal of this test is to quantify what percentage of unknown input files are correctly identified when proprietary file formats are added, and only Option 1 is enabled.

Thirty file type fingerprints are used for this test. (See APPENDIX D for a list of file types.) Four test files are selected for each file type and are combined into a single test directory, resulting in a total library of 120 files.

An identification report is generated for each of the 120 files, with only Option 1 enabled. The identified type for each file is recorded and compared to the actual type of the file. A percentage is calculated, representing the percentage of input files that are correctly identified.

Test 12 – Extended Accuracy Test (Option 2)

The goal of this test is to quantify what percentage of unknown input files are correctly identified when proprietary file formats are added, and only Option 2 is enabled.

Thirty file type fingerprints are used for this test. (See APPENDIX D for a list of file types.) Four test files are selected for each file type and are combined into a single test directory, resulting in a total library of 120 files.

An identification report is generated for each of the 120 files, with only Option 2 enabled. The identified type for each file is recorded and compared to the actual type of the file. A percentage is calculated, representing the percentage of input files that are correctly identified.

Test 13 – Extended Accuracy Test (Option 3)

The goal of this test is to quantify what percentage of unknown input files are correctly identified when proprietary file formats are added, and only Option 3 is enabled.

Thirty file type fingerprints are used for this test. (See APPENDIX D for a list of file types.) Four test files are selected for each file type, and combined into a single test directory, resulting in a total library of 120 files.

An identification report is generated for each of the 120 files, with only Option 3 enabled. The identified type for each file is recorded and compared to the actual type of the file. A percentage is calculated, representing the percentage of input files that are correctly identified.

This page intentionally left blank.

CHAPTER 4: RESULTS

Test 1 – Beta Sweep

A Word Perfect document (WPD) was selected at random as the test file used for this test. The value of β was then varied from 0.5 to 2.5, and the byte frequency scores were recorded for each fingerprint at each value of β. Figure 4-1 shows the resulting graph of byte frequency scores per fingerprint for each β.

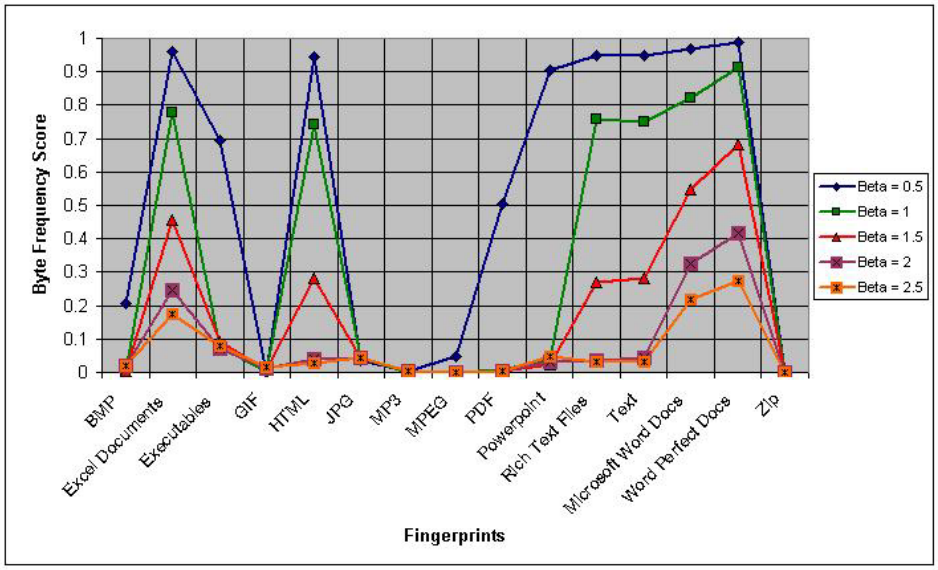

Figure 4-1 - Frequency scores per fingerprint for varying values of β.

This graph shows that lower values of β produce higher scores on average, resulting in more scores clustered closer together in the higher score range. Higher values of β produce lower scores on average, resulting in more scores clustered closer together in the lower score range.

Figure 4-2 plots the difference between the scores of the highest and second highest fingerprints for each value of β. The optimal value of β is defined as the one that produces the largest difference between these two byte frequency scores.

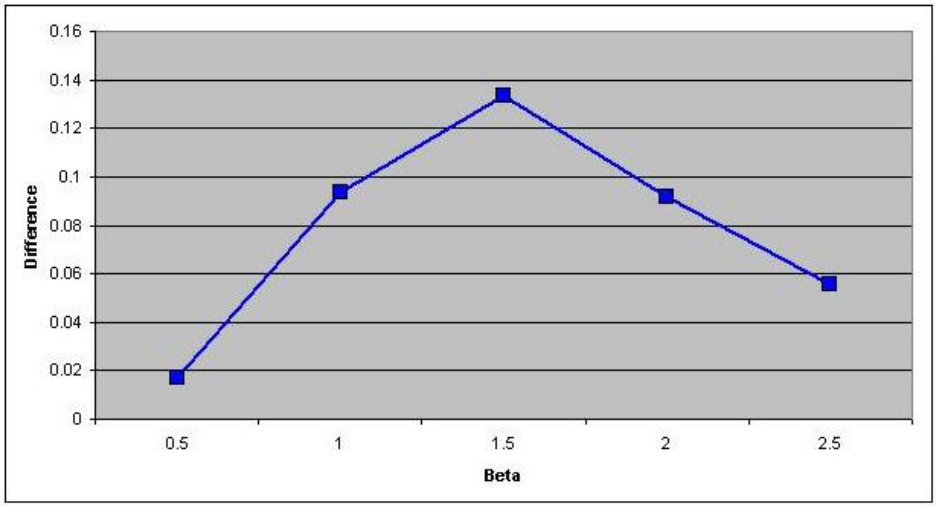

Figure 4-2 - Difference between first- and second-rated fingerprints as a function of β.

It is clear from Figure 4-2 that the largest differentiation between the highest and second-highest frequency scores occurs at $\beta = 1.5$.

Tests 2 and 3– Sigma Sweep and Linear Correlation vs. Bell

A bitmap (BMP) file was selected at random as the test file used for Test 2. The value of σ was then exponentially varied from 0.001563 to 0.8, doubling at each step, and the cross-correlation scores were recorded for each fingerprint at each value of σ. For Test 3, the bell-curve equation was replaced with the linear equation given in the test description. An identification report was then generated for the same bitmap file that was used in Test 2. The results of each test were then combined into a single graph, shown in Figure 4-3, that shows cross-correlation scores per fingerprint for each value of σ and for the linear equation.

Very low values of σ resulted in low cross-correlation scores with little variation between fingerprints. Conversely, very high values of σ resulted in high cross-correlation scores, also with little variation between fingerprints. The largest variations in cross-correlation scores occurred when σ was in the "middle" ranges, between 0.0125 and 0.1.

The linear equation produced differentiation that was worse than all values of σ except for $\sigma = 0.8$.

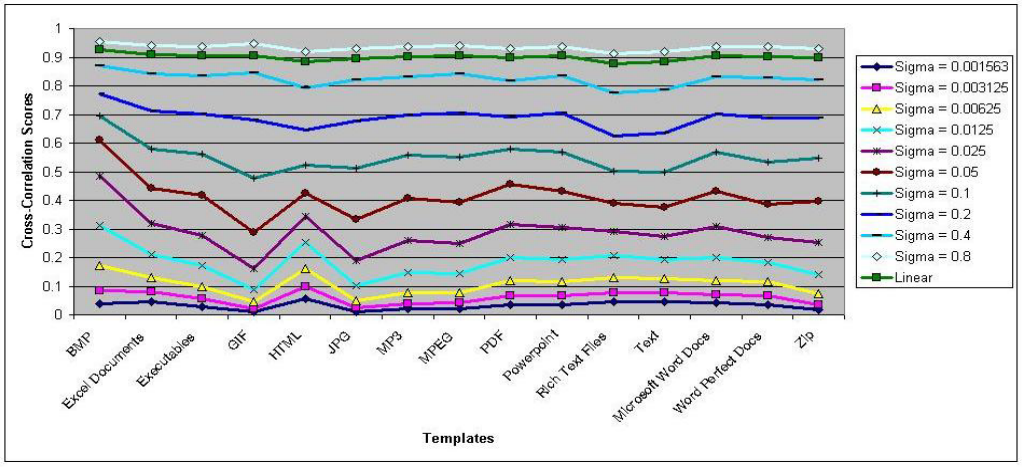

Figure 4-3 – Cross-correlation scores for each fingerprint as a function of σ.

Because the exponential scale for σ was used to cover such a broad range of values, the resolution around any one value is rather poor. Therefore, the test was rerun varying σ between 0.0125 and 0.1 in linear steps of 0.0125. Figure 4-4 graphs the resulting cross-correlation scores per fingerprint for each σ. Again, the results given by the linear equation are superimposed for comparison.

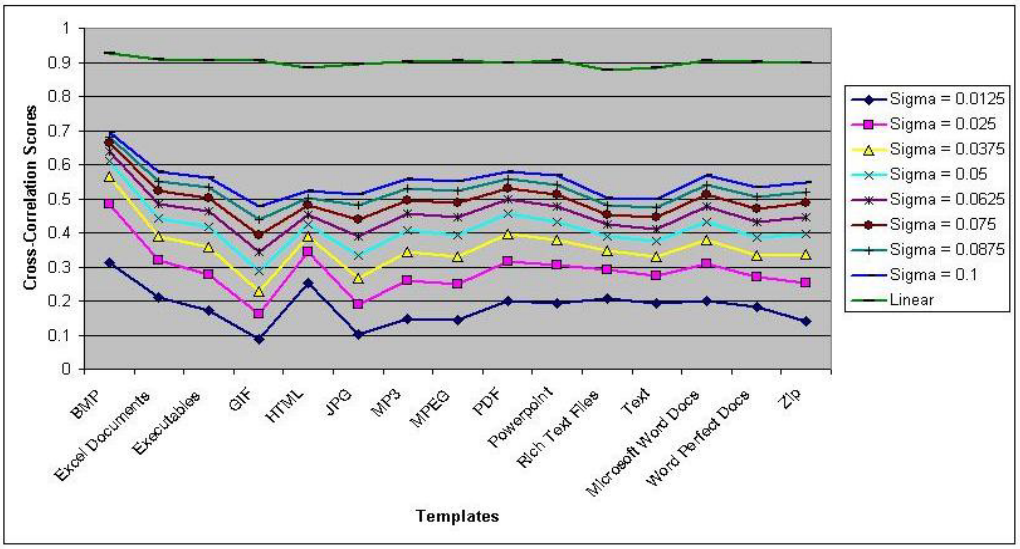

Figure 4-4 - Cross-correlation scores for each fingerprint as a function of σ.

The differences between the highest and second-highest scores were then calculated for each value of σ. Figure 4-5 shows the plot of these differences as a function of σ.

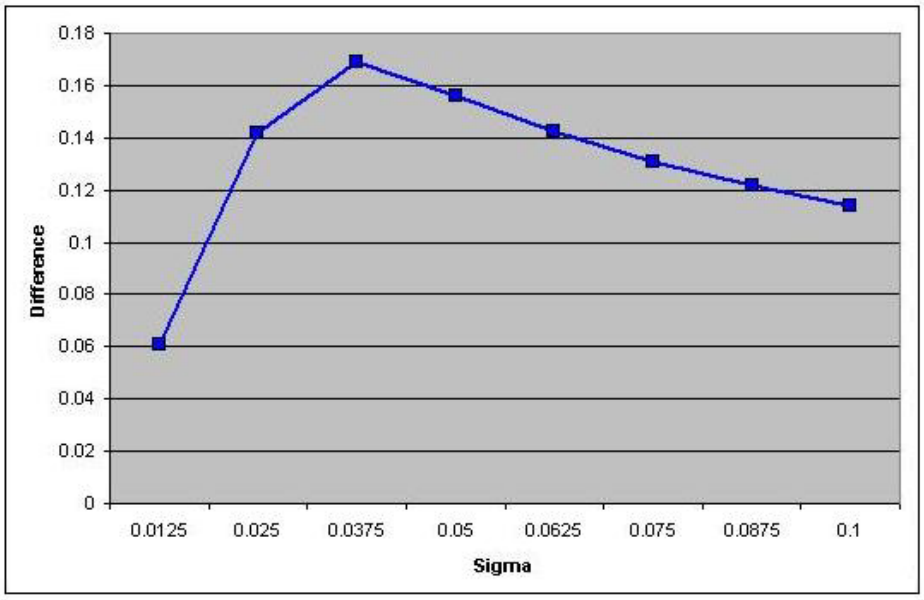

Figure 4-5 - Differences between first- and second-rated fingerprints as a function of σ.

This graph suggests the value of σ that produces the highest level of differentiation between the highest and second-highest cross-correlation scores is σ = 0.0375.

Test 4 – Header Length Sweep

Since header lengths affect the scores of each file type differently, depending upon the type and length of their file headers, all 60 test files were included in the header length sweep test (4 test files for each of the 15 file types.) The header length was varied from 5 to 50 in linear steps of 5, and recognition reports were generated for each value.

First, a set of tables was generated for each file type for each header length. Figure 4-6 shows a sample table for the MP3 file format and header length 5.

File Type: MP3 Header Length: 5	Effective Score 1	Effective Score 2	Effective Score 3	Effective Score 4	Average Effective Score
BMP	0.2763513	0.2763513	0.2783783	0.2783783	**0.2773648**
XLS	0.01583333	0.01583333	0.01583333	0.01583333	**0.01583333**
EXE	0.173913	0.3147826	0.173913	0.173913	**0.2091304**
GIF	0	0	0	0	**0**
HTML	0	0	0	0	**0**
JPG	0.1666667	0.3333333	0.1666667	0.1666667	**0.20833335**
MP3	0.7247059	0.5535294	0.7511764	0.7511764	**0.695147025**

MPEG	0.1489691	0.1489691	0.1489691	0.1489691	**0.1489691**
PDF	0	0	0	0	**0**
PPT	0	0	0	0	**0**
RTF	0	0	0	0	**0**
TXT	0.003353658	0	0.003353658	0.003353658	**0.002515244**
DOC	0	0	0	0	**0**
WPD	0	0.217057	0	0	**0.05426425**
ZIP	0.1583333	0.1583333	0.1583333	0.1583333	**0.1583333**

Figure 4-6 - Table of file header effective scores for the four MP3 test files.

The header effective scores were listed for each of the four test files per file type. These effective scores were then averaged together into a single effective score per file type for the given header length.

A similar table was built for each header length. Another table was then built out of the average effective scores per fingerprint for each header length**.** Figure 4-7 shows a graph of the table produced for the MP3 file header.

