

# Sliding Window Measurement for File Type Identification

Gregory A. Hall, Ph.D., Computer Forensics and Intrusion Analysis Group, ManTech Security & Mission Assurance, Gregory.Hall@ManTech.com

Wilbon P. Davis, Department of Computer Science, Texas State University-San Marcos, Wilbon.Davis@txstate.edu

**Abstract** - Knowing the file type associated with a sequence of bytes makes interpretation of those bytes far more meaningful. With the ever increasing number of file types in existence and the massive storage capacity of modern hardware, it is impractical to try interpreting a sequence of bytes as every known file type until one succeeds. Furthermore, some file types require specific header or footer information that, if missing, makes the data unusable. This paper presents new measurements that can be used to assist in determining the file type associated with a file of unknown type. These measurements can be used by a variety of algorithms to attempt to determine file type. Initial results are presented and avenues for future research are discussed.

**Keywords** – entropy, file carving, forensics, metrics

## I. Introduction

There are times when the type of file represented by a sequence of bytes needs to be determined using properties of the file's structure. One simplistic example is when someone attempts to disguise a file of one type as a file of another type. Simply changing the file extension on a Windows-based computer will often allow a user to make an image file masquerade as another type of file. Forensic investigators have a number of tools at their disposal that can overcome this simple ruse. Such tools often look for "magic numbers" that can identify whether or not a given file matches the type indicated by its extension. However there are some file types for which there are no magic numbers or there may be instances where the sequence of bytes represents a partial file that no longer includes the magic number. Such is the case for files recovered from slack space with file carving tools.

This paper describes a mechanism for using measurements taken on a file to determine the file type associated with that file. In some cases, the file type could be determined using the UNIX *file* command. However, the technique described here can be used for files that do not contain magic

numbers or to identify files that are very similar to one another.

While file type identification is the first application to which this measurement approach has been applied, there are a variety of other avenues of research that are promising. Among them is identifying encrypted versions of known files.

## II. Entropy and Compressibility

For the purpose of this research, two metrics were used to assess the information content of the file. The metrics were entropy and compressibility.

### *Entropy*

In information theory, entropy is a measure of the predictability or randomness of data. A file with a highly predictable structure or a value that repeats very frequently would have low entropy. Such files would be considered to have low information content (or low information density). Files in which the next byte value is relatively independent of the previous byte value would be considered to have high entropy. Such files would be thought to have high information content.

### *Compressibility*

Similar to entropy, the compressibility of a file is directly related to the information content of the file. If the file consists of several, repeating sequences, then that file will be more compressible than one with many unique values. Files that are information sparse would compress at a high ratio. Files that are already compressed (information rich) would not be very compressible at all.

The measurement of entropy was chosen because it is a commonly understood measure in information theory. The measurement of entropy has been used in a variety of information systems based studies [1] [2]. The measurement of compressibility was chosen because it appears related to file entropy but may be more efficient to calculate.

The inspiration for this research came from a paper by Shannon that used file entropy and the percentage of ASCII characters within a file to detect files which may not be of their reported type [3]. This paper extends that idea to use a sliding window approach to measuring entropy which accounts for variations in entropy caused by a file's internal structure. The goal of this approach is to guess the correct file type based on its entropy profile (which is defined by a set of entropy values taken from within the file). Similar work has been done by Li, et al, using n-gram analysis for file type identification [4]. They also make use of a sliding window metric, n-gram distribution, to categorize files by type.

### III. Sliding Window Measurement

Previous uses of entropy have focused on processing the file as a whole. In this manner, a particular file (or file type) can be characterized by a single number representing its information content. For example, text files containing written English have been identified as having a file entropy in the range of 3.25 to 4.5 bits. Compressed files, such as ZIP archives, have a higher entropy level typically just over 6 bits.

The limitation of using a single entropy value to characterize a file is that many file types may have an entropy value very close to that of another file type. For example, compressed JPEGs also have an entropy value of about 6 bits which is very close to that of ZIP archives. Indeed, many compressed files have a similar overall entropy value.

The structure of some file types results in some areas having lower entropy than others. For example, a file type which has a header containing file metadata in text format with compressed data in the body would differ from a binary file of purely random values. The lower entropy at the start of the file would distinguish it from the file of constant high entropy. To account for these structured file formats, and to make them distinguishable from other files with very similar overall entropy values, a sliding window approach to measurement was used.

In sliding window measurement, a fixed window size is determined prior to measurement. Naturally, the window size is some value smaller than the full size of the file. The process begins by measuring the first window's worth of bytes. The window then slides over one byte. The effect of moving the window is that the contribution of the first byte in the window is removed and the contribution of the byte entering the window is introduced. The newly repositioned

window yields a new measurement value. The window continues to be slid, gathering new measurements values, until the end of the file is reached. Note that for efficiency the entire window does not need to be remeasured, only the effect of removing the first byte and adding the new last byte needs to be computed.

