

# Differential Area Analysis for Ransomware Attack Detection within Mixed File Datasets

Simon R. Davies<sup>a,\*</sup>, Richard Macfarlane<sup>a</sup> and William J. Buchanan<sup>a</sup>

<sup>a</sup>*School of Computing, Edinburgh Napier University, Edinburgh, UK*

---

## ARTICLE INFO

*Keywords:*

Entropy

Ransomware Detection

Test Data Sets

---

Article history

---

## ABSTRACT

The threat from ransomware continues to grow both in the number of affected victims as well as the cost incurred by the people and organisations impacted in a successful attack. In the majority of cases, once a victim has been attacked there remain only two courses of action open to them; either pay the ransom or lose their data. One common behaviour shared between all crypto ransomware strains is that at some point during their execution they will attempt to encrypt the users' files. This paper demonstrates a technique that can identify when these encrypted files are being generated and is independent of the strain of the ransomware.

An enhanced mixed file ransomware data set of more than 130,000 files was developed based on the govdocs1[17] corpus. This data set was enriched to contain examples of files that reflect the more modern Microsoft file formats, as well as examples of high entropy file formats such as compressed files and archives. The data set also contained eight different sets of files that were generated as the result of different real-world high profile ransomware attacks such as WannaCry, Ryuk, Phobos, Sodinokibi and NetWalker.

Previous research [39, 56] has highlighted the difficulty in differentiating between compressed and encrypted files using Shannon entropy as both file types exhibit similar values. One of the experiments described in this paper shows a unique characteristic for the Shannon entropy of encrypted file header fragments. This characteristic was used to differentiate between encrypted files and other high entropy files such as archives. This discovery was leveraged in the development of a file classification model that used the differential area between the entropy curve of a file under analysis and one generated from random data. When comparing the entropy plot values of a file under analysis against one generated by a file containing purely random numbers, the greater the correlation of the plots is, the higher the confidence that the file under analysis contains encrypted data. The experiments demonstrate a high degree of confidence in the accuracy of the model achieving a success rate of more than 99.96% when examining only the first 192 bytes of a file, using a mixed data set of more than 80,000 files. This technique successfully addresses the problem of using file entropy to differentiate compressed and archived files from files encrypted by ransomware in a timely manner.

---

## 1. Introduction

Ransomware is a malicious class of software that utilises encryption as a method to implement an attack on system availability. The victim's data remaining encrypted and held captive by the attacker until the ransom demand is met [6].

The occurrence of ransomware infection continues to grow in scale, cost, complexity and impact since its initial discovery nearly 30 years ago [54, 55]. Security practitioners are engaged in a continual arms race with ransomware developers attempting to defend their digital infrastructure against such attacks. During the recent changes in working practices due to COVID-19 where an increasing number of employees are working from home, ransomware attacks have increased, and attack vectors have changed. Currently the most common attack vectors are Remote Desktop Protocol (RDP) and phishing emails [9], and average ransomware payments have increased by over 60% in the second quarter of 2020 to now be in excess of \$170,000 per incident. Examples of recent high profile attacks where it is believed that the ransom has been paid are the Wastedlocker strain that attacked Garmin [42], the Netwalker strain that attacked the

University of Utah [37] and the Sodinokibi/Revil strain that attacked the owner of the Jack Daniel's distillery [48].


Ransomware infection detection tends to fall into one of these approaches:

- Behavioural/Dynamic analysis. This is where the suspicious processes behaviour is monitored to see if it follows the profile of known attacks. Many techniques have been proposed using this approach for detection and infection prevention. These can be classified into four broad categories: file-based detection, system-based behaviour detection, resource-based behaviour detection, and connection-based behaviour detection [1, 2, 13, 34]. Candidates for monitoring could be CPU usage, disk access, system calls and attempted external communications.
- Signature/Static analysis. This is where the executable code is analysed prior to its execution in an attempt to identify known sequences of bytes that indicate the probability that the program is malicious.

Obfuscation techniques and polymorphism [1, 34] have been used by ransomware for some time in an attempt to avoid signature detection. With regards to the dynamic detection approach, the behaviour of different ransomware strains varies significantly making them difficult to profile and

---

\*Principal corresponding author

 s.davies@napier.ac.uk (S.R. Davies)

0000-0001-9377-4539 (S.R. Davies); 0000-0003-0809-3523 (W.J.