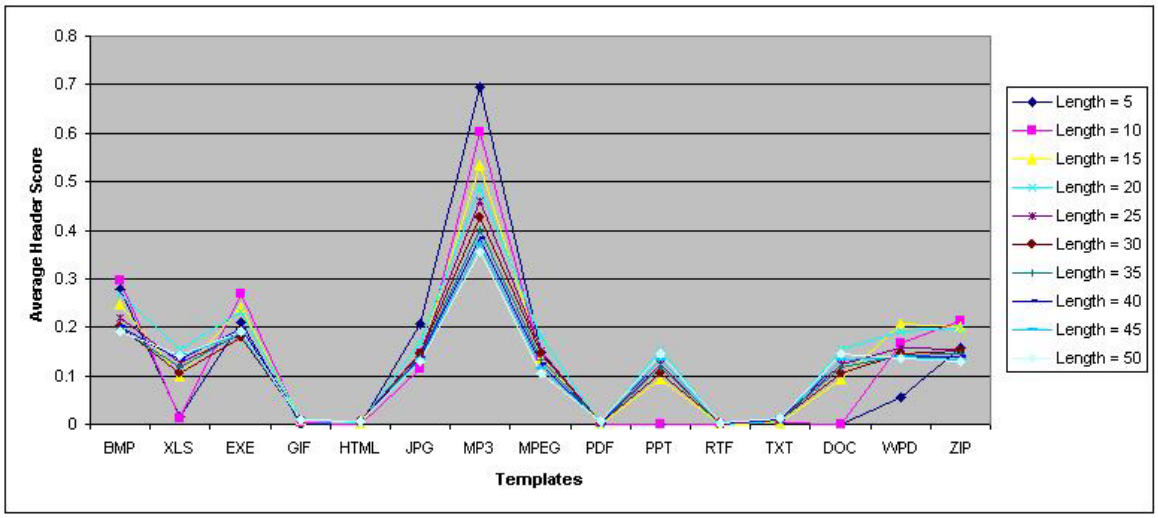

Figure 4-7 - Average header scores per file type for each header length

This graph shows a sharp spike in scores for the MP3 file format, which indicates a strongly characteristic file header. As was done in previous tests, the difference in scores between the highest and second-highest ranked file type can be plotted for each header length to determine the header length that provides the optimal differentiation between file types. Figure 4-8 shows this graph for the MP3 file format.

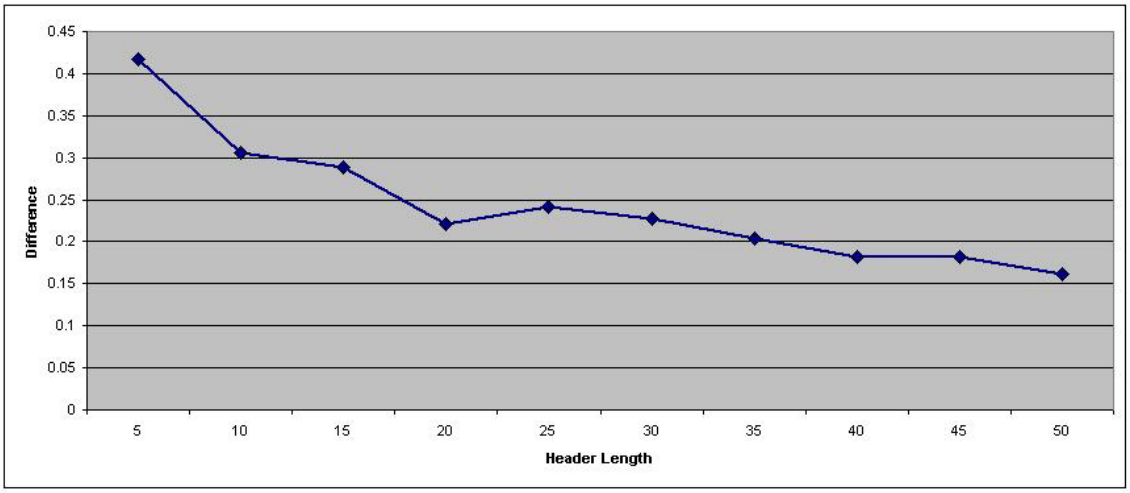

Figure 4-8 - Differences between the highest and second-highest header scores as a function of header length.

This graph suggests that the best differentiation occurs at a header length of 5 bytes. As the header length increases, the difference between the highest and second-highest ranked file types generally decreases. This suggests that the key portions of the MP3 file header occur within 5 to 10 bytes into the file. As the header length increases beyond five, the chances of random bytes matching the fingerprint increase. The more random matches there are, the less differentiation there is between file types. Therefore, as more bytes are analyzed in the MP3 file beyond those belonging to the file header, the performance of the Option 3 file header algorithm will decrease.

Figure 4-9 shows the average difference scores for each file type as a function of header length. A few of the file types, such as JPG and RTF, increase from a header length of 5 to a header length of 10. This indicates that these file types have characteristic file header byte patterns longer than five bytes, and less than 15 bytes. Despite the variations, though, most show the same overall trend. As the header length increases, the differentiation between file types decreases.

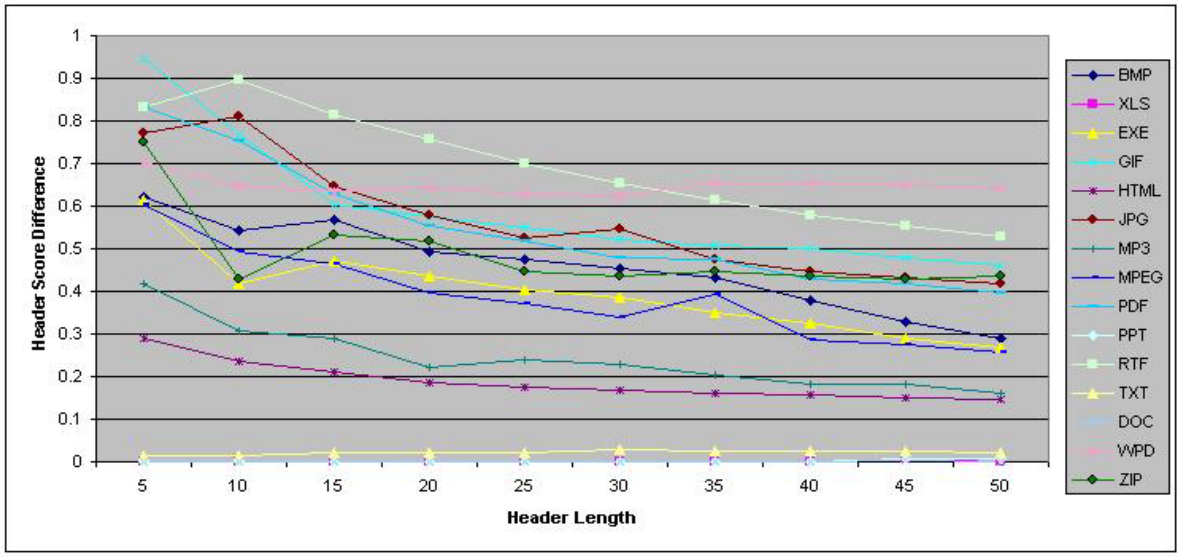

Figure 4-9 - Header score difference between the highest and second-highest file types as a function of header length.

The optimal header length is the value that results in the highest average level of differentiation across all file types. Figure 4-10 shows a graph of header score differences, averaged across all file types, as a function of header length.

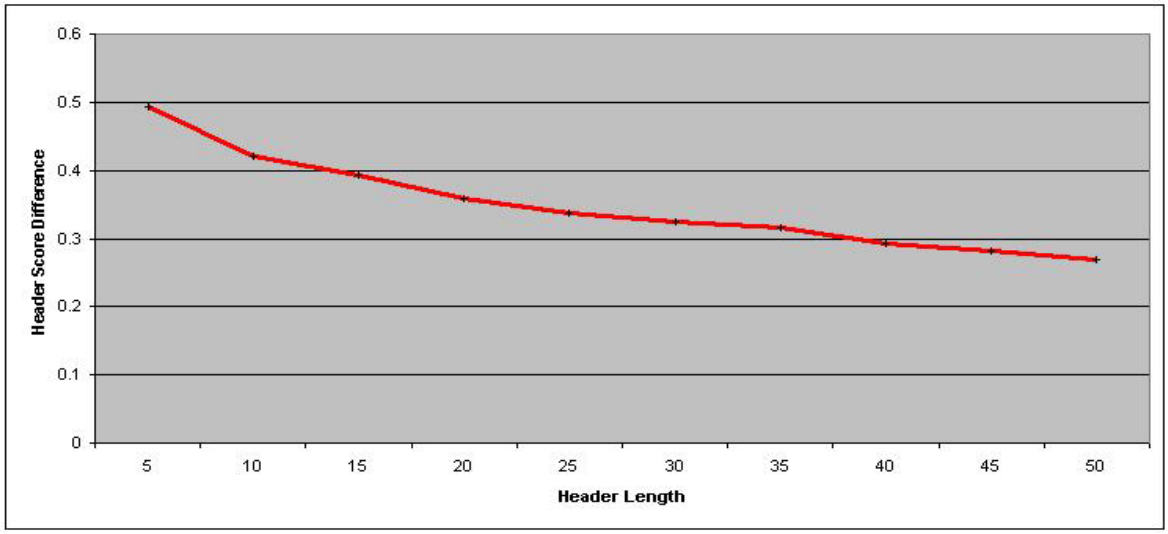

Figure 4-10 - Average header score differences as a function of header length.

Figure 4-10 confirms that of the lengths tested, a header length of 5 offers the best file recognition performance.

Test 5 – Trailer Length Sweep

As with header lengths, trailer lengths affect the scores of each file type differently, depending upon the type and length of their file headers. Therefore, all 60 test files were included in the trailer length sweep test. The trailer length was varied from 5 to 50 in linear steps of 5, and recognition reports were generated for each value.

A set of tables of the same format as Figure 4-6 was generated for each file type for each trailer length. In each table, the trailer effective scores were listed for each of the four test files per file type. These effective scores were then averaged together into a single effective score per file type for the given trailer length.

Another table was then built out of the average effective scores per fingerprint for each trailer length. Figure 4-11 shows a graph of the table produced for the MP3 file trailer.

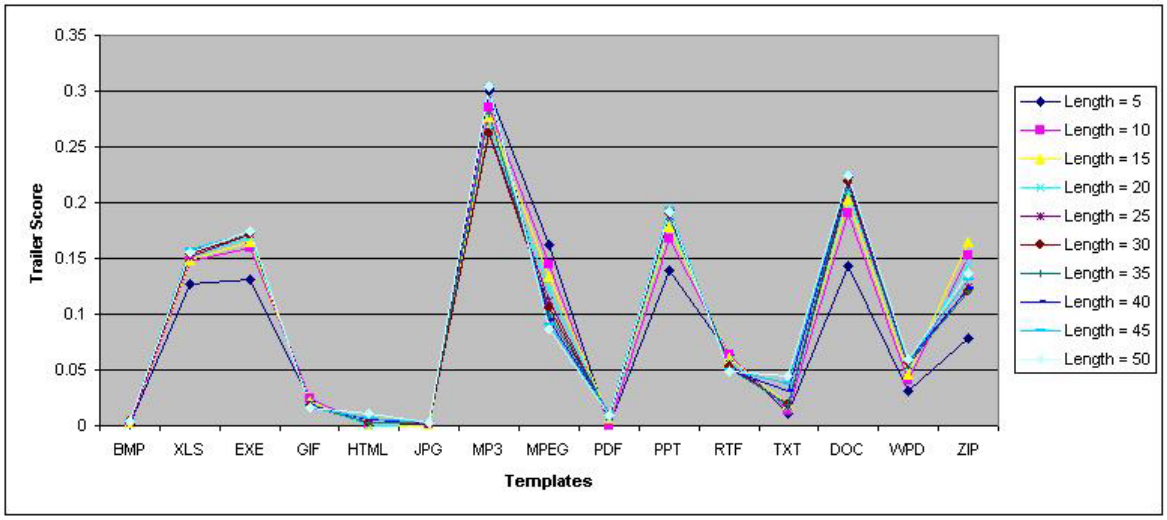

Figure 4-11 - Average trailer scores per file type for each header length

This graph shows a spike in scores for the MP3 file format, although it is much lower than the corresponding spike in Figure 4-7 for the MP3 header. Again, the difference in scores between the highest and second-highest ranked file type can be plotted for each trailer length to determine the trailer length that provides the optimal differentiation between file types. Figure 4-12 shows this graph for the MP3 file format.

Figure 4-12 - Differences between the highest and second-highest trailer scores as a function of header length.

This graph initially follows a pattern similar to that shown in Figure 4-8. The differentiation decreases as the trailer length increases. At a trailer length of 35, however, the differentiation begins to climb again. This indicates that for the MP3 file type, there is some data around 35 bytes from the end of the file that is characteristic of the MP3 file type. Even with these bytes, though, the score differences are still considerably below the difference offered by a trailer length of five bytes.

Figure 4-13 shows the average difference scores for each file type as a function of trailer length. File types with a strongly characteristic file trailer, such as JPG and PDF, follow a pattern very similar to the pattern observed in the file headers, for the same reasons. As the trailer length increases beyond the size of the actual file trailer, the chances of random bytes matching the fingerprint increases. The more random matches there are, the less differentiation there is between file types.

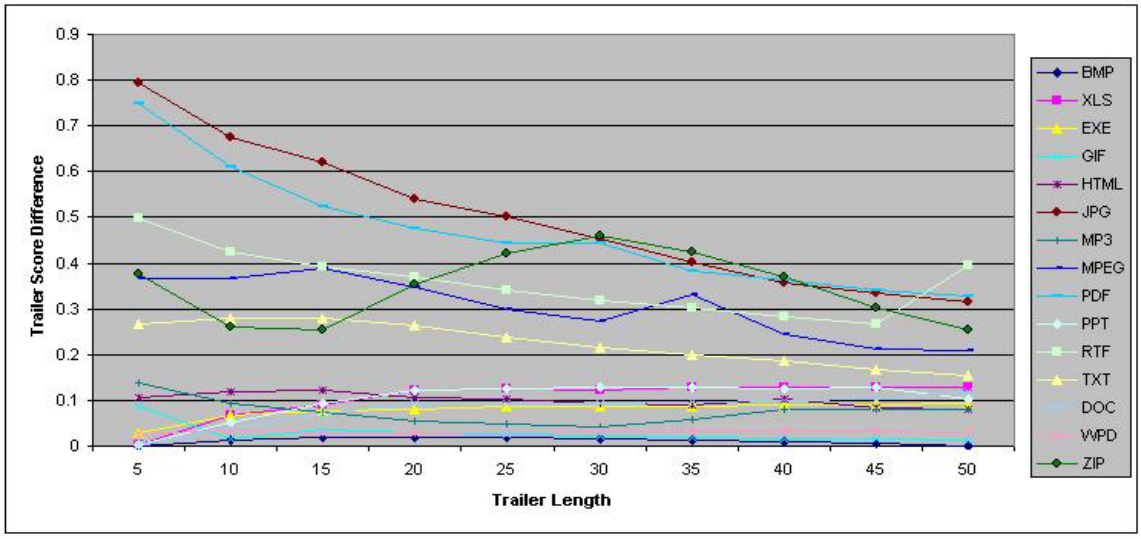

Figure 4-13 - Trailer score difference between the highest and second-highest file types as a function of trailer length.

File types that do not have a strongly characteristic file trailer, such as HTML and BMP, show consistently low scores across the range of trailer lengths.

As with header length, the optimal trailer length is the value that results in the highest average level of differentiation across all file types. Figure 4-14 shows a graph of trailer score differences, averaged across all file types, as a function of trailer length.

Figure 4-14 - Average trailer score differences as a function of trailer length.

Figure 4-14 confirms that of the lengths tested, a trailer length of 5 offers the best file recognition performance.