#### *Entropy algorithm*

With each position of the sliding window, we associate a measure of the average information per character in the window, namely Shannon's order-0 entropy. The value is given by

$$H(W) = - \sum_{c \in S_W} P(c) \log_2(P(c))$$

or

$$H(W) = - \frac{1}{|S_W|} \sum_{c \in S_W} f(c) \log_2(f(c))$$

where  $S_W$  is the set of characters in the window  $W$ ,  $w$  is the length of  $W$ ,  $P(c)$  is the probability that a character chosen at random from the window will be the character  $c$ , and  $f(c)$  is the frequency of the character  $c$  in  $W$ .

The second formula can be rewritten as

$$\begin{aligned} H(W) &= - \sum_{c \in S_W} \frac{f(c)}{n} \log_2 \frac{f(c)}{n}, \text{ where } n = |S_W| \\ &= - \sum_{c \in S_W} \frac{f(c)}{n} \log_2 f(c) + \sum_{c \in S_W} \frac{f(c)}{n} \log_2 n \\ &= - \frac{1}{n} \sum_{c \in S_W} f(c) \log_2 f(c) + \log_2 n \end{aligned}$$

which allows simple calculation of the contribution to a rolling entropy of each character dropped from a window and each character added as the window position moves. Note that  $n$  may change with each window movement.

Sliding a small sized window the entire length of a large data file (such as digital video) naturally results in a very large number of data points. To address this issue, only a subset of those data points is actually collected. The number of points to be collected is chosen prior to measurement and then that many

samples are taken at approximately equal intervals throughout the file.

#### *Compressibility algorithm*

For sliding window compressibility, the LZW algorithm was selected. It was chosen because the algorithm is well-known and implementations are easily available. For efficiency purposes, the actual compression is not performed. Instead, the number of changes made to the compression dictionary is used as a surrogate measure of the actual compressibility of the data within the window.

Rather than explore the cases of individual entropies of order higher than 0, we instead use a measure of local compressibility as a proxy. Following [8], we implement an abbreviated version of the LZW algorithm in which we record only sizes of the compression dictionary and use the changes in the size over a window as our measure.

#### IV. File Type Identification

Initial results using sliding window entropy indicated that many commonly used file types have characteristic fluctuations in entropy throughout the file. Figure 1 depicts the sliding window entropy, (a), and the sliding window compressibility, (b), for a single Microsoft Word DOC file. Figures 2 and 3 provide similar pairs of plots for Microsoft PowerPoint PPT and Microsoft Excel XLS files. A window size of 90 bytes was used and a total of 100 sample points were taken from within the file. The window size and number of points to plot were selected based on trial-and-error experience. Note that the window size and number of points to plot does impose a constraint on the files that can be meaningfully measured. At 90 bytes, a file must be no less than 9000 bytes in size in order to obtain 100 data points. This can preclude measurement of files under 9 Kb in size.

The first step in the research was to amass a collection of files of various types that could be measured using sliding window entropy and sliding window compressibility. An initial set of files was collected from a personal computer belong to the lead author. Software was written that would measure and record sliding window entropy or compressibility for all files beneath a specified root directory. A file naming convention was developed for the output measurements that would uniquely identify each file by the type indicated by its extension, its size, and its MD5 hash value. This precluded duplicate file measurements from entering the data set (the second

occurrence of the file would overwrite the first). A second, larger data set was available was made available by ManTech for the purposes of this research. The benefits of the second data set are its size and the fact that the files were not personally selected by the authors.

The first data set consisted of a total of 73,382 files and represented 454 file types (based on file extension or the results of the file command). The second data set consisted of a total of 265,340 files and represented 109 file types.