Buchanan)

this approach is often ineffective against unknown strains. While some strains are self-contained - knowing which tasks to perform and having the encryption keys included in the binary - others attempt to contact external command and control (C&C) servers for guidance on how to proceed. Some strains also exfiltrate data or attempt lateral movement prior to encryption. What is becoming increasingly obvious is that as the arms race continues and new strains incorporate an increasing amount of these diverse techniques into each iteration, thus the difficulty in ransomware infection detection continues to be a problem.

Many ransomware detection methods have a low detection rate. They also suffer from having high false positive rates, where they flag benign programs as malicious, and false-negative rates, and thus fail to identify malicious programs. Also, as current detection techniques need to collect significant amounts of information while monitoring, they have the disadvantage that they can consume a large amount of system resources [29].

One thing that is common amongst all crypto-ransomware strains is that, at some point, they will attempt to encrypt the users' files and write these encrypted files to disk. It is this common feature of ransomware execution that the experiments in this paper investigate. Continella et al. [8] define that filesystem access is a strategic point for monitoring typical ransomware activity.

This paper demonstrates a technique that could be used to rapidly identify the creation of these encrypted files. The proposed technique tests only the first few bytes of the file being written and performs analysis on this file sample to determine if the file being written is encrypted or not. Similar to other research [2, 8, 20, 23, 29, 46, 56], initially this analysis will focus on the Shannon entropy [49], but unlike previous research it will only be performed on the file's header. The paper describes a differential area analysis technique which compares the entropy values of the file under analysis against the entropy values of reference file that contains random data. It is envisaged that this technique could be expanded to include other types of tests along with Shannon entropy. The described technique could in theory, be used to alert the user of suspicious behaviour, prevent the files from being written or trigger some further live forensic investigation [10]. While this technique would not prevent any data exfiltration prior to the start of encryption, it may well be useful in preventing the actual encryption of the user's data, and thus mitigating a significant portion of the attack.

### 1.1. Paper Contribution

The use of file entropy as a reliable method for encrypted file identification has been previously investigated by several researchers [2, 8, 20, 29, 46, 56]. When evaluating the proposed detection methods in the reviewed research, it appears that only a limited subset of file types was used and no examples were found where compressed files formed part of the test data set. The proposed techniques also tended to focus on the overall entropy of the file with only Jung and Won [23] examining the entropy of PDF file fragments.

It is a relatively straightforward task, though resource-intensive, to use a file's overall entropy value to differentiate an encrypted file with the majority of the most commonly used file types. However, using the files entropy value to differentiation between compressed files and encrypted files becomes problematic as the overall entropy values are normally similar for these file types. This paper thus focuses on this specific problem of entropy similarity and proposes a technique that will allow the rapid identification of encrypted files. This ability to be able to reliably detect encrypted file creation can then be used as an input for a ransomware detection technique. The main contribution of this paper is to define a classification model that can be used to reliably classify encrypted data files, even amongst a data set that contains files with a high overall entropy.

### 1.2. Paper Structure

The remainder of the paper is structured as follows. Section 2 contains a review of related work. Section 3 describes the design philosophy followed in Section 4 with a description of the experiment implementation. A critical analysis of the experimental results and comparison to similar work in the field is provided in Section 5. The paper closes in Section 6 with a discussion of the findings and suggestions for further research.

## 2. Related Work

Previous uses of entropy have focused on processing the file as a whole. With this approach, a particular file (or file type) can be characterised by a bit value 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 six bits [20]. In some cases in malware detection, the entropy of the entire file being written has been used as an indicator for the presence of malicious activity. Products such as DropIT [46], ShieldFS [8] and Unveil [25] check the entropy of the whole file being created, and combined with other variables such as the file's magic number and file extension, decide on whether to allow the file to be created. However, the use of file extensions as an indicator is open to abuse as an adversary can easily change the extension of a file or its magic number at any time, preventing the operating system from identifying the file and allowing the attacker to circumvent the detection [33]. A major issue when using entropy for file type classification is raised by Zhao et al. [56] where they state that when considering entropy as a gauge, most compressed and encrypted data share similar characteristics and they confirm that more work is required to investigate the application of entropy to these file types.