Test 6 – Accuracy Test (All Options)

Ten additional file types were added for the accuracy tests. See APPENDIX C for a list of file types included in this set of tests. A library of 100 test files was generated, representing four files of each file type. Recognition reports were then generated for each of the 100 files. Figure 4-15 shows the reported file type for each file.

	File 1	File 2	File 3	File 4	Score
AVI	AVI	WAV	AVI	AVI	3
BMP	BMP	BMP	BMP	BMP	4
DOC	DOC	DOC	DOC	DOC	4
EXE	EXE	EXE	EXE	EXE	4
FNT	FNT	FNT	FNT	RPM	3
GIF	GIF	GIF	GIF	GIF	4
GZ	GZ	GZ	GZ	GZ	4
HTML	HTML	HTML	HTML	HTML	4
JPG	JPG	JPG	JPG	JPG	4
MOV	MOV	MOV	MOV	MOV	4
MP3	RM	MP3	MP3	MP3	3
MPEG	MPEG	MPEG	MPEG	MPEG	4
PDF	PDF	PDF	PDF	PDF	4
PPT	DOC	DOC	DOC	DOC	0
PS	PS	PS	PS	PS	4
RTF	RTF	RTF	RTF	RTF	4
RM	RM	RM	RM	RM	4
RPM	RPM	RPM	RPM	RPM	4
TAR	TAR	TAR	TAR	TAR	4
TXT	TXT	TXT	TXT	TXT	4
TTF	TTF	TTF	TTF	TTF	4
WAV	WAV	WAV	WAV	AVI	3
WPD	WPD	WPD	WPD	WPD	4
XLS	DOC	PPT	DOC	DOC	0
ZIP	ZIP	ZIP	ZIP	ZIP	4
			TOTAL CORRECT:		88
			TOTAL FILES:		100
				Accuracy:	88.00%

Figure 4-15 - Identified type of each test file. The actual type is shown down the left column.

It correctly identified all of the files for 19 of the 25 file types. It identified 75% of the files correctly for four file types. Performance was very poor for two file types however, Microsoft PowerPoint presentations (PPT) and Microsoft Excel spreadsheets (XLS). It confused these file types with each other, and with Microsoft Word documents (DOC). Overall, the algorithm was 88 percent accurate in this test, correctly identifying 88 of the 100 input files.

It is interesting that it had difficulty distinguishing between the DOC, PPT, and XLS file types, since they are all part of the Microsoft Office suite, and the programs are designed to interoperate with each other. A comparison of the fingerprints for each of these three formats (see APPENDIX E) reveals strong similarities. The byte frequency distributions and byte frequency cross-correlation plots are similar between the three formats, and the header and trailer plots are identical.

A closer analysis of the files reveals a likely cause for their similarities. All three of the Microsoft Office formats are stored in an OLE compound document wrapper.[7,8] Since only five bytes of the file header and trailer are analyzed, all three of these file formats would appear identical to the Option 3 tests. Since the OLE compound document wrapper has a very characteristic file header,

To test this theory, a new OLE DOC fingerprint was generated from the 60 input files originally used to generate the three separate fingerprints (20 of each file type.) The accuracy test was then rerun using the new single OLE DOC fingerprint instead of the separate DOC, PPT, and XLS fingerprints. Figure 4-16 shows the results of this test.

Using the single OLE DOC file type fingerprint, all DOC, PPT, and XLS files were correctly identified, increasing the overall accuracy to 96 percent. This shows that all three file types can be accurately identified by a single fingerprint.

[7] Ken Kyler, Understanding OLE Documents, Delphi Developer's Journal, September 1998, available online from: http://www.kyler.com/pubs/ddj9894.html

[8] The Binary Structure of OLE Compound Documents, available online from: http://user.cs.tu-berlin.de/~schwartz/pmh/guide.html

	File 1	File 2	File 3	File 4	Score
AVI	AVI	WAV	AVI	AVI	3
BMP	BMP	BMP	BMP	BMP	4
DOC	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
EXE	EXE	EXE	EXE	EXE	4
FNT	FNT	FNT	FNT	RPM	3
GIF	GIF	GIF	GIF	GIF	4
GZ	GZ	GZ	GZ	GZ	4
HTML	HTML	HTML	HTML	HTML	4
JPG	JPG	JPG	JPG	JPG	4
MOV	MOV	MOV	MOV	MOV	4
MP3	RM	MP3	MP3	MP3	3
MPEG	MPEG	MPEG	MPEG	MPEG	4
PDF	PDF	PDF	PDF	PDF	4
PPT	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
PS	PS	PS	PS	PS	4
RTF	RTF	RTF	RTF	RTF	4
RM	RM	RM	RM	RM	4
RPM	RPM	RPM	RPM	RPM	4
TAR	TAR	TAR	TAR	TAR	4
TXT	TXT	TXT	TXT	TXT	4
TTF	TTF	TTF	TTF	TTF	4
WAV	WAV	WAV	WAV	AVI	3
WPD	WPD	WPD	WPD	WPD	4
XLS	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
ZIP	ZIP	ZIP	ZIP	ZIP	4
			TOTAL CORRECT:		96
			TOTAL FILES:		100
				Accuracy:	96.00%

Figure 4-16 - Identified type of each test file with a single OLE DOC fingerprint. The actual type is shown down the left column.

Test 7 – Accuracy Test (Option 1)

The accuracy tests described in Test 6 were rerun after Options 2 and 3 were turned off, so the recognition only considered the results of Option 1, the byte frequency analysis. Recognition reports were then generated for each of the 100 test files used in Test 6. Figure 4-17 shows the results of this test with separate fingerprints for DOC, PPT, and XLS files.

	File 1	File 2	File 3	File 4	Score
AVI	RTF	PDF	RM	TXT	0
BMP	FNT	XLS	RTF	TXT	0
DOC	RTF	DOC	DOC	RTF	2
EXE	PDF	DOC	DOC	XLS	0
FNT	RTF	TXT	TXT	GIF	0
GIF	RM	RM	ZIP	RM	0
GZ	RM	ZIP	TAR	MP3	0
HTML	TXT	TXT	TXT	RTF	0
JPG	MP3	MP3	GZ	JPG	1
MOV	MOV	RM	RM	RM	1
MP3	MP3	MP3	GZ	MP3	3
MPEG	PDF	TAR	MP3	PDF	0
PDF	PPT	TXT	PDF	PDF	2
PPT	DOC	DOC	DOC	RTF	0
PS	RTF	TXT	TXT	TXT	0
RTF	RTF	RTF	RTF	RTF	4
RM	PDF	MPEG	RM	RM	2
RPM	GZ	GZ	PDF	GZ	0
TAR	TXT	TXT	ZIP	PDF	0
TXT	TXT	TXT	TXT	TXT	4
TTF	XLS	DOC	TTF	TTF	2
WAV	FNT	TXT	TXT	TXT	0
WPD	WPD	TXT	TXT	WPD	2
XLS	DOC	FNT	DOC	XLS	1
ZIP	ZIP	ZIP	GIF	GIF	2
			TOTAL CORRECT:		26
			TOTAL FILES:		100
				Accuracy:	26.00%

Figure 4-17 - Identified type of each test file with only Option 1 enabled and separate fingerprints for DOC, PPT, and XLS. The actual type is shown down the left column.

The accuracy of Option 1 alone is only 26 percent. This is better than purely random guesses, which would provide an accuracy of 4 percent, but not accurate enough for practical use.

While there were errors across many file types, there was some confusion between the DOC, PPT, and XLS file types as there was in Test 7. Again, the three separate fingerprints were replaced with a single OLE DOC fingerprint, and the test was rerun. Figure 4-18 shows the resulting grid.

	File 1	File 2	File 3	File 4	Score
AVI	RTF	RM	RM	TXT	0
BMP	FNT	WPD	RTF	TXT	0
DOC	RTF	WPD	RTF	RTF	0
EXE	AVI	WPD	WPD	FNT	0
FNT	RTF	TXT	TXT	GIF	0
GIF	RM	RM	ZIP	RM	0
GZ	RM	ZIP	TAR	MP3	0
HTML	TXT	TXT	TXT	RTF	0
JPG	MP3	MP3	GZ	JPG	1
MOV	MOV	RM	RM	RM	1
MP3	MP3	MP3	GZ	MP3	3
MPEG	PDF	TAR	MP3	PDF	0
PDF	EXE	TXT	AVI	PDF	1
PPT	RTF	WPD	WPD	RTF	0
PS	RTF	TXT	TXT	TXT	0
RTF	RTF	RTF	RTF	RTF	4
RM	AVI	MPEG	RM	RM	2
RPM	GZ	GZ	PDF	GZ	0
TAR	TXT	TXT	ZIP	PDF	0
TXT	TXT	TXT	TXT	TXT	4
TTF	TTF	WPD	TTF	TTF	3
WAV	FNT	TXT	TXT	TXT	0
WPD	WPD	TXT	TXT	WPD	2
XLS	TXT	FNT	TXT	WPD	0
ZIP	ZIP	ZIP	GIF	GIF	2
			TOTAL CORRECT:		23
			TOTAL FILES:		100
				Accuracy:	23.00%

Figure 4-18 - Identified type of each test file with only Option 1 enabled and a single OLE DOC fingerprint. The actual type is shown down the left column.

With a combined OLE DOC fingerprint, the accuracy for Option 1 actually decreased to 23 percent. The decrease in accuracy occurred because the OLE DOC fingerprint was not selected at all, even for the DOC, PPT, and XLS file types. Where some of these had been correctly identified with the separate fingerprints, none of them were identified as OLE DOC with the single fingerprint.

Comparing the fingerprints for the DOC, PPT, and XLS types (see APPENDIX E) shows that while the Option 3 file headers and trailers are identical between the three types, there are differences in both Option 1 and Option 2 results. Combining the file types into a single fingerprint resulted in an Option 1 fingerprint that was an average of

the three types. This average was different enough from the three original types to prevent accurate detection by using Option 1.

Test 8 – Accuracy Test (Option 2)

For Test 8, Options 1 and 3 were turned off, so the recognition only considered the results of Option 2, the byte frequency analysis. Note that the frequency data produced by Option 1 is required to perform the byte frequency cross-correlations for Option 2. Therefore the processing of Option 1 must still occur, however the results for Option 1 were ignored when performing recognition calculations.

Recognition reports were then generated for each of the same 100 test files that were used in the previous accuracy tests. Figure 4-19 shows the results of this test with separate fingerprints for DOC, PPT, and XLS files.

The accuracy of Option 2 alone is 41 percent. This is a significant improvement over Option 1, but still not accurate enough for practical use.

There was again some confusion between the DOC, PPT, and XLS file types, although much less than in previous accuracy tests. Option 2 was able to successfully identify all DOC files, three out of four PPT files, and two out of four XLS files.

These results were again compared to those produced by combining these three types into a single OLE DOC fingerprint. Figure 4-20 shows the results of rerunning this test using the single fingerprint.

	File 1	File 2	File 3	File 4	Score
AVI	DOC	DOC	XLS	XLS	0
BMP	TTF	PPT	PPT	DOC	0
DOC	DOC	DOC	DOC	DOC	4
EXE	PPT	DOC	DOC	EXE	1
FNT	PPT	DOC	DOC	DOC	0
GIF	XLS	GIF	GIF	GIF	3
GZ	RPM	DOC	GZ	GZ	2
HTML	TXT	TXT	TXT	RTF	0
JPG	DOC	DOC	DOC	DOC	0
MOV	GIF	MOV	GIF	XLS	1
MP3	DOC	MP3	MP3	MP3	3
MPEG	MPEG	MPEG	DOC	DOC	2
PDF	PDF	TXT	PDF	PDF	3
PPT	PPT	DOC	PPT	PPT	3
PS	TXT	TXT	TXT	TXT	0
RTF	RTF	TXT	TXT	RTF	2
RM	RM	RM	RM	PPT	3
RPM	RPM	RPM	DOC	RPM	3
TAR	TXT	TXT	RPM	XLS	0
TXT	TXT	TXT	TXT	TXT	4
TTF	DOC	DOC	TTF	TTF	2
WAV	TXT	TXT	TXT	TXT	0
WPD	WPD	DOC	WPD	WPD	3
XLS	XLS	XLS	PPT	DOC	2
ZIP	DOC	DOC	DOC	DOC	0
			TOTAL CORRECT:		41
			TOTAL FILES:		100
				Accuracy:	41.00%

Figure 4-19 - Identified type of each test file with only Option 2 enabled and separate fingerprints for DOC, PPT, and XLS file types. The actual type is shown down the left column.

As with the Option 1 Only test, performance actually decreased with a single OLE DOC fingerprint, although for a different reason. Figure 4-20 shows that all DOC, PPT, and XLS files were correctly identified as OLE DOC files. However, many other file types were incorrectly identified as OLE DOC, including some that had been correctly identified when separate fingerprints were used.

	File 1	File 2	File 3	File 4	Score
AVI	OLE DOC	OLE DOC	OLE DOC	OLE DOC	0
BMP	TTF	OLE DOC	OLE DOC	OLE DOC	0
DOC	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
EXE	OLE DOC	OLE DOC	OLE DOC	OLE DOC	0
FNT	OLE DOC	OLE DOC	OLE DOC	OLE DOC	0
GIF	OLE DOC	GIF	GIF	GIF	3
GZ	RPM	OLE DOC	OLE DOC	OLE DOC	0
HTML	TXT	TXT	TXT	RTF	0
JPG	OLE DOC	OLE DOC	OLE DOC	OLE DOC	0
MOV	GIF	MOV	GIF	OLE DOC	1
MP3	OLE DOC	MP3	MP3	MP3	3
MPEG	MPEG	OLE DOC	OLE DOC	OLE DOC	1
PDF	RTF	TXT	TXT	RTF	0
PPT	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
PS	TXT	TXT	TXT	TXT	0
RTF	RTF	TXT	TXT	RTF	2
RM	RM	RM	RM	OLE DOC	3
RPM	RPM	RPM	OLE DOC	RPM	3
TAR	TXT	TXT	RPM	OLE DOC	0
TXT	TXT	TXT	TXT	TXT	4
TTF	OLE DOC	OLE DOC	TTF	TTF	2
WAV	TXT	TXT	TXT	TXT	0
WPD	WPD	TXT	WPD	WPD	3
XLS	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
ZIP	OLE DOC	OLE DOC	OLE DOC	OLE DOC	0
			TOTAL CORRECT:		37
			TOTAL FILES:		100
				Accuracy:	37.00%

Figure 4-20 - Identified type of each test file with only Option 2 enabled and a single OLE DOC fingerprint. The actual type is shown down the left column.

When the Option 2 results from the three file types were averaged into a single fingerprint, it produced a new Option 2 graph that was similar enough to each of the three input types to allow for proper recognition of those types. However, the new average that was produced was closer to several other file types that the three separate fingerprints had been. This resulted in the overall decrease in performance.

Test 9 – Accuracy Test (Option 3)

For Test 9, Options 1 and 2 were turned off, so the recognition only considered the results of Option 3, the file header and trailer analysis. Recognition reports were generated for each of the same 100 test files that were used in the previous accuracy tests.

Figure 4-21 shows the results of this test with separate fingerprints for DOC, PPT, and XLS files.