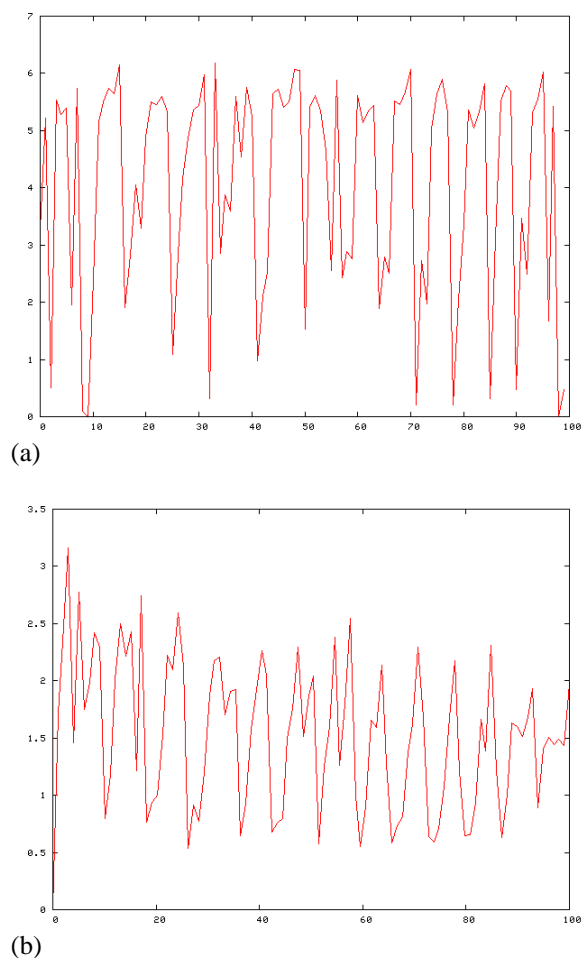


Figure 1. Example DOC file (a) sliding window entropy (b) compressibility

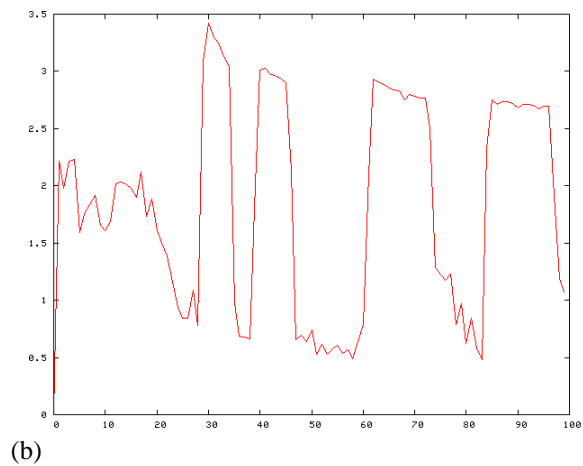
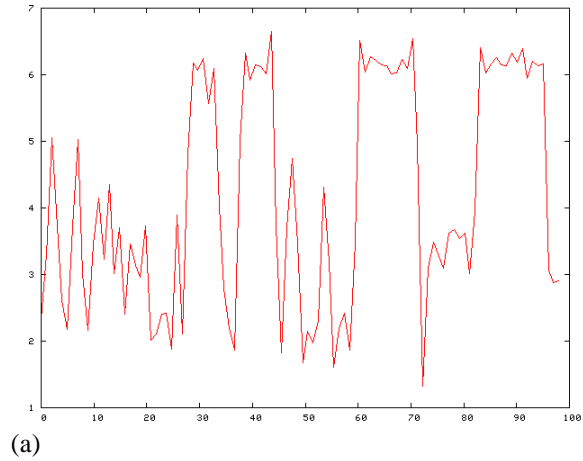


Figure 2. Example PPT file (a) sliding window entropy (b) compressibility

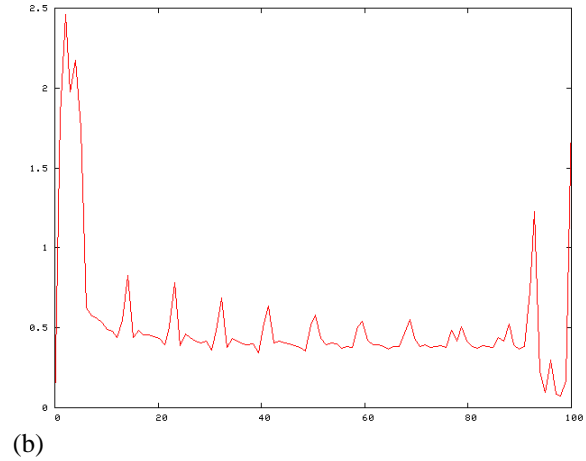
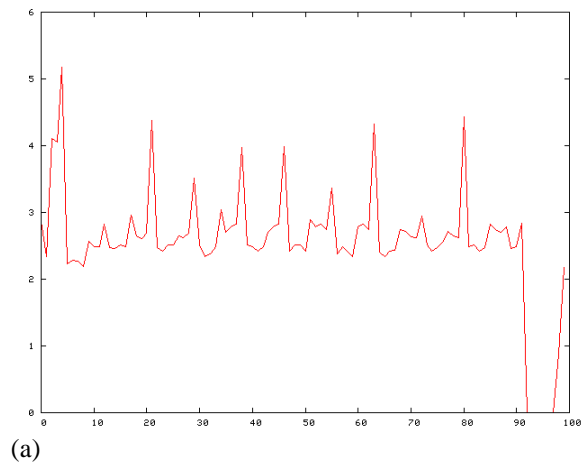
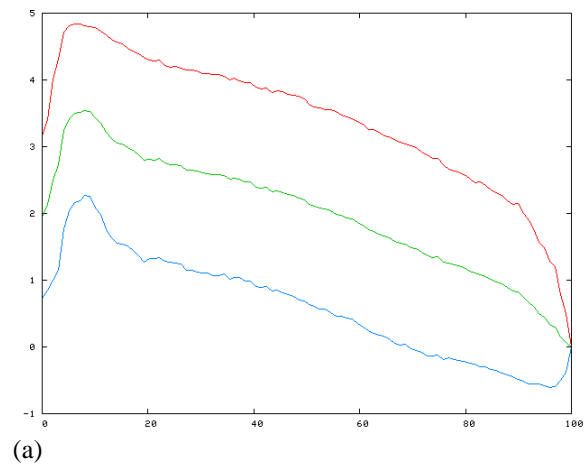
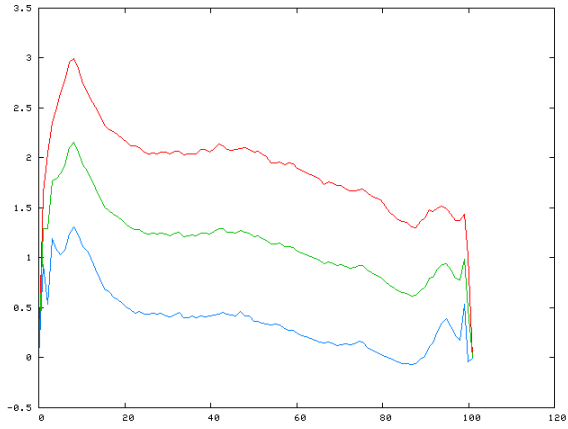