### 2.1. File fragment analysis

A similar, but related field of research, is in file fragment classification. This involves research into determining the type of a target file by only analysing a portion or a fragment of the file. This is particularly useful in the fields of

digital forensics. Some methods used are entropy, n-gram analysis, statistical analysis, machine learning and support vector machines [7]. McDaniel and Heydari [33] describe a method of using byte frequency analysis of a file's header and footer to build a *fingerprint* that can be used to identify files of a similar type. During their testing, they identified that it was a viable approach but that it needs to be combined with other methods to improve accuracy. They also postulate that file header/trailer analysis (FHT) could be used for ransomware executable detection as only a small sample is required. They found that for successful identification, a fingerprint size of only 53 bytes was required. The algorithm could possibly be of use in cryptanalysis and also to automatically differentiate between real data and encrypted samples [33].

With regards to file fragment classification, it has been noted that the structure of some file types, for example, GIF or PPT, results in some areas of the file having lower entropy than other areas within the same file. 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 a file of consistently high entropy. To account for these structured file formats, and to make them distinguishable from other files with similar overall entropy values, a sliding window approach to measurement has been evaluated [20].

Li et al. [30] used a similar approach where the file types were determined by only analysing either the first 20 or the first 200 bytes of a file using n-gram analysis. They reported that their tests successfully identified files and found that their technique performed as well, if not better, than analysing the whole file, with superior computational performance [30]. However, no compressed or encrypted files were used as part of their test data. The work performed in this paper differs from Li's work in that instead of using n-gram analysis, we will be using the standard Shannon entropy of the file header as the indicator of encryption, as well as having a large variety of encrypted and compressed files in the data set. It was considered to use an improved Shannon entropy based on 2 KB blocks to allow differentiation between compressed and encrypted files [22], but this was rejected due to the small sample window being used. Jung and Won [23] did perform analysis of entropy on file fragments including file header and trailer and while their findings were promising, they had limited their investigation to the PDF document format.

## 2.2. Randomness, Encryption and Ransomware

One of the aims of encryption is to transform useful data into a ciphertext having the property of appearing completely random with no obvious pattern or relationship to the original data [5]. The more random the ciphertext appears, the better the encryption.

Statistical tests can be employed to determine the randomness of a file, with higher randomness suggesting that the contents may be encrypted. Creation of encrypted files

could be the consequence of the action of a ransomware attack so detecting the presence or creation of these files could be used as part of a ransomware detection technique. One drawback with this approach is that the content of some legitimate files such as archive and image files can also appear to be very random.

Generally, the problem of detecting ransomware can be reduced to the problem of detecting random data being written to the file system. Random (or encrypted) data should comprise of an approximately equal number of each byte, unpredictably distributed across the data.

Therefore, it is possible to apply widely-used tests for randomness across these byte distributions to identify the presence of random data. This may then indicate the presence of encryption, and possibly a ransomware attack [40]. Statistical techniques [40] that can be used to identify the randomness of a data stream are shown below:

- **Chi-Square** A test that measures how a model compares to an actual distribution. When considering a test for randomness for Chi-square we would expect an equal frequency of byte values for the expected distribution
- **Arithmetic Mean** This is the sum of all the individual bytes in the sample divided by the total number of bytes in that sample. As all the possible byte values range between zero and 255 then the closer this calculated value is to 127.5 then in theory the more random the data under investigation is.
- **Monte Carlo** This technique is based on the repeated random sampling of the data under investigation and then performing a time averaging statistical analysis on these samples in order to make a decision on the data.
- **Serial Byte Correlation Coefficient** [41] A lightweight statistical test that looks at the relationship between consecutive numbers.
- **Shannon Entropy** [49] In information theory, entropy is a measure of a given input's level of uncertainty and is considered to be an indication of the amount of information contained within each byte.

## 3. Design

The work in this paper could be considered a special case of file fragment analysis where only one fragment is analysed and is limited to the first part of the file. Instead of creating specialised classifications for each file type as proposed by Roussev and Garfinkel [44], or using a generated *fingerprint* [33], the experiments described in this paper will use the Shannon [49] entropy calculation of the file fragment.

Based on the calculated value, the technique will judge if the file is encrypted, where a compressed file will be classed

as an unencrypted file. This approach is supported by, amongst others, Alekseev and Platonov [2] who confirm that testing for entropy is a good indicator for identifying encrypted files.

The aim of the experiments described in this paper is not to accurately identify the specific file type [14, 20, 29, 28, 30, 33] rather it is to identify if the file is encrypted or not.