	File 1	File 2	File 3	File 4	Score
AVI	AVI	AVI	AVI	AVI	4
BMP	BMP	BMP	BMP	BMP	4
DOC	DOC	PPT	DOC	DOC	3
EXE	EXE	EXE	EXE	EXE	4
FNT	FNT	FNT	FNT	RPM	3
GIF	GIF	GIF	GIF	GIF	4
GZ	GZ	GZ	GZ	GZ	4
HTML	HTML	HTML	HTML	HTML	4
JPG	JPG	JPG	JPG	JPG	4
MOV	MOV	MOV	MOV	MOV	4
MP3	RM	MP3	MP3	MP3	3
MPEG	MPEG	MPEG	MPEG	MPEG	4
PDF	PDF	PDF	PDF	PDF	4
PPT	DOC	DOC	DOC	DOC	0
PS	PS	PS	PS	PS	4
RTF	RTF	RTF	RTF	RTF	4
RM	RM	RM	RM	RM	4
RPM	RPM	RPM	RPM	RPM	4
TAR	TAR	TAR	TAR	TAR	4
TXT	TXT	TXT	TXT	TXT	4
TTF	TTF	TTF	TTF	TTF	4
WAV	WAV	AVI	WAV	AVI	2
WPD	WPD	WPD	WPD	WPD	4
XLS	DOC	DOC	DOC	DOC	0
ZIP	ZIP	ZIP	ZIP	ZIP	4
			TOTAL CORRECT:		87
			TOTAL FILES:		100
				Accuracy:	87.00%

Figure 4-21 - Identified type of each test file with only Option 3 enabled and separate fingerprints for DOC, PPT, and XLS file types. The actual type is shown down the left column.

The accuracy of Option 3 alone is 87 percent. This is another significant improvement over Option 2, and may be accurate enough for some fault-tolerant applications.

Most of the errors occurred between the DOC, PPT, and XLS file types. Figure 4-22 shows the results of rerunning this test using the single OLE DOC fingerprint.

	File 1	File 2	File 3	File 4	Score
AVI	AVI	AVI	AVI	AVI	4
BMP	BMP	BMP	BMP	BMP	4
DOC	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
EXE	EXE	EXE	EXE	EXE	4
FNT	FNT	FNT	FNT	RPM	3
GIF	GIF	GIF	GIF	GIF	4
GZ	GZ	GZ	GZ	GZ	4
HTML	HTML	HTML	HTML	HTML	4
JPG	JPG	JPG	JPG	JPG	4
MOV	MOV	MOV	MOV	MOV	4
MP3	RM	MP3	MP3	MP3	3
MPEG	MPEG	MPEG	MPEG	MPEG	4
PDF	PDF	PDF	PDF	PDF	4
PPT	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
PS	PS	PS	PS	PS	4
RTF	RTF	RTF	RTF	RTF	4
RM	RM	RM	RM	RM	4
RPM	RPM	RPM	RPM	RPM	4
TAR	TAR	TAR	TAR	TAR	4
TXT	TXT	TXT	TXT	TXT	4
TTF	TTF	TTF	TTF	TTF	4
WAV	WAV	AVI	WAV	AVI	2
WPD	WPD	WPD	WPD	WPD	4
XLS	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
ZIP	ZIP	ZIP	ZIP	ZIP	4
			TOTAL CORRECT:		96
			TOTAL FILES:		100
				Accuracy:	96.00%

Figure 4-22 - Identified type of each test file with only Option 3 enabled and a single OLE DOC fingerprint. The actual type is shown down the left column.

This time, the combined OLE DOC fingerprint resulted in the correct identification of all three component file types, without affecting the identification of any other file types. As a result, the accuracy of Option 3 increased to 96 percent with a single OLE DOC fingerprint.

Test 10 – Extended Accuracy Test (All Options)

Five additional file types were added for the extended accuracy tests. (See APPENDIX D.) A library of 120 test files was generated, representing four files of each file type. Recognition reports were then generated for each of the 120 files. Figure 4-23 shows the reported file type for each file.

	File 1	File 2	File 3	File 4	Score
3TF	3TF	3TF	3TF	3TF	4
ACD	ACD	ACD	ACD	ACD	4
AVI	AVI	AVI	WAV	AVI	3
BMP	BMP	BMP	BMP	BMP	4
CAT	CAT	CAT	CAT	CAT	4
CRP	CRP	CRP	CRP	CRP	4
DOC	ACD	DOC	DOC	ACD	2
EXE	EXE	EXE	EXE	EXE	4
FNT	RPM	FNT	FNT	FNT	3
GIF	GIF	GIF	GIF	GIF	4
GZ	GZ	GZ	GZ	GZ	4
HTML	HTML	HTML	HTML	HTML	4
JPG	JPG	JPG	JPG	JPG	4
MDL	MDL	CAT	MDL	MDL	3
MOV	MOV	MOV	MOV	MOV	4
MP3	RM	MP3	MP3	MP3	3
MPEG	MPEG	MPEG	MPEG	MPEG	4
PDF	PDF	PDF	PDF	PDF	4
PPT	ACD	DOC	DOC	ACD	0
PS	PS	PS	PS	PS	4
RTF	RTF	RTF	RTF	RTF	4
RM	RM	RM	RM	RM	4
RPM	RPM	RPM	RPM	RPM	4
TAR	TAR	TAR	TAR	TAR	4
TXT	TXT	TXT	TXT	TXT	4
TTF	TTF	TTF	TTF	TTF	4
WAV	AVI	WAV	WAV	WAV	3
WPD	WPD	WPD	WPD	WPD	4
XLS	DOC	ACD	PPT	ACD	0
ZIP	ZIP	ZIP	ZIP	ZIP	4
			TOTAL CORRECT:		105
			TOTAL FILES:		120
				Accuracy:	87.50%

Figure 4-23 Identified type of each test file with additional types added and all Options.

Again there was poor performance for the DOC, PPT, and XLS file formats. Rerunning this test with a combined OLE DOC fingerprint produced the results shown in Figure 4-24.

	File 1	File 2	File 3	File 4	Score
3TF	3TF	3TF	3TF	3TF	4
ACD	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
AVI	AVI	WAV	AVI	AVI	3
BMP	BMP	BMP	BMP	BMP	4
CAT	CAT	CAT	CAT	CAT	4
CRP	CRP	CRP	CRP	CRP	4
DOC	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
EXE	EXE	EXE	EXE	EXE	4
FNT	FNT	FNT	FNT	RPM	3
GIF	GIF	GIF	GIF	GIF	4
GZ	GZ	GZ	GZ	GZ	4
HTML	HTML	HTML	HTML	HTML	4
JPG	JPG	JPG	JPG	JPG	4
MDL	MDL	CAT	MDL	MDL	3
MOV	MOV	MOV	MOV	MOV	4
MP3	RM	MP3	MP3	MP3	3
MPEG	MPEG	MPEG	MPEG	MPEG	4
PDF	PDF	PDF	PDF	PDF	4
PPT	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
PS	PS	PS	PS	PS	4
RTF	RTF	RTF	RTF	RTF	4
RM	RM	RM	RM	RM	4
RPM	RPM	RPM	RPM	RPM	4
TAR	TAR	TAR	TAR	TAR	4
TXT	TXT	TXT	TXT	TXT	4
TTF	TTF	TTF	TTF	TTF	4
WAV	AVI	WAV	WAV	WAV	3
WPD	WPD	WPD	WPD	WPD	4
XLS	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
ZIP	ZIP	ZIP	ZIP	ZIP	4
			TOTAL CORRECT:		115
			TOTAL FILES:		120
				Accuracy:	95.83%

Figure 4-24 Identified type of each test file with additional types added, a combined OLE DOC fingerprint, and all Options enabled.

The ACD files were all identified as OLE DOC files. Closer inspection of the ACD files revealed that they were in fact stored as OLE compound documents. This identification was therefore correct.

Test 11 – Extended Accuracy Test (Option 1)

The accuracy tests described in Test 10 were rerun after Options 2 and 3 were turned off, so the recognition only considered the results of Option 1, the byte frequency analysis. Recognition reports were then generated for each of the 120 test files used in Test 10. Figure 4-25 shows the results of this test with separate fingerprints for ACD, DOC, PPT, and XLS files.

	File 1	File 2	File 3	File 4	Score
3TF	3TF	3TF	3TF	3TF	4
ACD	3TF	ACD	3TF	ACD	2
AVI	3TF	RM	CRP	3TF	0
BMP	3TF	3TF	3TF	FNT	0
CAT	CAT	CAT	CAT	CAT	4
CRP	CRP	CRP	CRP	CRP	4
DOC	3TF	3TF	DOC	3TF	1
EXE	XLS	DOC	DOC	CRP	1
FNT	GIF	3TF	3TF	3TF	0
GIF	RM	ZIP	RM	RM	0
GZ	MP3	TAR	ZIP	CRP	0
HTML	RTF	TXT	CAT	CAT	0
JPG	JPG	GZ	MP3	MP3	1
MDL	CAT	CAT	CAT	CAT	0
MOV	RM	CRP	RM	CRP	0
MP3	MP3	MP3	GZ	MP3	3
MPEG	CRP	MP3	CRP	CRP	0
PDF	CRP	PDF	TXT	PPT	1
PPT	3TF	3TF	DOC	3TF	0
PS	TXT	TXT	TXT	CAT	0
RTF	RTF	RTF	CAT	RTF	3
RM	RM	RM	CRP	CRP	2
RPM	CRP	GZ	GZ	GZ	0
TAR	CRP	ZIP	TXT	CAT	0
TXT	TXT	TXT	CAT	TXT	3
TTF	TTF	TTF	DOC	XLS	2
WAV	CAT	TXT	3TF	FNT	0
WPD	3TF	3TF	TXT	WPD	1
XLS	XLS	3TF	FNT	3TF	1
ZIP	GIF	GIF	ZIP	ZIP	2
			TOTAL CORRECT:		35
			TOTAL FILES:		120
				Accuracy:	29.17%

Figure 4-25 Identified type of each test file with additional types added and only Option 1.

Rerunning this test with a combined OLE DOC fingerprint produced the results shown in Figure 4-26.

	File 1	File 2	File 3	File 4	Score
3TF	3TF	3TF	3TF	3TF	4
ACD	3TF	3TF	OLE DOC	OLE DOC	2
AVI	3TF	CRP	RM	3TF	0
BMP	3TF	3TF	FNT	3TF	0
CAT	CAT	CAT	CAT	CAT	4
CRP	CRP	CRP	CRP	CRP	4
DOC	WPD	3TF	3TF	3TF	0
EXE	FNT	3TF	3TF	CRP	0
FNT	3TF	3TF	3TF	GIF	0
GIF	RM	ZIP	RM	RM	0
GZ	MP3	TAR	ZIP	CRP	0
HTML	RTF	TXT	CAT	CAT	0
JPG	JPG	GZ	MP3	MP3	1
MDL	CAT	CAT	CAT	CAT	0
MOV	CRP	CRP	RM	RM	0
MP3	MP3	GZ	MP3	MP3	3
MPEG	CRP	CRP	MP3	CRP	0
PDF	CRP	PDF	EXE	TXT	1
PPT	3TF	3TF	3TF	3TF	0
PS	TXT	TXT	CAT	TXT	0
RTF	RTF	RTF	RTF	CAT	3
RM	RM	CRP	RM	CRP	2
RPM	GZ	CRP	GZ	GZ	0
TAR	CRP	CAT	TXT	ZIP	0
TXT	TXT	CAT	TXT	TXT	3
TTF	TTF	TTF	TTF	WPD	3
WAV	CAT	TXT	FNT	3TF	0
WPD	3TF	3TF	WPD	TXT	1
XLS	WPD	FNT	3TF	3TF	0
ZIP	GIF	ZIP	ZIP	GIF	2
			TOTAL CORRECT:		33
			TOTAL FILES:		120
				Accuracy:	27.50%

Figure 4-26 Identified type of each test file with additional types added, a combined OLE DOC fingerprint, and only Option 1 enabled.

As was the case in Test 7, accuracy decreased with the combined OLE DOC fingerprint, for the same reason. The combined frequency distribution in the OLE DOC

fingerprint was different enough from the separate file formats to prevent it from being selected except for two ACD files, even in instances where files had previously been correctly identified with separate types.

The accuracies of the results with separate and combined fingerprints are slightly higher, but comparable to, the accuracies obtained in the initial set of accuracy tests. The slight increase in accuracy came from the fact that several of the new file formats were able to be accurately identified by their frequency distributions.

Test 12 – Extended Accuracy Test (Option 2)

For Test 12, Options 1 and 3 were turned off, so the recognition only considered the results of Option 2, the byte frequency analysis. As was noted in Test 8, the frequency data produced by Option 1 is required to perform the byte frequency cross-correlations for Option 2. Therefore the processing of Option 1 must still occur, however the results for Option 1 were ignored when performing recognition calculations.

Recognition reports were then generated for each of the same 120 test files that were used in the previous accuracy tests. Figure 4-27 shows the results of this test with separate fingerprints for ACD, DOC, PPT, and XLS files.

The accuracy of Option 2 alone in this test is 44 percent. This is a significant improvement over Option 1, but still not accurate enough for practical use. It is slightly higher than the accuracy obtained in the initial accuracy tests, although comparable.

There was again some confusion between the ACD, DOC, PPT, and XLS file types, although much less than in previous accuracy tests. Option 2 was able to successfully identify all DOC files, two out of four PPT files, and one out of four XLS files.

	File 1	File 2	File 3	File 4	Score
3TF	3TF	3TF	3TF	3TF	4
ACD	ACD	ACD	ACD	ACD	4
AVI	XLS	XLS	CAT	3TF	0
BMP	3TF	3TF	3TF	TTF	0
CAT	CAT	CAT	CAT	CAT	4
CRP	CRP	CRP	CRP	CRP	4
DOC	DOC	DOC	DOC	DOC	4
EXE	EXE	DOC	DOC	ACD	1
FNT	3TF	3TF	3TF	3TF	0
GIF	GIF	GIF	3TF	3TF	2
GZ	3TF	3TF	3TF	3TF	0
HTML	RTF	TXT	CAT	CAT	0
JPG	3TF	3TF	3TF	3TF	0
MDL	MDL	CAT	MDL	MDL	3
MOV	3TF	GIF	3TF	GIF	1
MP3	3TF	MP3	MP3	MP3	3
MPEG	3TF	3TF	3TF	MPEG	1
PDF	PDF	PDF	TXT	PDF	3
PPT	3TF	PPT	DOC	PPT	2
PS	TXT	TXT	TXT	CAT	0
RTF	RTF	TXT	TXT	RTF	2
RM	3TF	RM	RM	RM	3
RPM	DOC	RPM	RPM	RPM	3
TAR	XLS	RPM	3TF	CAT	0
TXT	TXT	TXT	CAT	TXT	3
TTF	TTF	TTF	DOC	DOC	2
WAV	TXT	TXT	TXT	TXT	0
WPD	WPD	WPD	DOC	WPD	3
XLS	3TF	3TF	ACD	XLS	1
ZIP	3TF	3TF	3TF	3TF	0
			TOTAL CORRECT:		53
			TOTAL FILES:		120
				Accuracy:	44.17%

Figure 4-27 Identified type of each test file with additional types added and only Option 2.

These results were again compared to those produced by combining these three types into a single OLE DOC fingerprint. Figure 4-28 shows the results of rerunning this test using the single fingerprint.