Figure 3. Example XLS file (a) sliding window entropy (b) compressibility

For each file type, the measurements were averaged and standard deviations calculated for each corresponding data point. Figures 4 through 9 depict plots of the average point values for each of the 100 data points collected, with Figure 4 providing both average sliding window entropy, (a) and average compressibility, (b). In each of the graphs, the green center line is the average point value and the red and blue lines represent one standard deviation above and below the average. Obviously, the closer the blue and red lines are to the green line at any point the more confidence there is in the average value for that point. Notice that the Y axis of each graph changes, affecting the maximum and minimum depicted values as well as the scale of the graph. Some graphs that appear to have very wide confidence intervals about the average may only appear so because of the scale of the Y axis.





(b)

Figure 4. Average DOC (a) sliding window entropy (b) compressibility with confidence intervals (1928 files)

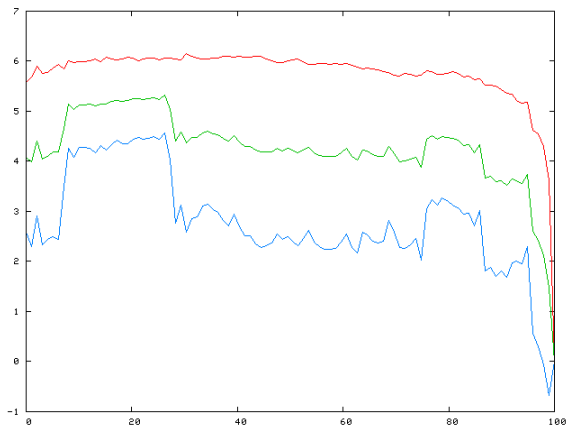


Figure 5. Average EXE sliding window entropy with confidence intervals (355 files)

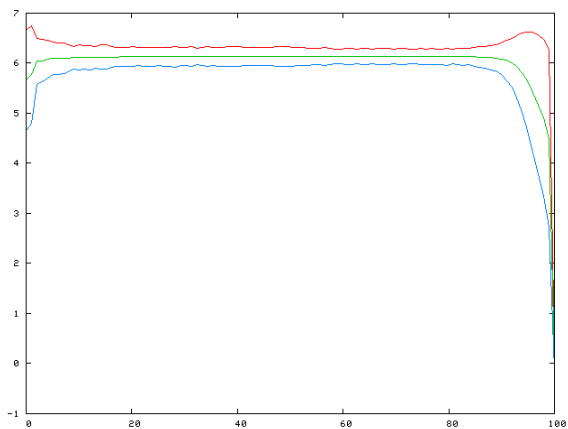


Figure 6. Average JPG sliding window entropy with confidence intervals (1604 files)

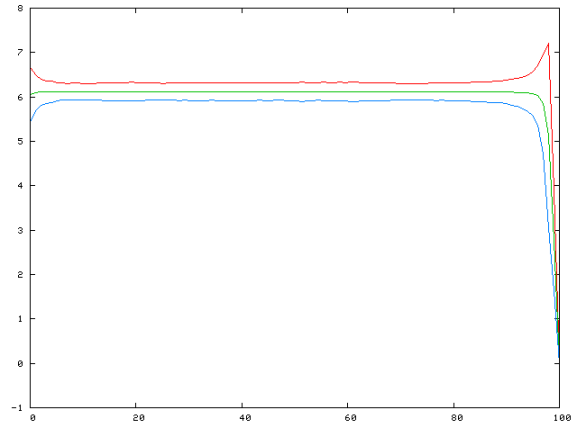


Figure 7. Average MP3 sliding window entropy with confidence intervals (32238 files)