### 3.1. Data Set

To facilitate the repeatability of the experiments performed in this paper, it was necessary that a well respected, realistic and easily accessible standard data set be used for the experiments [15]. This approach is confirmed by [16, 44] who stress that test data must be representative of data likely to be encountered in real-world situations. It is a known issue within the research community [1, 18, 32, 41] that there is often a lack of readily available, researchable ransomware data sets.

Garfinkel et al. [15] created a publicly available govdocs1 corpus containing one million files collected using random searches of the .gov domain. These files are stored in 1,000 directories, each containing 1,000 files, and also 10 randomly assigned streams for development and testing purposes. This corpus can be obtained from [17]. Many recent digital forensics studies [7, 14, 18, 26, 36, 39, 40, 41] also obtained their test data from Garfinkel et al.'s [15] govdocs1 data set, supporting the claim that this data set is a well-known and respected source of test data.

An advantage of this data set is that it does not contain files that change frequently, such as ransomware or virus samples or files that have been encrypted using these samples. The inclusion of these files would cause the data set to quickly become outdated. Even though the govdocs1 data set is well known and respected, there remain some concerns regarding its use, not least the fact that it is now more than 10 years old. While the govdocs1 corpus is a broad well-known mixed file data set containing more than 55 different file types, the number of some specific file types are not significant. For example, there are only a total of 40 files in the newer Microsoft DOCX and XLSX formats.

A class of files known as archives are also poorly represented. Archive files are normally compressed and are often used to collect multiple files together into a single file for easier portability or to use less storage space. Archive files often store directory structures, error detection, and sometimes use built-in encryption. There are multiple types of archives currently in use, with varying properties and characteristics, however, the only type present in the original data set is *gzip*.

To address the identified shortcomings of the govdocs1 mixed file data set, the following enhancements were performed prior to its use in this research:

- All existing DOC and XLS files present in the govdocs1 corpus are converted to have a corresponding instance in the new Microsoft document format DOCX and XLSX.
- Several sets of various archive types were created. Each set containing 1,000 examples of the specific archive type and used the govdocs1 corpus as their source files. They were created using different compression programs such as tar, 7zip, WinRAR, WinZip, as well as archives using different compression options such as high compression, header compression and encryption. In most cases the tool's default configuration values were used. For WinZip this was the DEFLATE algorithm with one pass. 7zip used the LempelZiv-Markov chain algorithm(LZMA2) with a dictionary size of 16MB and a standard Branch Converter Filter(BCJ). For high compression the LZMA2 algorithm was also used but with a dictionary size of 64MB and a newer version of the standard Branch Converter Filter(BCJ2), producing the tools maximum available compression.
- The govdocs1 files were encrypted by different ransomware samples to produce data sets from each of the selected ransomware strains. Once the ransomware had completed executing, the files affected were gathered into a data set for that ransomware sample. The ransomware strains used during this experiment were: BadRabbit, Netwalker, NotPetya, Phobos, Ryuk, Sodinokibi, WannaCry and WastedLocker.
- A synthetic data set of 1,000 files was created of files with lengths varying between 512 and 2,048 bytes that contained pseudo-random data generated using the Python 'os.urandom' function. When generating random numbers, pseudo-random number generators use mathematical algorithms, whereas true random number generators use unpredictable physical means, for example, atmospheric noise [19]. The purpose of this is to provide a baseline data set that contained pseudo-random data which could be used as a comparison for the encrypted and compressed data files.

### 3.2. Entropy

File entropy refers to a specific measure of randomness. One such measure is called *Shannon Entropy* [49] and is used to express information content. This value is essentially a measure of the predictability of any specific byte in the file, based on preceding bytes [43]. It is basically a measure of the *randomness* of the data in a file - measured in a scale of zero to eight (eight bits in a byte). Typical text files that contain only alphanumeric characters and no formatting will have a low value, whereas encrypted or compressed files will have a high measure [52].

Another way to consider entropy is that it is a measure of the predictability or randomness of data. A file with a highly predictable structure or a bit pattern that repeats frequently has 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

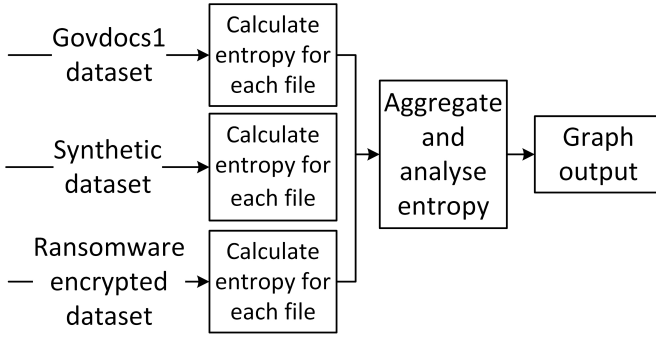


Fig. 1. Experiment Overview.

entropy. Such files would be thought to have high information content [20, 47].