	File 1	File 2	File 3	File 4	Score
3TF	3TF	3TF	3TF	3TF	4
ACD	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
AVI	OLE DOC	CAT	OLE DOC	3TF	0
BMP	3TF	3TF	TTF	3TF	0
CAT	CAT	CAT	CAT	CAT	4
CRP	CRP	CRP	CRP	CRP	4
DOC	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
EXE	OLE DOC	OLE DOC	OLE DOC	OLE DOC	0
FNT	3TF	3TF	3TF	3TF	0
GIF	GIF	GIF	3TF	3TF	2
GZ	3TF	3TF	3TF	3TF	0
HTML	RTF	TXT	CAT	CAT	0
JPG	3TF	3TF	3TF	3TF	0
MDL	MDL	CAT	MDL	MDL	3
MOV	GIF	GIF	3TF	3TF	0
MP3	3TF	MP3	MP3	MP3	3
MPEG	3TF	MPEG	3TF	OLE DOC	1
PDF	PDF	PDF	PDF	TXT	3
PPT	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
PS	TXT	TXT	CAT	TXT	0
RTF	RTF	TXT	RTF	TXT	2
RM	OLE DOC	RM	RM	RM	3
RPM	RPM	OLE DOC	RPM	RPM	3
TAR	OLE DOC	CAT	3TF	RPM	0
TXT	TXT	CAT	TXT	TXT	3
TTF	TTF	TTF	OLE DOC	OLE DOC	2
WAV	TXT	TXT	TXT	TXT	0
WPD	WPD	WPD	WPD	TXT	3
XLS	3TF	OLE DOC	OLE DOC	OLE DOC	3
ZIP	3TF	3TF	3TF	3TF	0
			TOTAL CORRECT:		55
			TOTAL FILES:		120
				Accuracy:	45.83%

Figure 4-28 Identified type of each test file with additional types added, a combined OLE DOC fingerprint, and only Option 2 enabled.

Unlike Test 8, performance increased with a single OLE DOC fingerprint. Some files that had been correctly identified with separate fingerprints were incorrectly identified by the combined fingerprint. However there were enough files that had been previously misidentified that were correctly identified by the combined fingerprint to result in an overall increase in performance.

Test 13 – Extended Accuracy Test (Option 3)

For Test 13, Options 1 and 2 were turned off, so the recognition only considered the results of Option 3, the file header and trailer analysis. Recognition reports were generated for each of the same 120 test files that were used in the previous accuracy tests. Figure 4-29 shows the results of this test with separate fingerprints for ACD, DOC, PPT, and XLS files.

The accuracy of Option 3 alone is 85 percent. This is another significant improvement over Option 2, and may be accurate enough for some fault-tolerant applications. It is comparable to the results obtained in Test 9, although slightly lower because of added confusion between DOC and ACD formats.

	File 1	File 2	File 3	File 4	Score
3TF	3TF	3TF	3TF	3TF	4
ACD	ACD	ACD	ACD	ACD	4
AVI	AVI	AVI	AVI	AVI	4
BMP	BMP	BMP	BMP	BMP	4
CAT	CAT	CAT	CAT	CAT	4
CRP	CRP	CRP	CRP	CRP	4
DOC	ACD	ACD	PPT	ACD	0
EXE	EXE	EXE	EXE	EXE	4
FNT	RPM	FNT	FNT	FNT	3
GIF	GIF	GIF	GIF	GIF	4
GZ	GZ	GZ	GZ	GZ	4
HTML	HTML	HTML	HTML	HTML	4
JPG	JPG	JPG	HTML	JPG	3
MDL	MDL	CAT	MDL	MDL	3
MOV	MOV	MOV	MOV	MOV	4
MP3	RM	MP3	MP3	MP3	3
MPEG	MPEG	MPEG	MPEG	MPEG	4
PDF	PDF	PDF	PDF	PDF	4
PPT	ACD	ACD	ACD	ACD	0
PS	PS	PS	PS	PS	4
RTF	RTF	RTF	RTF	RTF	4
RM	RM	RM	RM	RM	4
RPM	RPM	RPM	RPM	RPM	4
TAR	TAR	TAR	TAR	TAR	4
TXT	TXT	TXT	TXT	TXT	4
TTF	TTF	TTF	TTF	TTF	4
WAV	AVI	WAV	AVI	WAV	2
WPD	WPD	WPD	WPD	WPD	4
XLS	ACD	ACD	ACD	ACD	0
ZIP	ZIP	ZIP	ZIP	ZIP	4
			TOTAL CORRECT:		102
			TOTAL FILES:		120
				Accuracy:	85.00%

Figure 4-29 Identified type of each test file with additional types added and only Option 3.

Again, most of the errors occurred between the ACD, DOC, PPT, and XLS file types. Figure 4-30 shows the results of rerunning this test using a combined fingerprint.

	File 1	File 2	File 3	File 4	Score
3TF	3TF	3TF	3TF	3TF	4
ACD	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
AVI	AVI	AVI	AVI	AVI	4
BMP	BMP	BMP	BMP	BMP	4
CAT	CAT	CAT	CAT	CAT	4
CRP	CRP	CRP	CRP	CRP	4
DOC	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
EXE	EXE	EXE	EXE	EXE	4
FNT	FNT	FNT	FNT	RPM	3
GIF	GIF	GIF	GIF	GIF	4
GZ	GZ	GZ	GZ	GZ	4
HTML	HTML	HTML	HTML	HTML	4
JPG	JPG	JPG	JPG	JPG	4
MDL	MDL	CAT	MDL	MDL	3
MOV	MOV	MOV	MOV	MOV	4
MP3	RM	MP3	MP3	MP3	3
MPEG	MPEG	MPEG	MPEG	MPEG	4
PDF	PDF	PDF	PDF	PDF	4
PPT	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
PS	PS	PS	PS	PS	4
RTF	RTF	RTF	RTF	RTF	4
RM	RM	RM	RM	RM	4
RPM	RPM	RPM	RPM	RPM	4
TAR	TAR	TAR	TAR	TAR	4
TXT	TXT	TXT	TXT	TXT	4
TTF	TTF	TTF	TTF	TTF	4
WAV	AVI	WAV	WAV	AVI	2
WPD	WPD	WPD	WPD	WPD	4
XLS	OLE DOC	OLE DOC	OLE DOC	OLE DOC	4
ZIP	ZIP	ZIP	ZIP	ZIP	4
			TOTAL CORRECT:		115
			TOTAL FILES:		120
				Accuracy:	95.83%

Figure 4-30 Identified type of each test file with additional types added, a combined OLE DOC fingerprint, and only Option 3 enabled.

This time, the combined OLE DOC fingerprint resulted in the correct identification of all three component file types, without affecting the identification of any other file types. As a result, the accuracy of Option 3 increased to 95.8 percent with a single OLE DOC fingerprint. This is almost identical to the 96 percent accuracy shown in Test 9.

CHAPTER 5: ANALYSIS

Conclusions and Future Work

Six key design goals were identified for the file type recognition algorithm. Three of the goals were accuracy, speed, and flexibility. These three factors proved to be closely related. In addition, speed is directly proportional to the number of fingerprints that have been created (although it is independent of the size of files that were originally added into the fingerprint.)

Flexibility was provided by allowing for three options that could be used independently or in combinations. The options provided different methods of file comparison with different accuracies and speed of execution.

The first option, byte frequency analysis proved to be the fastest. An unknown file takes an average of 0.010 seconds to compare to 25 fingerprints and identify the closest match.[9] The accuracy of this option was the worst of the three, though, at only 28% for a combined OLD DOC fingerprint, and 29% for separate ACD, DOC, PPT, and XLS fingerprints. (All accuracies given will be those produced by the extended accuracy tests, since those tests included the broadest range of file types.)

By itself, Option 1 would be of very limited use. The calculations performed in Option 1, though, are used as the basis for the second Option, byte frequency cross-correlation. Option 2 proved to be by far the slowest, and only moderately more accurate than the first option. An unknown file takes an average of 1.19 seconds to compare to 25 fingerprints and identify the closest match. Option 2 offers slightly improved accuracy of 46% for a combined OLE DOC fingerprint and 44% for separate ACD, DOC, PPT, and XLS fingerprints. This accuracy is still too low to be of practical use in most applications.

Option 3 provides the best combination of speed and accuracy. An unknown file takes an average of 0.015 seconds to compare to 25 fingerprints and identify the closest match, which is almost as fast as Option 1. Option 3 had by far the highest accuracy at 96% for a combined OLE DOC fingerprint and 85% for separate ACD, DOC, PPT, and XLS fingerprints.

All three options together take an average of 1.202 seconds to compare an unknown file to 25 fingerprints and identify the closest match. The accuracy produced by all three options was essentially the same as Option 3 alone at 96% for a combined OLE DOC fingerprint and 88% for separate ACD, DOC, PPT, and XLS fingerprints.

[9] All times were taken on an 800 MHz Pentium III laptop with 512 MB RAM.

Since the speed is proportional to the number of filters, it would be useful to compare the times for each of the options to perform a comparison against a single fingerprint. These times could then be easily scaled to any number of fingerprints.

Figure 5-1 shows a summary of the times it takes each option to compare an unknown file to one fingerprint, as well as the accuracies of each option with a combined OLE DOC fingerprint and with separate DOC, PPT, and XLS fingerprints.

	Time per Fingerprint Comparison	Accuracy With Combined OLE DOC	Accuracy With Separate ACD, DOC, PPT, XLS
Option 1	0.0004 sec.	28 %	29 %
Option 2	0.0476 sec.	46 %	44 %
Option 3	0.0006 sec.	96 %	85 %
All Options	0.0481 sec.	96 %	88 %

Figure 5-1 – Summary of the times it takes each option to compare an unknown file to one fingerprint and the option's accuracies for a single OLE DOC fingerprint and for separate DOC, PPT, and XLS fingerprints

Another factor related to the speed of the algorithm is the length of time it takes to build fingerprints from a library of known files. Generating 25 file type fingerprints from 500 files, totaling 233 MB, took 66 seconds. This averages to 2.64 seconds per fingerprint with 9.32 MB of data in 20 files per fingerprint.

Considering that generation of the fingerprints is a one-time operation, this is acceptable performance. It appears even more attractive when compared to an analyst manually examining files and defining characteristics that can be used to identify a file type.

The speed of fingerprint generation could be dramatically improved if only Option 3 is used. If only the header and trailer analysis is being performed, there is no reason to load the entire input files into memory. Only the number of bytes given by the selected header and trailer lengths need to be read and processed. This would not only significantly speed fingerprint generation, but would also increase the recognition time of unknown files for the same reason.

There would be a tradeoff in only using Option 3 in this manner, however. Not all file types have consistent file headers or trailers. File types that don't have a fixed header or trailer would most likely not be correctly recognized if only Option 3 were used. Options 1 and/or 2 could help with the identification of the few files Option 3 was unable to identify.

The overall difference in accuracy between All Options and only Option 3 was negligible. When Option 3 was wrong, the overall guess was normally wrong, even in cases where Option 1 or 2 provided the correct type. For example, when Option 3 was only 25% accurate across DOC, PPT, and XLS files, Option 2 was 75% accurate. However, when the results were combined, more weight was placed on Option 3, resulting in the misidentification of most of the files.

Another example of this is the first MP3 file used for accuracy testing. This file was incorrectly identified by Option 3 as an RM file. Option 1 correctly identified it as an MP3 file, however more weight was placed on Option 3 resulting in the overall misidentification of the file as an RM file. This suggests that improvements could be made in the assurance level equations for use in combining the scores.

Other improvements could be investigated in the methods used to score Options 1 and 2. Perhaps more sophisticated curve-matching algorithms could be tested to see if they would improve the accuracy of these options.

Improvements could be made to the header and trailer analysis as well. The header and trailer tests both showed degradation in performance as longer header and trailer lengths were used. It should be possible to modify the Option 3 algorithm to prevent this degradation. Ideally, if many extra bytes were included in the analysis, the variations in the extra byte values between files should cause the correlation strengths for these bytes (and thus their impact on the Option 3 score) to quickly reduce to near zero.

This would mean that only the bytes involved in the fixed headers and trailers would play a significant role in generating the Option 3 score. Longer header and trailer regions could then be analyzed without degrading accuracy. This could help with the differentiation between different file types that all use the OLE compound document format. If a header length were used longer than the OLE compound document header, then data corresponding to the true file types would be included in the analysis.

Another important goal of the algorithm was automatic generation of file type fingerprints. This was accomplished, except for the process of gathering a set of input files of known types. Each fingerprint requires a set of files of the appropriate file type for the fingerprint. This process requires a bit of care to make sure the files are of the correct type for the fingerprint. A file of the wrong type mixed into a fingerprint could degrade the recognition ability for that fingerprint.

Some thought also has to be put into selecting appropriate file types for fingerprints. The tests comparing accuracy using one OLE DOC fingerprint versus using separate ACD, DOC, PPT, and XLS fingerprints highlight the possible effects of choosing different fingerprints. Creating a single OLE DOC fingerprint dramatically increased the accuracy of Option 3, but decreased the accuracy of Option 1.

The final two design goals were small fingerprint size and independence from input file size. APPENDIX A shows the format of the fingerprint files. The size of the

fingerprint file depends upon which options were selected. If Option 3 was selected it also depends upon the selected header and trailer lengths.

The size of input files has no impact on fingerprint size, however. A fingerprint made up of twenty 10 MB video files can be stored in the same sized fingerprint as one made up of twenty 2 KB font files. Figure 5-2 shows a summary of the fingerprint file sizes in bytes for different option combinations.

	Size of Fingerprint in Bytes
Option 1 Only	2,053
Option 2 Only	262,153
Option 3 Only	$13 + 4(H + T)$
Options 1, 2, and 3	$264{,}209 + 4(H + T)$

Figure 5-2 - Fingerprint file sizes in bytes for different option combinations, where *H* is header length and *T* is trailer length.

Even if the input files are shorter than either the header or the trailer lengths, then no calculations will be performed for the missing byte positions, and all the corresponding array entries will be set to zero to show that no byte values occurred at those positions. If the input file length is longer than the header and trailer lengths, but shorter than the sum of the two lengths, then there will simply be overlap between the processed regions. The fingerprint size will remain constant for given options and fixed header and trailer lengths.

Overall, the algorithm proved effective at correctly identifying the file types of files based solely upon the content of the files. Option 3 identified executable files with 100 percent accuracy. This option could therefore be of use to virus scanning packages that are configured to only scan executable files. Option 3 was extremely fast (0.0006 seconds per fingerprint comparison,) and for header and trailer lengths of five bytes, the total fingerprint size for an executable fingerprint would be only 53 bytes. If just one fingerprint were required (the EXE fingerprint for example,) the extra processing and memory requirements to implement the recognition would be negligible.

The algorithm could possibly be of use to cryptanalysts as well. It could be used to automatically differentiate between real data and "random" encrypted traffic. To use it for this kind of purpose, a minimum score threshold could be added. If no fingerprint scores exceed the minimum score, the unknown data would be identified as random (possibly encrypted) traffic. If any fingerprint scores exceeded the threshold, then the highest-scoring fingerprint would be identified as the type of the unknown traffic.

A number of other systems could also benefit from the described file recognition approach. These include forensic analysis systems, firewalls configured to block

transfers of certain file types, and security downgrading systems. Further refinements would be required, however, before the algorithm would be fast enough or accurate enough to be used by an operating system that must reliably deal with a large number of varied file types.