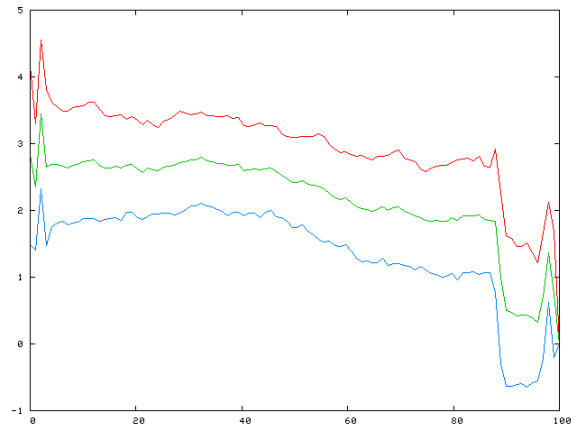


Figure 8. Average XLS sliding window entropy with confidence intervals (242 files)

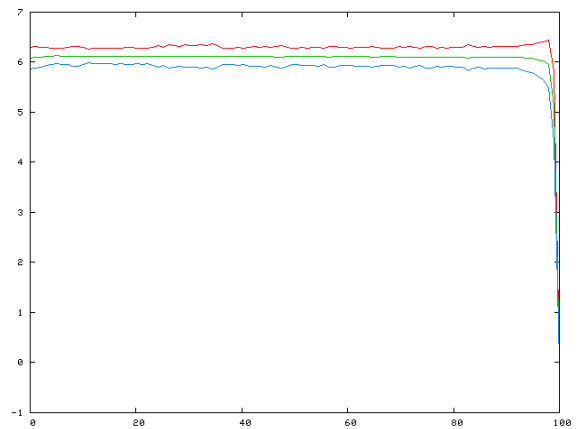


Figure 9. Average ZIP sliding window entropy with confidence intervals (1755 files)

Some file types (for example BMP) can vary greatly in their internal entropy or compressibility from file to file. Such file types can be difficult if not impossible to correctly classify based on their sliding window measures due to this variability. However it

would still be possible to classify a file into the group of types having high variability or lack of consistent internal structure.

With two large data sets and average entropy and compressibility profiles computed for each file type, the next step in the research was to develop a mechanism for associating a file with a file type. As the sliding window measurements can be plotted as a graph or curve and the average profiles are represented in the same manner, a graph or curve fitting approach seems appropriate.

*Point-by-point Delta*

The most straight-forward approach to fitting a sliding window plot to an average profile plot is to do a point by point comparison. At each point throughout the profile, the file’s value is subtracted from the average profile’s value. The absolute value of this difference is accumulated for all points in the plot to form a measure of “goodness of fit.” The smaller this value, the closer the file is thought to fit that particular average profile. The goodness of fit is computed for all of the file types for which an average profile exists. The best guess as to the file’s type is then the file type whose average profile resulted in the smallest goodness of fit value.

*Pearson’s Rank Order Correlation*

Another method for determining how well two data sets relate is to compute the correlation of their data. By computing the Pearson Rank Order Correlation coefficient, an assessment of how tightly correlated the set of data values is can be obtained. The higher the correlation, the more closely the data values rise and fall with one another. High, positive correlation values were used to indicate that the file’s data values rose and fell similarly to the average profile’s values. Small or negative correlation values were considered indicative of data sets that did not have similar plots.

*Others*

There are other measures for curve fitting or comparing similarity of data sets. Some other measures were experimented with. One approach used was to compute the point-by-point delta values for only the header and footer of the file, omitting the interior portion of the file. Another approach was to keep a count of the number of times the plot of the file crossed the average profile’s center line. Another as yet untested approach would be to count the number of times the file’s plot left the region containing one standard deviation above or below the average. Future research will likely include investigation into better algorithms for fitting files to file type profiles.

V. Results

The first experiment performed was to use point-by-point delta comparison of file plots to average entropy plots for data set 1. Accuracy was determined by the number of files of a given type that were correctly identified over the total number of files of that type. The results were promising for a few file types (such as ARJ and RM), but successful identification of other file types was limited (low percentage of correctly identified files).

Table 1 displays part of the results obtained by performing a point-by-point comparison of a specific ZIP file’s sliding window entropy against a set of average entropy plots.

File Type	Score
ra	6.33938686631
zip	7.05641975523
bz2	8.6999825
jar	9.26838999996
ram	9.510545

Table 1. Point-by-point delta results for a sample ZIP file

For the ZIP file in question, the sliding window entropy plot is most similar to the RA file format’s average sliding window entropy plot using the point-by-point delta method. However, the correct file type’s average entropy plot is ranked second. Also, the top five matches are all compressed files, some of which may even use a similar compression algorithm to that used for ZIP files.