The maximum possible entropy per byte is eight bits of entropy per byte signifying that it is completely random. The generally accepted formula for entropy ( $H$ ) [49], is given as:

$$H(X) = - \sum_{i=1}^n P(x_i) \log_2 P(x_i) \quad (1)$$

where  $H$  is the entropy (measured in bits),  $n$  is the number of bytes in the sample and  $P(x_i)$  is the probability of byte  $i$  appearing in the stream of bytes. The negative sign ensures that the result is always positive or zero.

### 3.3. Experimental design

The experiments described in this paper can be broken down into two distinct areas. Initially, to discover the overall entropy profiles for different file types, the method analyses only the first sequence of bytes of the files under investigation. Once these entropy values have been determined, the second experiment aims to correctly identify encrypted files in a data set.

#### 3.3.1. Entropy profile

The header of each file found in the test data set is processed in turn. Each selected file is analysed 32 times, starting with the first eight bytes of the file header, the first 16, the first 24, and so on, up to the first 256 bytes in eight-byte increments. For each analysis, the entropy of the sampled bytes was calculated and recorded. The averages of the entropy calculations are then grouped by byte length and file type. A graphical representation of this experiment is shown in Figure 1.

A basic outline of the experimental steps performed is illustrated in Algorithm 1.

#### 3.3.2. File classification

In the second experiment, each file is analysed to determine how closely its entropy values match a file that contained purely random numbers. A classification decision is then made based on the similarity of these two.

When applied to file fragments, the resulting calculated Shannon entropy value for small sample sizes is typically rel-

#### Algorithm 1: Header Entropy Test Procedure

```

function EntropyPlot (files);
Input : List of files (files) to analyse
Output: list of arrays contain x,y plot coordinates
plot_coord = ();
foreach file_type of files do
  foreach f of file_type do
    buff = open(f);
    for len = 8; len <= 256; len = len + 8 do
      fragment = buff[len];
      // entropy of fragment
      file_plot[len] = log2(255 × fragment);
    close(f);
  for len = 8; len <= 256; len = len + 8 do
    plot_coord[file_type][len] = average(
      file_plot[len]);
  file_plot = [];
return plot_coord;
    
```

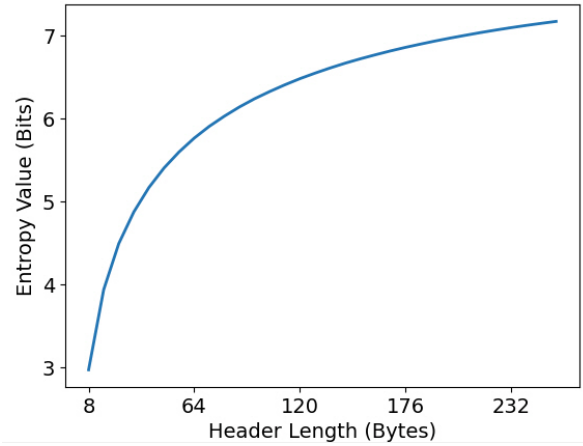


Fig. 2. Entropy value plot for a file containing purely random data.

atively low and increases as the sample size increases. This is illustrated in Figure 2 which shows a plot of the Shannon entropy values for different sample sizes taken from a file that contains random numbers. The byte length of the file fragment analysed is shown in bytes along the x-axis. This plot represents an ideal curve for random data, but it may be closely approximated to encrypted data. As previously stated, encrypted data share similar Shannon entropy ranges to random data.