Future Work

There are numerous areas in which additional research could be done. Many file types have recurrent patterns throughout the body that could be used for identification as well. An Option 4 could be evaluated and tested that would analyze language constructs and syntax. One example could be to identify the frequency and locations of recurrent patterns, similar to the operation of the ZIP compression algorithm.

This type of analysis could help differentiate between similar types by looking for file-type-specific tags and patterns. Tags used throughout HTML and RTF files could then be used to improve identification. As another example, the three-character pattern consisting of a semicolon followed by a carriage-return line-feed pair would be very frequent in a C/C++ source code file. A Visual Basic source code file, on the other hand, would have different recurrent keywords and patterns.

A risk to this type of syntax analysis would be that it could increase the chances that a large file of one type wrapped in another file type could result in a misidentification of the real type. If a large HTML file, for example, were embedded in an executable file, this type of option would likely recognize a large number of HTML-specific patterns, and could incorrectly identify the file type as HTML rather than EXE.

Another problem to this type of approach is that the processing overhead would be significant. This could be minimized, perhaps, by selecting a maximum length into the file to process, similar to the file header and trailer lengths. This would reduce the effectiveness of the option, however.

A simple, but very useful, extension would be to combine the described algorithm with available compression and archival algorithms. Files identified as a format such as ZIP, GZ, or TAR could then be run through the appropriate algorithm to extract the contents into memory. The file type recognition algorithm could then be rerun on each component, thereby identifying the type of each file contained within the original file.

Accuracy and speed were analyzed when each option was enabled separately, as well as when all options were enabled. The results showed that Option 1 comparisons do not provide any significant assistance to identification, and could conceivably even reduce accuracy. The arrays generated by Option 1 are used by Option 2, however. Therefore, Option 1 needs to be performed as part of fingerprint generation, but frequency distribution comparisons may not need to be performed when identifying unknown files. It could be informative to test the accuracy and speed when using only Options 2 and 3 for comparison, without Option 1.

Header and trailer lengths could be set individually per file type. This would allow for more bytes to be analyzed for file types that have longer headers or trailers, without incurring the overhead (and possible performance degradation) of processing the extra bytes on all file types.

An optimization could include having a maximum number of bytes from the beginning and end of a file to analyze for Options 1 and 2. If the number of bytes was set large enough to provide a meaningful sample of data, it could remain effective while dramatically decreasing the time required to process large files. A test would have to be run to assess any impact this optimization may have on accuracy.

To increase assurance in the accuracy of the algorithm, larger sample sizes should be analyzed. Various means of collecting large samples could be evaluated, including automated generation of sample data files, or customized web crawlers that would look for and download files of specified types. Manufactured sample data may not sufficiently reflect the variety that can be seen in real-world data. Real world data, however, would have to be verified prior to use to ensure that the files are in fact the types they claim to be.

The optimal values of constants were identified through empirical tests. More sophisticated statistical techniques could be investigated to improve on the accuracy of these values.

Similarly, the bell curve and weighted average equations themselves may not have been the best possible equations. Neural networks can be very good at identifying optimal equations for complex situations. Further research could involve developing a small neural network to detect the optimal equations for correlation strength and assurance level calculations.

Other areas where additional research could be performed are in regard to the assurance level and correlation strength equations. The assurance level algorithms could be improved to avoid the situation where one option identifies the correct type, but the overall algorithm misidentifies the type because an incorrect option was given too much weight. Similarly, there may be functions that would be more effective for correlation strength calculations than a bell curve.

Another interesting area of research would be to investigate the effectiveness of this approach in differentiating between normal executables and those containing viruses. A library of files infected with a variety of virus forms would have to be gathered. Fingerprints could then be generated for various forms of viruses, and identification tests could then be run to test the accuracy at distinguishing between normal files and infected files.

Finally, the algorithm could be extended and tested on forms of binary data other than simple static files. These could include binary data streams such as streaming audio or video, finer level analysis such as differentiating between IP and IPX traffic on a network, or even individual segments within a larger file. Using the previous example, if

a large HTML file were embedded in an executable wrapper, the current algorithm would identify the file as an executable because of the presence of non-printable characters and an executable file header. If segments within the file could be independently analyzed, both the wrapper format and the embedded data formats could potentially be identified. The most important factor in this type of application is likely to be the speed of the algorithm, so a balance would have to be reached between accuracy and speed.

This page intentionally left blank.

APPENDIX A: FINGERPRINT FILE FORMAT

Bytes	Field Description
1	Options Bitfield (76543210) (Bit 0 = Option 1, Bit 1 = Option2, Bit 2 = Option 3, Bits 3 through 7 are unused.) If a bit is 1, then that option is enabled in this fingerprint. If a bit is 0, then that option is not enabled.
4	Number of files added to fingerprint (4-byte integer value)
2048	Option 1 Data (256 sets of two four-byte floating point numbers.) This field is present if the Option 1 bit is set in the Options Bitfield. The first byte of each pair is the magnitude; the second byte is the corresponding correlation strength.
4	Sigma value. This field is present if the Option 2 bit is set in the Options Bitfield.
262144	Option 2 Data (256 x 256 four-byte floating point numbers.) This field is present if the Option 2 bit is set in the Options Bitfield.
4	Header Length. This field is present if the Option 3 bit is set in the Options Bitfield.
4	Trailer Length. This field is present if the Option 3 bit is set in the Options Bitfield.
X	Header Data. (The length of this field depends upon the value of Header Length. There is one four-byte floating-point number for each byte included in the Header Length.) This field is present if the Option 3 bit is set in the Options Bitfield.
Y	Trailer Data. (The length of this field depends upon the value of Trailer Length. There is one four-byte floating-point number for each byte included in the Trailer Length.) This field is present if the Option 3 bit is set in the Options Bitfield.

This page intentionally left blank.

APPENDIX B: FILE TYPES USED FOR CONSTANT TESTS

Archive Formats

- ZIP

Audio Formats

- MP3

Document Formats

- DOC
- HTML
- PDF
- PPT
- RTF
- TXT
- WPD
- XLS

Executable Formats

- EXE

Graphic Formats

- BMP
- GIF
- JPG

Video Formats

- MPEG

This page intentionally left blank.

APPENDIX C: FILE TYPES USED FOR ACCURACY TESTS

Archive Formats

- GZ
- ZIP
- TAR

Audio Formats

- WAV
- MP3

Document Formats

- DOC
- HTML
- PDF
- PPT
- PS
- RTF
- TXT
- WPD
- XLS

Executable Formats

- EXE
- RPM

Font Formats

- TTF
- FNT

Graphic Formats

- BMP
- GIF
- JPG

Video Formats

- AVI
- MPEG
- RM
- MOV

This page intentionally left blank.

APPENDIX D: FILE TYPES USED FOR EXTENDED ACCURACY TESTS

Archive Formats

- GZ
- ZIP
- TAR

Audio Formats

- WAV
- MP3

Document Formats

- DOC
- HTML
- PDF
- PPT
- PS
- RTF
- TXT
- WPD
- XLS

Executable Formats

- EXE
- RPM

Font Formats

- TTF
- FNT

Graphic Formats

- BMP
- GIF
- JPG

Video Formats

- AVI
- MPEG
- RM
- MOV

Proprietary Formats

- 3TF
- ACD
- CAT
- CRP
- MDL

This page intentionally left blank.

APPENDIX E: FILE TYPE FINGERPRINT OVERVIEW

This appendix contains summaries of the fingerprint of each file type. The top of each figure shows the Option 1 byte frequency distribution. The lower left portion of each figure shows the Option 2 byte frequency cross-correlation plot, split between average difference and correlation strength, as described in Chapter 2. The lower right portion of each figure shows the Option 3 file header plot (the top half of the Option 3 pane) and the file trailer plot (the bottom half of the Option 3 pane.)

The combined OLE DOC fingerprint is shown, as well as the individual fingerprints (ACD, DOC, PPT, and XLS.) Although there are some differences, they are very similar, and all share exactly the same file header and trailer patterns, characteristic of the OLD compound document format.