It became clear from analyzing these initial results that this approach may be more effective at narrowing down the potential file types than it is at deterministically assigning the type. The assessment of accuracy was then adjusted to determine how frequently the correct file type was one of the top ten matches. Figure 10 displays the results of this assessment for ZIP files.

```
zip ( 1934 ) :
  1 ( 235 ) : 0.121509824199
  2 ( 700 ) : 0.361944157187
  3 ( 340 ) : 0.175801447777
  4 ( 320 ) : 0.165460186143
  5 ( 264 ) : 0.136504653568
  6 ( 16 ) : 0.00827300930714
  7 ( 3 ) : 0.00155118924509
  8 ( 1 ) : 0.000517063081696
  9 ( 1 ) : 0.000517063081696
 10 ( 3 ) : 0.00155118924509
Total percentage: 0.973629782834
```

Figure 10. Accuracy of categorizing ZIP file type

Only twelve percent of ZIP files were correctly identified by using the closest match found using point-by-point deltas. However, 97 percent of ZIP files had the correct file type appear in the top ten matches. In fact, 96 percent of ZIP files had the correct file type appear in the top five matches.

The results for all file types were then re-evaluated to determine how often the correct file type is associated with a file within the top ten matches. Those results are presented in Table 2 for several common file types using the point-by-point delta method. The table indicates the percentage of files of a given type that were correctly identified by one of the top ten matches for both data sets 1 and 2. Cells in the table that contain a dash indicate that there were no files of that type in data set 2. A value of zero indicates that no files of that type were correctly identified within the top ten matches.

It is evident from the results for data set 2 that sliding window entropy was not as successful at associating the correct file type with a file. Take for example the ZIP file type. For data set 1, ZIP files were identified 97 percent of the time in the top ten guesses. For data set 2, however, ZIP files were only correctly identified in the top ten guesses 9 percent of the time. This situation was investigated to determine the reason for this discrepancy.

Data set 2, although much larger in size than data set 1, contains only 100 files that end with a .zip file extension. Data set 1 contains 1934 such files. Due to the fact that the files in data set 1 were files on the lead author’s personal computer, there is a high degree of confidence that files with .zip extensions actually are of the ZIP file type. The files in data set 2 were amassed from a variety of sources, some of which were not completely trustworthy. Inspection of the 100 supposedly ZIP files, 14 files were found not to be ZIP files at all. These files were evaluated using the UNIX file command. 10 of the files were simply classified as “data” while others were Assembler source, Windows Shortcut, TrueType Font data, and MPEG. Of the files that were reported as being ZIP compressed data, 43 were version 1.0 and 43 were version 2.0. The resulting average entropy profile for the ZIP file format, with erroneous profiles included, looks like Figure 11.

Comparing Figure 9 with Figure 11, it can be seen that the confidence in the average values at each point is much higher in Figure 9. The source of the data used to produce Figure 9 along with the larger sample size would tend to indicate it is a more accurate and realistic profile of the ZIP file type.

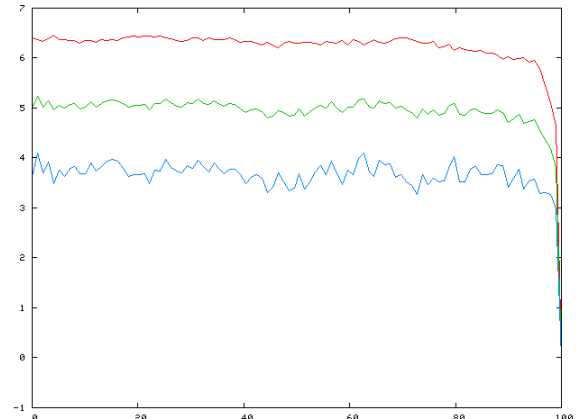


Figure 11. Average ZIP sliding window entropy for 100 files with erroneous data included.

The result of the smaller sample size and the inclusion of bad (non-ZIP) profiles results in an average plot in Figure 11 that is lower than expected and with a higher standard deviation. It is unknown what the effect of different ZIP archive versions (1.0 versus 2.0) may have had on the analysis. Furthermore, of the 100 ZIP files in data set 2, only 83 of them were above 9000 bytes. All of these inconsistencies in data set 2 for the ZIP file format alone make it unreliable as a validating data set.

Using the average entropy plot depicted in Figure 9 to identify sliding window entropy plots of supposed ZIP files from data set 2 resulted in an accuracy of 54 percent. This is still incorrect due to the inclusion of profiles that were too small or not taken from actual ZIP files. Removal of the bad data yields an accuracy of 78 percent.

File Type	Set 1	Set 2
bz2	100	-
cgm	94	-
doc	51	2
elf	91	-
eps	75	-
exe	0	29
fig	100	-
gif	94	4
html	90	4
jar	100	-
jpg	99	12
mp3	97	11
o	91	-
pcx	92	-
pdf	20	15
ppt	49	21
ra	100	-
rm	100	71

tgz	100	-
tif	85	6
wma	100	-
xls	84	7
xml	82	3
z	100	-
zip	97	9

Table 2. Sliding Window Entropy point-by-point delta

Table 3 presents the results obtained when using correlation to determine file type. In some instances, correlation was less accurate. For example, the BZ2 file type falls from 100 percent to 63 percent. However the EXE file type, for data set 1, rises from zero to 42 percent.