One method that can be used to quantify the closeness to random samples would be to compare it to a plot generated by a reference file that contains purely random numbers. While there are several possible techniques available to analyse the variations, such as the Chi-square [38] method, the technique selected for these experiments determines the differential area between the sampled target file and a reference sample of random data. This approach is selected over just calculating the difference between the value at each point, as the area calculation takes into account how close the whole

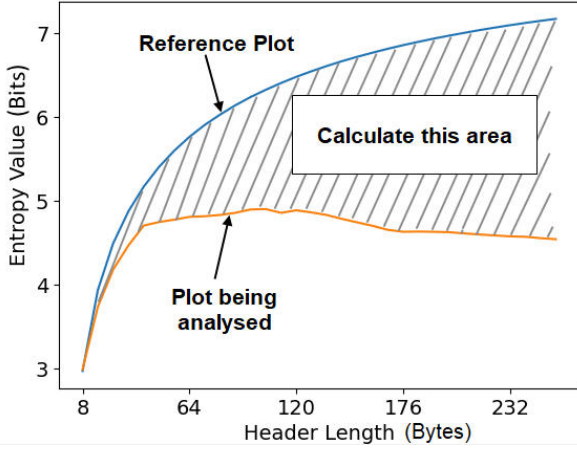


Fig. 3. Plot differences.

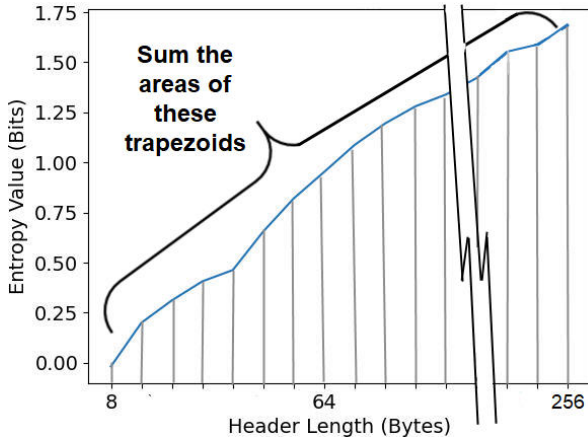


Fig. 4. Trapezoidal area of the derived plot.

curve matches over the complete interval and not just at a specific sampling interval.

The values on the y-axis are given in bits and the values on the x-axis are given in bytes, resulting in the area unit being Bit-Bytes. This calculated *Bit-Byte* area value is then used to determine if the file represented by the entropy target file contains encrypted data or not. The lower the area value, the closer the sampled target file matches the randomised data source, the more likely it is that the file contains random or encrypted data. A graphical representation of this analysis is shown in Figure 3.

This area calculation can be achieved in reality by generating a new third plot. The values of which are derived by subtracting the plot under analysis values from the random data plot values. The area between the x-axis and this new third plot can then be calculated. A graphical representation of this calculation is shown in Figure 4.

The newly created plot is not a precise curve, rather it is a series of straight lines joining each sample point. Due to this property and using an equal step size, the optimum method for calculating the area under the plot is to use the

*Composite Trapezoidal Rule* [4] given as:

$$I_{composite} = \frac{h}{2} \left[ f(a) + 2 \sum f(x + h_i) + f(b) \right] \quad (2)$$

where  $a$  is the start point,  $b$  is the endpoint and  $h$  is the step value between the plot points.

The closer the plot under analysis is to the reference plot, the lower the calculated area value will be and the higher the confidence is that the file contains random or encrypted data. A basic outline of the experimental steps performed is illustrated in Algorithm 2.

---

**Algorithm 2:** File Classification
 

---

```

function FileClassification (files);
Input : List of files (files) to analyse
Output: list of pass/fail statistics for each header
        length an file type
// Shannon values for random data plot
rand_plot = [x y] ;
stats = ();
foreach threshold=start; threshold < max;
    threshold++ do
        foreach file_type of files do
            foreach f of file_type do
                buff = open(f);
                for len=8; len<=256; len=len+8 do
                    fragment= buff[len];
                    // entropy of fragment
                    e = log2(255 × fragment);
                    // subtract from random plot
                    plot[len] = rand_plot[len] - e;
                close(f);
                area = area(plot);
                if area<=threshold then
                    stats(file_type)(threshold)(pass)
                        +=1 ;
                else
                    stats(file_type)(threshold)(fail)
                        +=1;
            end
        end
    end
return stats;
    
```

---

When classifying the file types there are four possible outcomes. Possible classifications are shown in Table 1. The proposed model uses these classification values to determine the overall accuracy of the model [50]. The formula to compute the Accuracy(*ACC*) is given as:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

### 3.3.3. File classification example

To illustrate the proposed method in more detail, the following example will be used. In this example, we will be analysing entropy plots for header lengths of 40 bytes and