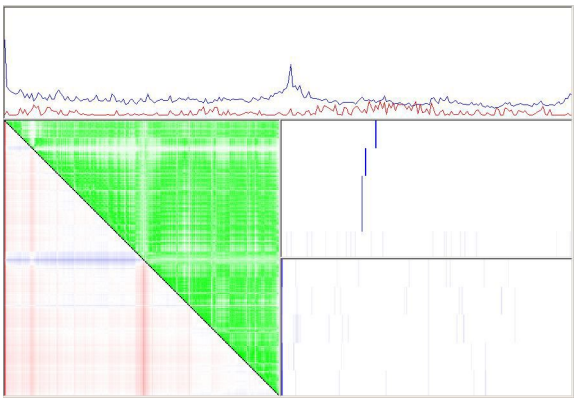

Figure D-1 - AVI fingerprint summary

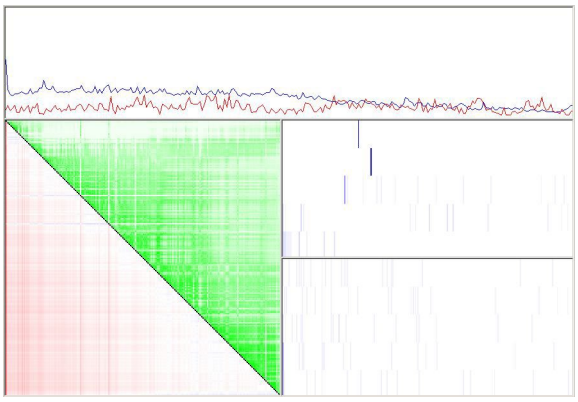

Figure D-2 - BMP fingerprint summary

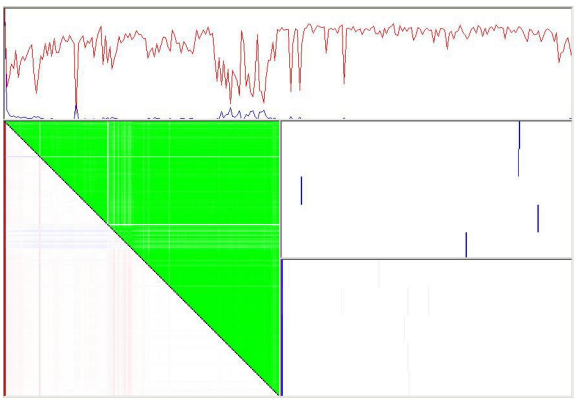

Figure D-3 - DOC fingerprint summary

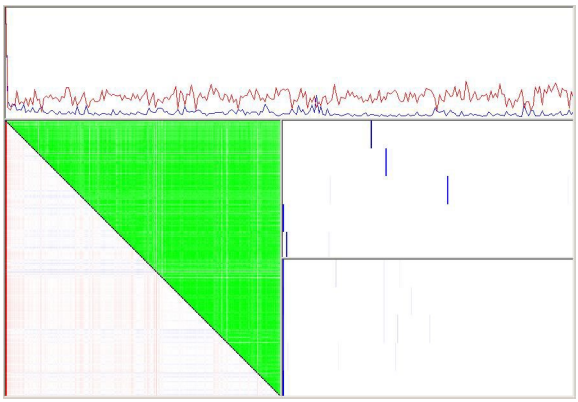

Figure D-4 - EXE fingerprint summary

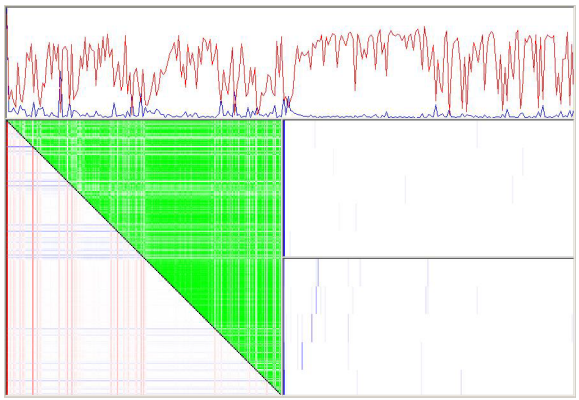

Figure D-6 - GIF fingerprint summary

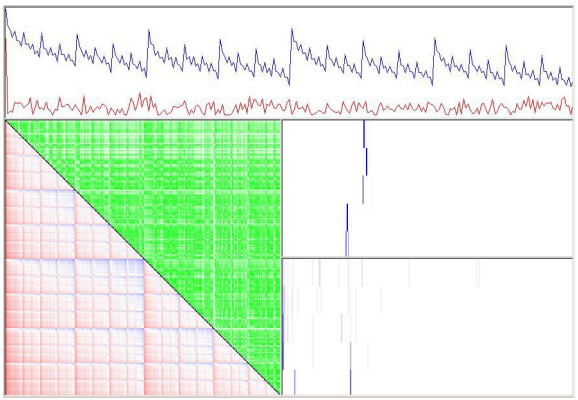

Figure D-5 - FNT fingerprint summary

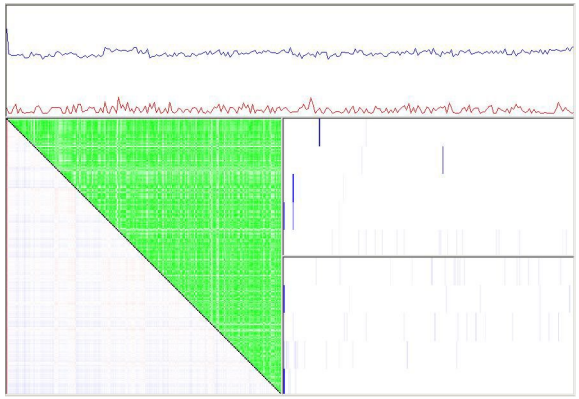

Figure D-7 - GZ fingerprint summary

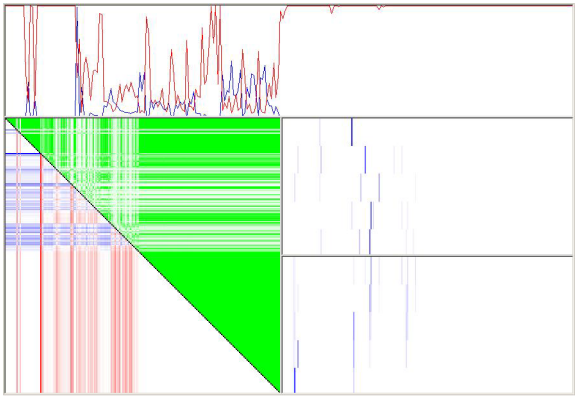

Figure D-8 - HTML fingerprint summary

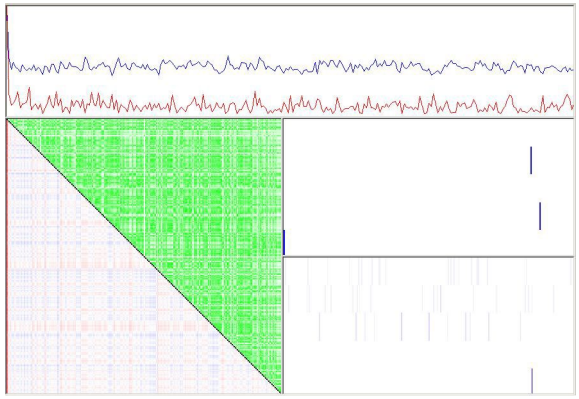

Figure D-9 - JPG fingerprint summary

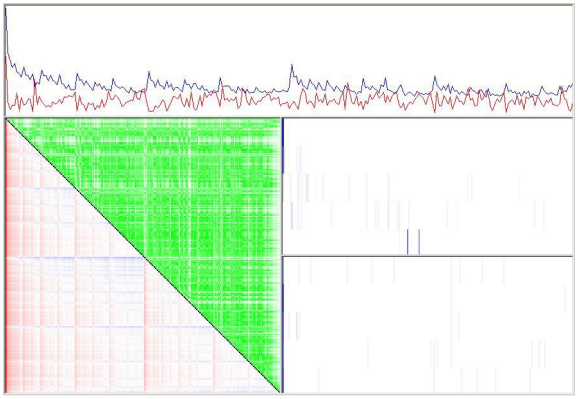

Figure D-10 - MOV fingerprint summary

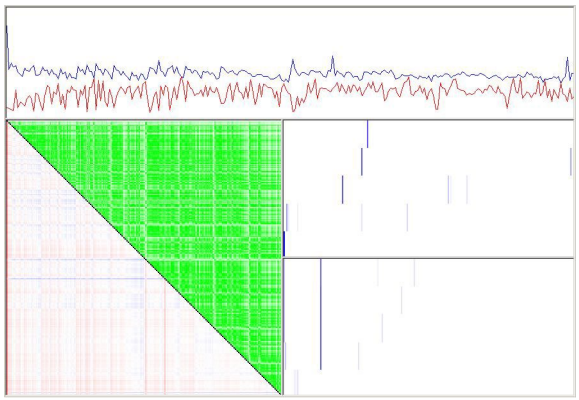

Figure D-11 – MP3 fingerprint summary

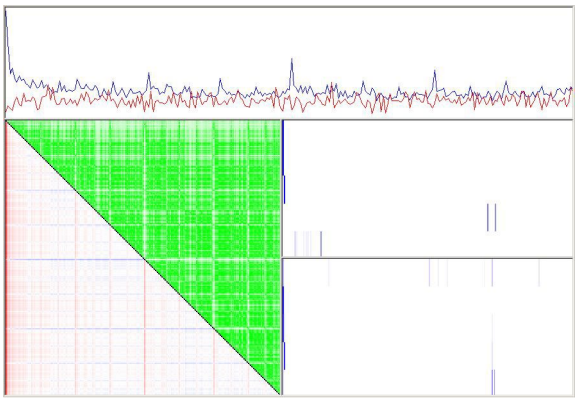

Figure D-12 - MPEG fingerprint summary

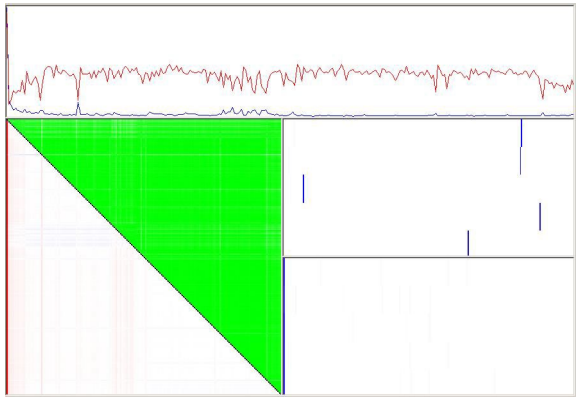

Figure D-13 – OLE DOC fingerprint summary

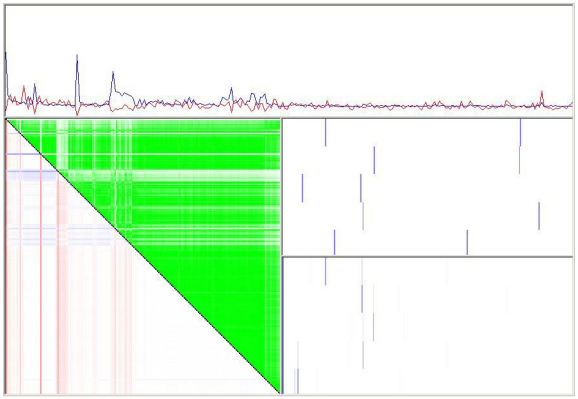

Figure D-14 - PDF fingerprint summary

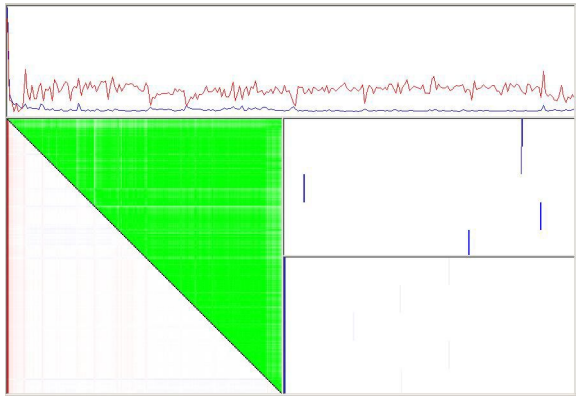

Figure D-15 - PPT fingerprint summary

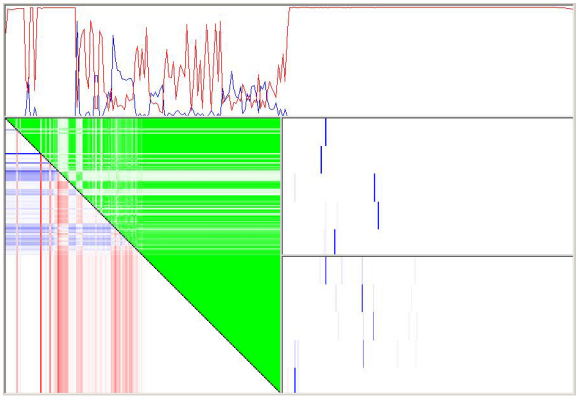

Figure D-16 - PS fingerprint summary

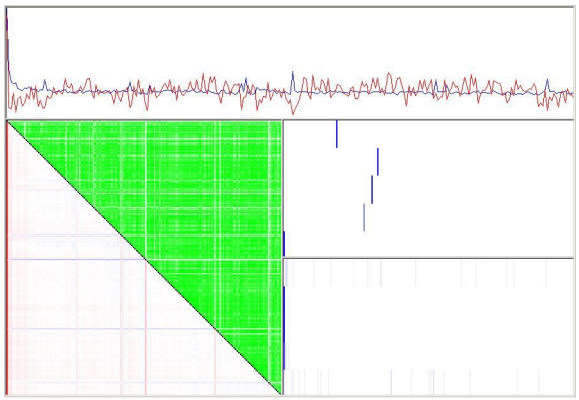

Figure D-17 - RM fingerprint summary

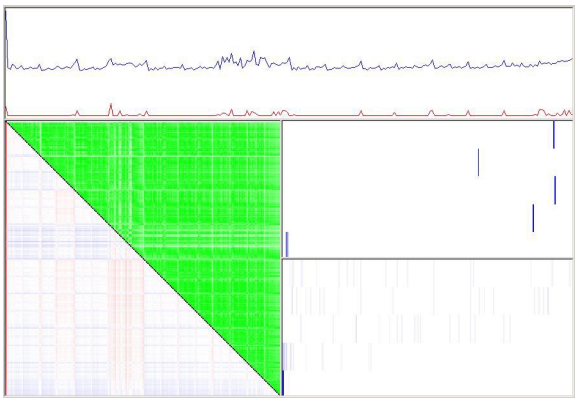

Figure D-18 - RPM fingerprint summary

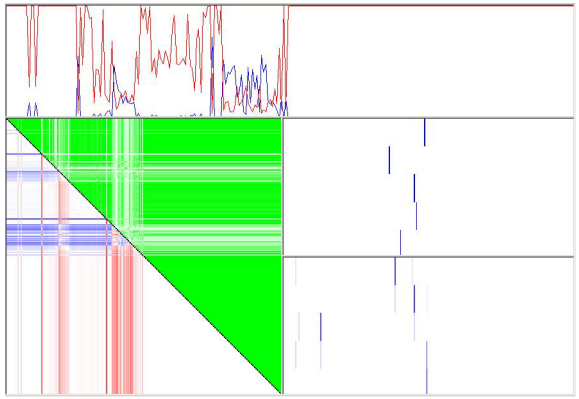

Figure D-19 - RTF fingerprint summary

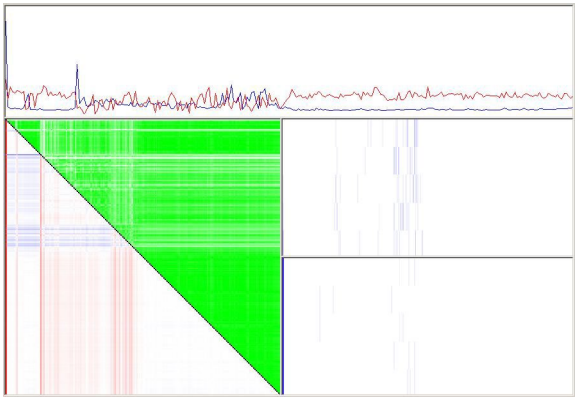

Figure D-20 - TAR fingerprint summary

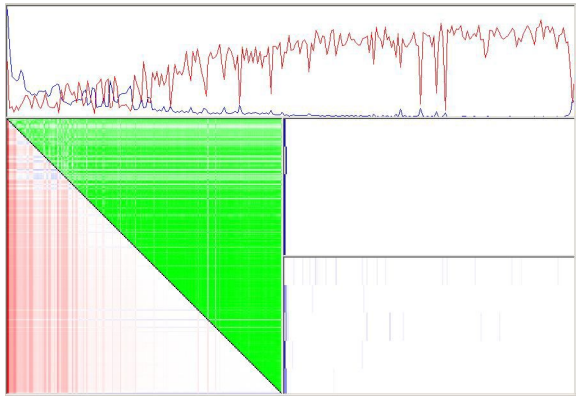

Figure D-21 - TTF fingerprint summary

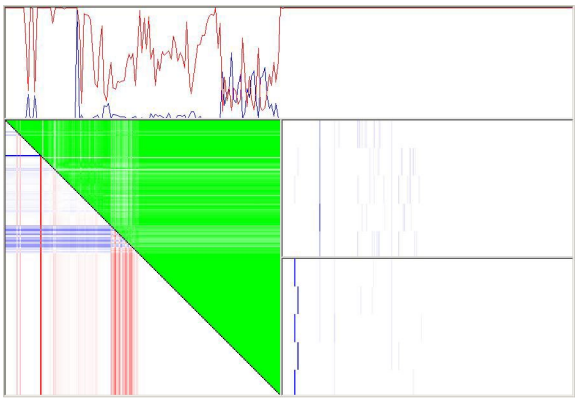

Figure D-22 - TXT fingerprint summary

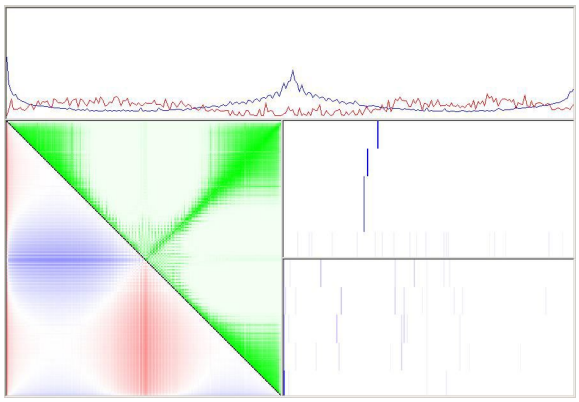

Figure D-23 - WAV fingerprint summary

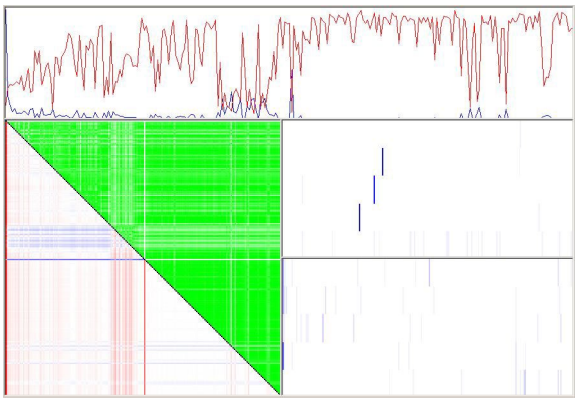

Figure D-24 - WPD fingerprint summary

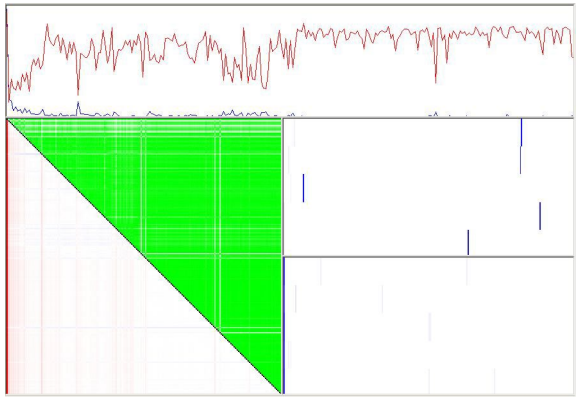

Figure D-25 - XLS fingerprint summary

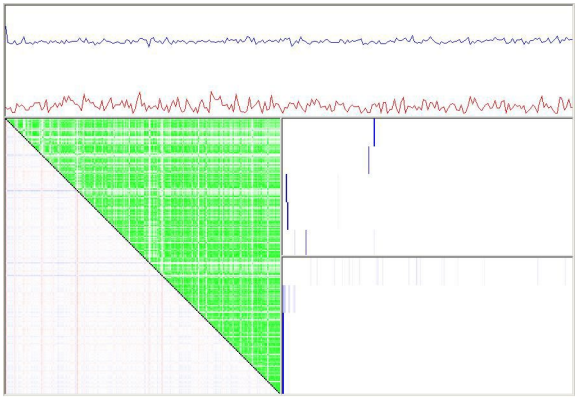

Figure D-26 - ZIP fingerprint summary

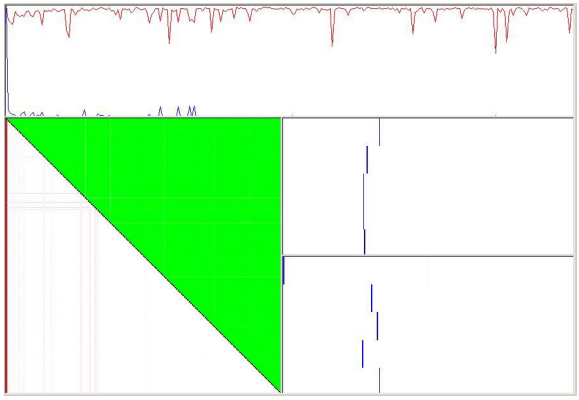

Figure D-27 – 3TF fingerprint summary

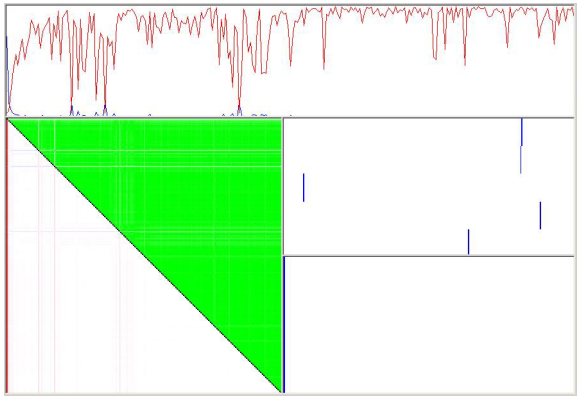

Figure D-28 – ACD fingerprint summary

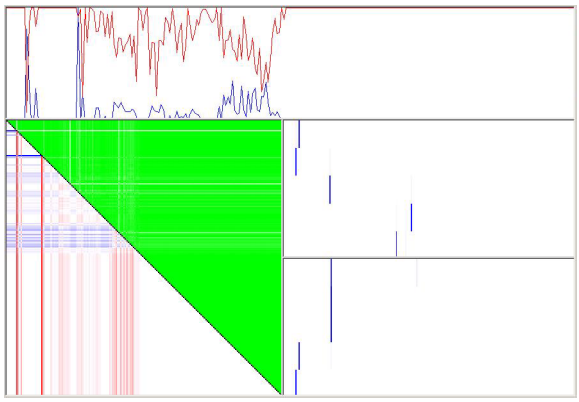

Figure D-29 – CAT fingerprint summary

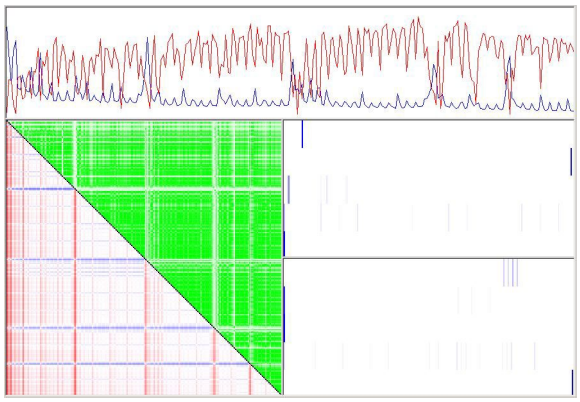

Figure D-30 – CRP fingerprint summary

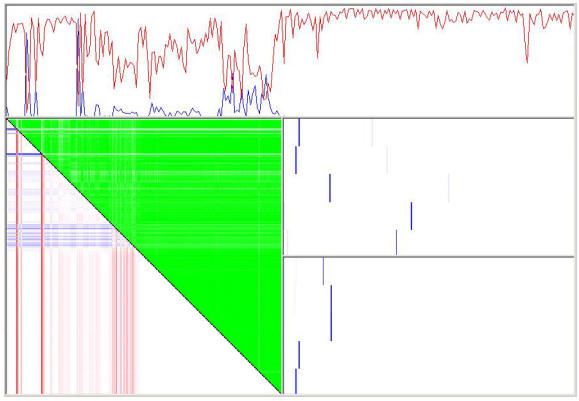

Figure D-31 – MDL fingerprint summary

APPENDIX F: SAMPLE FILE RECOGNITION REPORT

Filename: C:\Development\FTR\bin\Input Test Files\PDFinput2

Overall Closest Match: PDF

Byte Frequency Closest Match: TXT
Cross-Correlation Closest Match: PDF
File Header Closest Match: PDF
File Trailer Closest Match: PDF

Overall Scores:

0.0642511	BMP
0.2214751	XLS
0.09202309	EXE
0.03012224	GIF
0.3025353	HTML
0.04225569	JPG
0.0553562	MP3
0.05165713	MPEG
0.684445	PDF
0.0865095	PPT
0.3084091	RTF
0.3042476	TXT
0.2276875	DOC
0.2275722	WPD
0.06118989	ZIP

Frequency Scores:

Raw Score	Assurance Level	Effective Score	
0.009720683	0.2332596	0.002267443	BMP
0.5574311	0.8970471	0.500042	XLS
0.2017136	0.4924424	0.09933233	EXE
0.004134795	0.2348216	0.000970939	GIF
0.7862476	0.8130449	0.6392546	HTML
4.99264E-07	0.2475621	1.235989E-07	JPG
6.825497E-13	0.4824367	3.29287E-13	MP3
2.962304E-07	0.3988085	1.181392E-07	MPEG
0.0001700508	0.4646344	7.901146E-05	PDF
0.07778424	0.5287558	0.04112887	PPT
0.7367388	0.8620027	0.6350709	RTF
0.7031381	0.9123371	0.641499	TXT
0.597475	0.9016744	0.5387279	DOC
0.575063	0.8515723	0.4897077	WPD
0.02841559	0.2274891	0.006464239	ZIP

Cross-correlation Scores:

Raw Score	Assurance Level	Effective Score	
0.354186	0.314932	0.1115445	BMP
0.4234604	0.4816526	0.2039608	XLS
0.3670453	0.4727979	0.1735383	EXE
0.1584871	0.4229975	0.06703965	GIF
0.5378711	0.3981051	0.2141292	HTML
0.2737767	0.3969876	0.1086859	JPG
0.3299123	0.4501055	0.1484953	MP3
0.3240619	0.4454128	0.1443414	MPEG
0.4770142	0.4738864	0.2260505	PDF
0.4350897	0.4847711	0.2109189	PPT
0.488732	0.4488078	0.2193467	RTF
0.4573833	0.4824656	0.2206717	TXT
0.4526621	0.4914457	0.2224588	DOC
0.3755475	0.4609013	0.1730903	WPD
0.3986118	0.384787	0.1533806	ZIP

Header Scores:

Raw Score	Assurance Level	Effective Score	
0.004054054	1	0.004054054	BMP
0	0.95	0	XLS
0	1	0	EXE
0	1	0	GIF
0.03037975	0.9	0.02734177	HTML
0	1	0	JPG
0	1	0	MP3
0	1	0	MPEG
1	1	1	PDF
0	1	0	PPT
0.1666667	1	0.1666667	RTF
0.00609756	0.55	0.003353658	TXT
0	1	0	DOC
0	0.95	0	WPD
0	1	0	ZIP

Trailer Scores:

Raw Score	Assurance Level	Effective Score	
0.002941176	0.3	0.0008823529	BMP
0	0.85	0	XLS
0	1	0	EXE
0	0.6	0	GIF
0	0.8	0	HTML
0.003061224	1	0.003061224	JPG
0	0.7500001	0	MP3
0	0.95	0	MPEG
0.7904255	0.95	0.7509043	PDF
0	0.9	0	PPT
0	1	0	RTF
0	0.9	0	TXT
0	0.95	0	DOC
0	0.65	0	WPD
0	1	0	ZIP

Detailed Fingerprint Scores:

Fingerprint 0: BMP - SCORE: 0.0642511
  Byte Frequency:
    Score: 0.009720683
    Assurance Level: 0.2332596
  Cross-Correlation:
    Score: 0.354186
  Assurance Level: 0.314932
  File Header:
    Score: 0.004054054
    Assurance Level: 1
  File Trailer:
    Score: 0.002941176
    Assurance Level: 0.3

Fingerprint 1: XLS - SCORE: 0.2214751
  Byte Frequency:
    Score: 0.5574311
    Assurance Level: 0.8970471
  Cross-Correlation:
    Score: 0.4234604
    Assurance Level: 0.4816526
  File Header:
    Score: 0
    Assurance Level: 0.95
  File Trailer:

Score: 0
Assurance Level: 0.85

Fingerprint 2: EXE - SCORE: 0.09202309
Byte Frequency:
Score: 0.2017136
Assurance Level: 0.4924424
Cross-Correlation:
Score: 0.3670453
Assurance Level: 0.4727979
File Header:
Score: 0
Assurance Level: 1
File Trailer:
Score: 0
Assurance Level: 1

Fingerprint 3: GIF - SCORE: 0.03012224
Byte Frequency:
Score: 0.004134795
Assurance Level: 0.2348216
Cross-Correlation:
Score: 0.1584871
Assurance Level: 0.4229975
File Header:
Score: 0
Assurance Level: 1
File Trailer:
Score: 0
Assurance Level: 0.6

Fingerprint 4: HTML - SCORE: 0.3025353
Byte Frequency:
Score: 0.7862476
Assurance Level: 0.8130449
Cross-Correlation:
Score: 0.5378711
Assurance Level: 0.3981051
File Header:
Score: 0.03037975
Assurance Level: 0.9
File Trailer:
Score: 0
Assurance Level: 0.8

Fingerprint 5: JPG - SCORE: 0.04225569
Byte Frequency:

Score: 4.99264E-07
Assurance Level: 0.2475621
Cross-Correlation:
Score: 0.2737767
Assurance Level: 0.3969876
File Header:
Score: 0
Assurance Level: 1
File Trailer:
Score: 0.003061224