File Type	Set 1	Set 2
bz2	63	-
cgm	75	-
doc	58	41
elf	97	-
eps	75	-
exe	42	43
fig	83	-
gif	31	21
html	12	26
jar	100	-
jpg	49	29
mp3	96	58
o	55	-
pcx	77	-
pdf	56	20
ppt	52	68
ra	59	-
rm	99	83
tgz	53	-
tif	49	23
wma	83	-
xls	60	42
xml	55	11
z	100	-
zip	75	46

Table 3. Sliding Window Entropy correlation

The remaining tables, 4 and 5, present the results obtained by using the compressibility measurement in both the point-by-point delta (Table 4) and correlation (Table 5) methods. Overall, the compressibility measure did not seem to be as effective at correctly identifying file type. Yet for some file types, such as DOC files, the accuracy was superior to that obtained using the sliding window entropy measure.

File Type	Set 1	Set 2
bz2	0	-
cgm	0	-
doc	72	52
elf	0	-
eps	0	-
exe	0	86
fig	0	-
gif	0	84
html	13	24
jar	0	-
jpg	0	95
mp3	0	31
o	63	-
pcx	0	-
pdf	0	32
ppt	50	11
ra	0	-
rm	100	83
tgz	0	-
tif	93	94
wma	93	-
xls	93	52
xml	88	15
z	0	-

Table 4. LZ compressibility point-by-point delta

File Type	Set 1	Set 2
bz2	0	-
cgm	0	-
doc	65	30
elf	0	-
eps	0	-
exe	0	71
fig	0	-
gif	0	34
html	32	46
jar	0	-
jpg	0	52
mp3	0	83
o	48	-
pcx	0	-
pdf	0	23
ppt	66	68
ra	0	-
rm	99	83
tgz	0	-
tif	91	38
wma	100	-
xls	90	48
xml	75	33
z	0	-
zip	0	61

Table 5. LZ compressibility correlation

In light of these results, it would seem that a combination of measurements and comparison methods might be most effective. The results obtained by all four sources might be merged to improve the overall accuracy of file type identification.

## VI. Future Work

The initial results of using sliding window measurement for file type identification are promising. While the closest match is not always the correct one, the correct match is usually one of the top ten for many common file types. Furthermore, if the exact file type is not completely necessary, identifying compressed versus non-compressed data is very accurate with this method. There has also been some initial success categorizing kinds of text documents (HTML, XML, Python scripts, etc.).

The effect of file size needs to be further investigated. For example, JPG image files ranged in size for a few kilobytes to over a megabyte. By taking the same number of samples from a smaller file as are taken from a larger file, more data points from the header are involved for that file. This difference in the number of points taken from high entropy versus low entropy regions of a file could affect the results. More analysis needs to be done to determine if accuracy is affected when files of similar size are used for comparison.

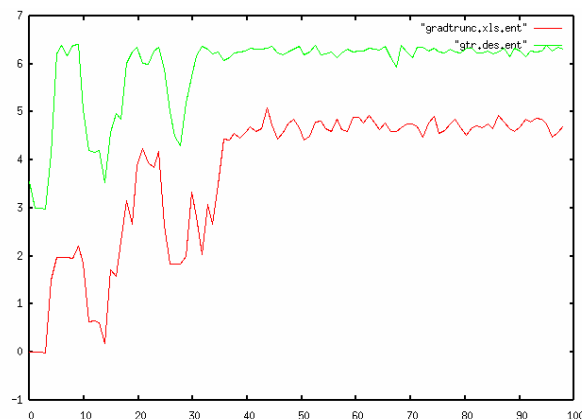
In file carving, it is often the case that only part of a file can be recovered. Future research will involve using sliding window measurements of partial files and routines for fitting these partial plots to profiles for known file types. It is expected that these routines for fitting partial plots will be more computationally intensive, having to take into account where the best fit exists within each profile. The potential exists for this approach to not only determine the file type of a fragment, but also to determine where within the file the fragment is likely to have been.

Sliding window measurement may also be useful in finding similar files. If a file has only been slightly modified from its original, the entropy plot of the original file may be very similar to that of the modified file. This is an area that has not currently been investigated.

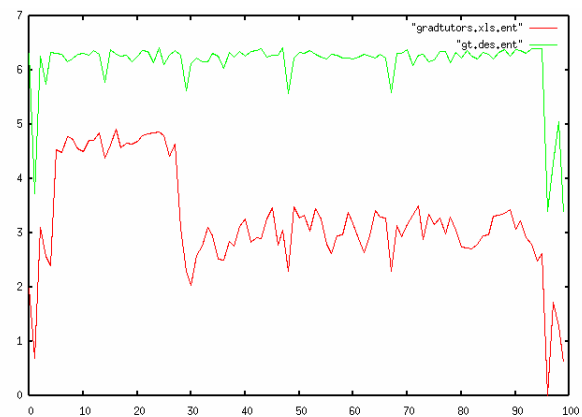
Some interesting work has been done by Karresand and Shahmeri using byte frequency distribution (BFD) and rate of change (RoC) to associate file

fragments with file types based on their binary structure [5]. They have reported a high success rate at identifying JPG file fragments. Future research may seek to incorporate their measures into the sliding window process or perform a comparison of the effectiveness of the two different techniques.