Assurance Level: 1

Fingerprint 6: MP3 - SCORE: 0.0553562
Byte Frequency:
Score: 6.825497E-13
Assurance Level: 0.4824367
Cross-Correlation:
Score: 0.3299123
Assurance Level: 0.4501055
File Header:
Score: 0
Assurance Level: 1
File Trailer:
Score: 0
Assurance Level: 0.7500001

Fingerprint 7: MPEG - SCORE: 0.05165713
Byte Frequency:
Score: 2.962304E-07
Assurance Level: 0.3988085
Cross-Correlation:
Score: 0.3240619
Assurance Level: 0.4454128
File Header:
Score: 0
Assurance Level: 1
File Trailer:
Score: 0
Assurance Level: 0.95

Fingerprint 8: PDF - SCORE: 0.684445
Byte Frequency:
Score: 0.0001700508
Assurance Level: 0.4646344
Cross-Correlation:
Score: 0.4770142
Assurance Level: 0.4738864

File Header:
Score: 1
Assurance Level: 1
File Trailer:
Score: 0.7904255
Assurance Level: 0.95

Fingerprint 9: PPT - SCORE: 0.0865095
Byte Frequency:
Score: 0.07778424
Assurance Level: 0.5287558
Cross-Correlation:
Score: 0.4350897
Assurance Level: 0.4847711
File Header:
Score: 0
Assurance Level: 1
File Trailer:
Score: 0
Assurance Level: 0.9

Fingerprint 10: RTF - SCORE: 0.3084091
Byte Frequency:
Score: 0.7367388
Assurance Level: 0.8620027
Cross-Correlation:
Score: 0.488732
Assurance Level: 0.4488078
File Header:
Score: 0.1666667
Assurance Level: 1
File Trailer:
Score: 0
Assurance Level: 1

Fingerprint 11: TXT - SCORE: 0.3042476
Byte Frequency:
Score: 0.7031381
Assurance Level: 0.9123371
Cross-Correlation:
Score: 0.4573833
Assurance Level: 0.4824656
File Header:
Score: 0.00609756
Assurance Level: 0.55
File Trailer:
Score: 0

Assurance Level: 0.9

Fingerprint 12: DOC - SCORE: 0.2276875
Byte Frequency:
Score: 0.597475
Assurance Level: 0.9016744
Cross-Correlation:
Score: 0.4526621
Assurance Level: 0.4914457
File Header:
Score: 0
Assurance Level: 1
File Trailer:
Score: 0
Assurance Level: 0.95

Fingerprint 13: WPD - SCORE: 0.2275722
Byte Frequency:
Score: 0.575063
Assurance Level: 0.8515723
Cross-Correlation:
Score: 0.3755475
Assurance Level: 0.4609013
File Header:
Score: 0
Assurance Level: 0.95
File Trailer:
Score: 0
Assurance Level: 0.65

Fingerprint 14: ZIP - SCORE: 0.06118989
Byte Frequency:
Score: 0.02841559
Assurance Level: 0.2274891
Cross-Correlation:
Score: 0.3986118
Assurance Level: 0.384787
File Header:
Score: 0
Assurance Level: 1
File Trailer:
Score: 0
Assurance Level: 1

This page intentionally left blank.

GLOSSARY

3TF – Perspective ChartX template file format.

ASSURANCE LEVEL – A value indicating how much confidence can be placed on an associated score. A file with no characteristic file trailer should have a very low assurance level associated with file trailer scores.

ACD – AllClear diagram file format.

AVI – Audio Visual Interface, a file format for video files.

BIT – The smallest unit of data a computer can manipulate, representing one of two states, often represented by a 0 or a 1.

BMP – Bitmap graphics file format.

BYTE – An eight-bit number with a valid range of 0 to 255.

BYTE FREQUENCY – The relative number of occurrences of each byte value in a file.

BYTE FREQUENCY CROSS-CORRELATION – The relationship between byte value frequencies. If two different byte values always occur with the same frequency, then the cross-correlation between the two byte values will be high.

CAT – Rational Rose catalog file format.

CORRELATION FACTOR – A numeric score that denotes how closely the results of a new file match the results already in a fingerprint. The correlation factor from each new file is added into the fingerprint to generate a correlation strength.

CORRELATION STRENGTH – A rating of how consistent a result is across all files in a fingerprint.

CRP – Electronic Arts, Need for Speed Porsche Unleashed data model file format.

DOC - Microsoft Word document.

EXE – Executable file format.

FILE HEADER – Consistent pattern of bytes that appears at the beginning of a file.

FILE TRAILER – Consistent pattern of bytes that appears at the end of a file.

FINGERPRINT – A summary of a file type containing information required for recognition of files of that type.

FNT – A bitmapped font file format.

GIF – Graphics Interchange Format, a file format for graphic files.

GZ – Gzip compression file format.

HTML – Hypertext Markup Language, the dominant language for defining web pages.

JPG – Short for JPEG, Joint Photographic Experts Group. JPG is a compressed graphic file format.

MDL – Rational Rose model file format.

MOV – QuickTime movie file format.

MP3 – MPEG audio layer 3, a file format for storing compressed audio.

MPEG – Short for "moving picture experts group", a file format for storing compressed multimedia data.

PDF – Short for "portable document format", a document file format developed by Adobe.

PPT – Microsoft PowerPoint presentation file.

PS – PostScript file.

RTF – Rich Text Format, a file format for storing text with additional markup tags.

RM – Real Media audio/visual file format.

RPM – Short for "RedHat package manager", a Linux file format for storing self-installing software packages.

TAR – Short for "tape archive". A UNIX archive file format.

TXT – Text file format.

TTF – True Type Font file format.

WAV – A file format for storing audio data.

WPD – Word Perfect document file format.

XLS – Microsoft Excel spreadsheet file format.

ZIP – File format for storing data compressed using the ZIP compression algorithm.

BIBLIOGRAPHY

/etc/magic Help File, available online from:
http://qdn.qnx.com/support/docs/qnx4/utils/m/magic.html

Bellamy, John, Digital Telephony, Second Edition, John Wiley & Sons, Inc., New York, New York, 1991, pp 110-119.

The Binary Structure of OLE Compound Documents, available online from:
http://user.cs.tu-berlin.de/~schwartz/pmh/guide.html

Kyler, Ken, Understanding OLE Documents, Delphi Developer's Journal, September 1998, available online from: http://www.kyler.com/pubs/ddj9894.html

Stallings, William, Cryptography and Network Security, Prentice Hall, upper Saddle River, New Jersey, 1999, p. 32.

The Advanced Missile Signature Center Standard File Format, available online from:
http://fileformat.virtualave.net/archive/saf.zip

To Associate a File Extension with a File Type, Windows 2000 Professional Documentation, available online from:
http://www.microsoft.com/WINDOWS2000/en/professional/help/win_fcab_reg_filetype.htm

Why do some scripts start with #!, Chip Rosenthal, available online from:
http://baserv/uci/kun.nl/unix-faq.html