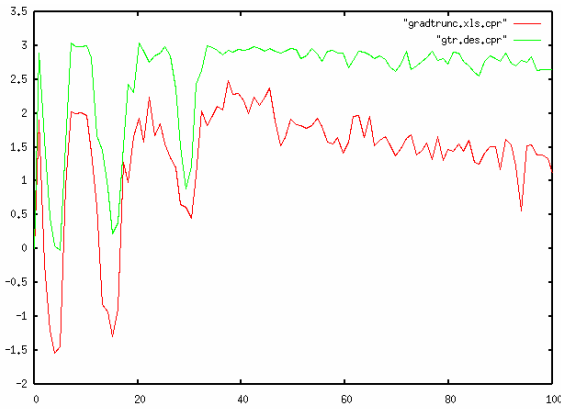
Some initial investigation has been performed to link encrypted and non-encrypted versions of files using their sliding window entropy measurements. Figure 12 shows the encrypted (green) and unencrypted (red) plots for four Excel spreadsheet files. While the encrypted versions of the files have slightly elevated entropy, the overall shape of the plot seems to be preserved (although smoothed out a bit). Further experimentation needs to be performed to see just how accurately we can match encrypted file plots to their un-encrypted plots.



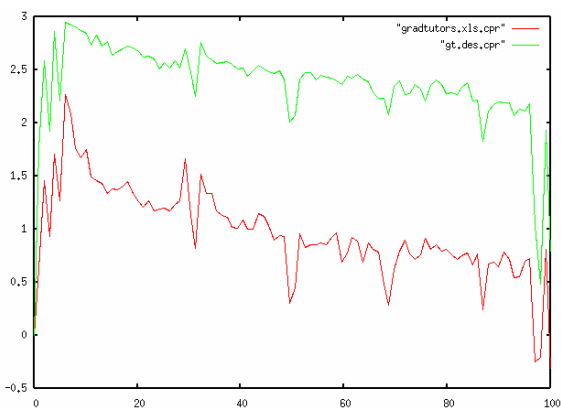
(a)



(b)



(c)



(d)

Figure 12. Example sliding window entropy for encrypted and unencrypted versions of (a) Excel spreadsheet 1, (b) Excel spreadsheet 2, (c) Excel spreadsheet 3 and (d) Excel spreadsheet 4

Shamir and Someren have used entropy to identify cryptographic keys using the entropy of hard disk data fragments [6]. This initial work speaks to the promise of using sliding window entropy in the encryption arena.

## VII. References

- [1] Osborne, M., "Using Maximum Entropy for Sentence Extraction," Proceedings of the Workshop on Automatic Summarization, Philadelphia, July 2002, pp. 1-8.
- [2] Cardoso, A., Crespo, R. and Kokol, P., "Assessing Software Structure by Entropy and Information Density," Software Engineering Notes, Volume 29, Number 2, March 2004, pp. 1-3.
- [3] Shannon, M., "Forensic Relative Strength Scoring: ASCII and Entropy Scoring,"

International Journal of Digital Evidence, Spring 2004, Volume 2, Issue 4, <http://www.ijde.org>

- [4] Lei, W., Wang, K., Stolfo, S. and Herzog, B., "Fileprints: Identifying File Types by n-gram Analysis," Proceedings from the Sixth Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop, June 15-17, 2005, New York, New York, pp. 64-71.
- [5] Karrasand, M. and Shahmehri, N., "File Type Identification of Data Fragments by Their Binary Structure," 2006 IEEE Information Assurance Workshop, June 21-23, 2006, United States Military Academy, West Point, New York, pp. 140-147.
- [6] Shamir, A. and Someren, N., "Playing 'hide and seek' with stored keys," in Financial Cryptography: Third International Conference, FC'99, Volume 1648 of LNCS, Springer-Verlag, 1999, pp. 118-124.

## VIII. About the Authors

Dr. Gregory A. Hall is a principal forensic engineer in the Computer Forensics and Intrusion Analysis Group within ManTech International Corporation. Dr. Hall is a former Assistant Professor of Computer Science from Texas State University-San Marcos specializing in digital forensics, software engineering and quality assurance.

Professor Wilbon P. Davis is the interim Computer Science Department Chair at Texas State University-San Marcos. Professor Davis is an active researcher and educator in digital forensics. In his forty years as a Professor, Wilbon Davis has taught courses in all areas of computer science at both the undergraduate and graduate levels.