**Table 1.** Possible classification outcomes

Classification	Description
True Positive (TP)	Correctly classified encrypted file
True Negative (TN)	Correctly classified non-encrypted file
False Positive (FP)	Classified normal file as an encrypted file
False Negative (FN)	Failed to classified encrypted file

**Table 2.** Calculation of the derived plot entropy values

Header length	8	16	24	32	40
Random plot (y)	2.976	3.941	4.496	4.878	5.171
Sample plot (y)	3.000	3.807	3.336	3.430	3.903
Derived plot (y)	-0.024	0.134	1.160	1.448	1.268

**Table 3.** Calculated area for each trapezoid

Header length	16	24	32	40
Trapezoid Area	0.44	5.176	10.432	10.864

assuming that the variable threshold is 20 Bit-Bytes. If the calculated area is below the threshold value, it is assessed to be an encrypted file. As a sample for analysis, entropy values are taken from a file generated using the 7Zip program, new plot points for a derived graph are calculated by subtracting the values for the graph under analysis from the points that describe a file that contains purely random data. The values and the results from this analysis is shown in Table 2.

The sum of the areas of each trapezoid of the new plot is then determined using equation 2 where,  $a=8$ ,  $b=40$  and  $h=8$ . For clarity, the individual trapezoidal areas are shown in Table 3.

The calculated area in this example gives a final value of **26.91** Bit-Bytes. This value is then compared to the threshold value and a classification decision is made. In this case, the value is greater than the threshold, so this file is correctly classified as a non-encrypted file, also known as True-Negative (TN) classification. If the area had been below the threshold of 20 Bit-Bytes, it would have been incorrectly classified as an encrypted file and would have been a False-Positive (FP) classification.

## 4. Implementation

A subset of files taken from the first 100 archives of the complete govdocs1 [15] archives were initially selected as the input for these experiments together with the additional data sets for XLSX/DOCX, compressed files and ransomware encrypted files mentioned in Section 3.1. When generating the data sets relating to specific ransomware samples, a set of govdocs1 files were placed on a completely isolated target machine and the ransomware sample executed, paying particular attention to the ethical consequences of running such programs. Once the ransomware program execution had completed, the files placed on the machine were examined to determine if they had been updated by the ran-

**Table 4.** Excluded file entropy values for the first 40 bytes.

File Type	Sample Size	Entropy	File Type	Sample Size	Entropy
csv	1,032	3.849	ppt	5,642	2.267
data	2	2.374	sgml	9	3.765
dbase3	171	2.256	sql	7	3.493
doc	8,025	2.266	text	188	3.176
f	130	3.704	tmp	4	2.940
fits	39	2.246	troff	21	3.964
gif	2,026	3.551	ttf	1	2.146
hlp	7	2.805	txt	15,008	3.496
java	36	3.993	unk	434	2.864
jpg	7,167	3.793	wp	42	3.142
png	317	3.753	xls	4,300	2.280
pps	67	2.267			

somware's execution. If they had been changed, they were then added to the data set for that ransomware sample. Different ransomware strains can target different file types, which can result in different data set sizes for different ransomware strains.

Programs written in Python were created to perform the experiment steps outlined in algorithms 1 and 2. The entropy calculation algorithm used was adapted from the code developed for the 'findaes' [27] and 'interrogate' [31] programs, and are based on the work by Trenholme [51]. Some additional validation of the results was also performed by running the same calculations via the website [3].

Many of the file types present in the original data set had a low value calculated for the entropy of their headers. Low entropy files would be trivial to correctly classify using the proposed technique and could bias the experimental results. For this reason, file types in the original data set with a header entropy below 4.0 were excluded from the final analysis. Details of excluded file types with low header entropy are given in Table 4. The ransomware samples used are all examples of recent high impact strains and are defined in Table 5. VirusTotal [53] was used to confirm that they were valid ransomware programs.

Programs used to build data sets containing compressed files were WinZip Version 25, WinRAR Version 3.42 and 7Zip Version 9.2. The resulting test data set contained a total of 84,308 files. 8,685 or 10% of the data set were ransomware encrypted files and the remainder were regular or compressed archive files.

## 5. Results and Discussion

Individual entropy values were calculated for each file in the data set. The average entropy was then calculated for each of the file types for header lengths from eight to 256 bytes and the resulting plots for some of these are shown in Figure 5. An example of the data recorded for each file type when examining the first 40 bytes is shown in Table 6.

It can be seen from the entropy graph that as the sample size used for the calculation increases, the entropy values for the compressed files become similar to the entropy

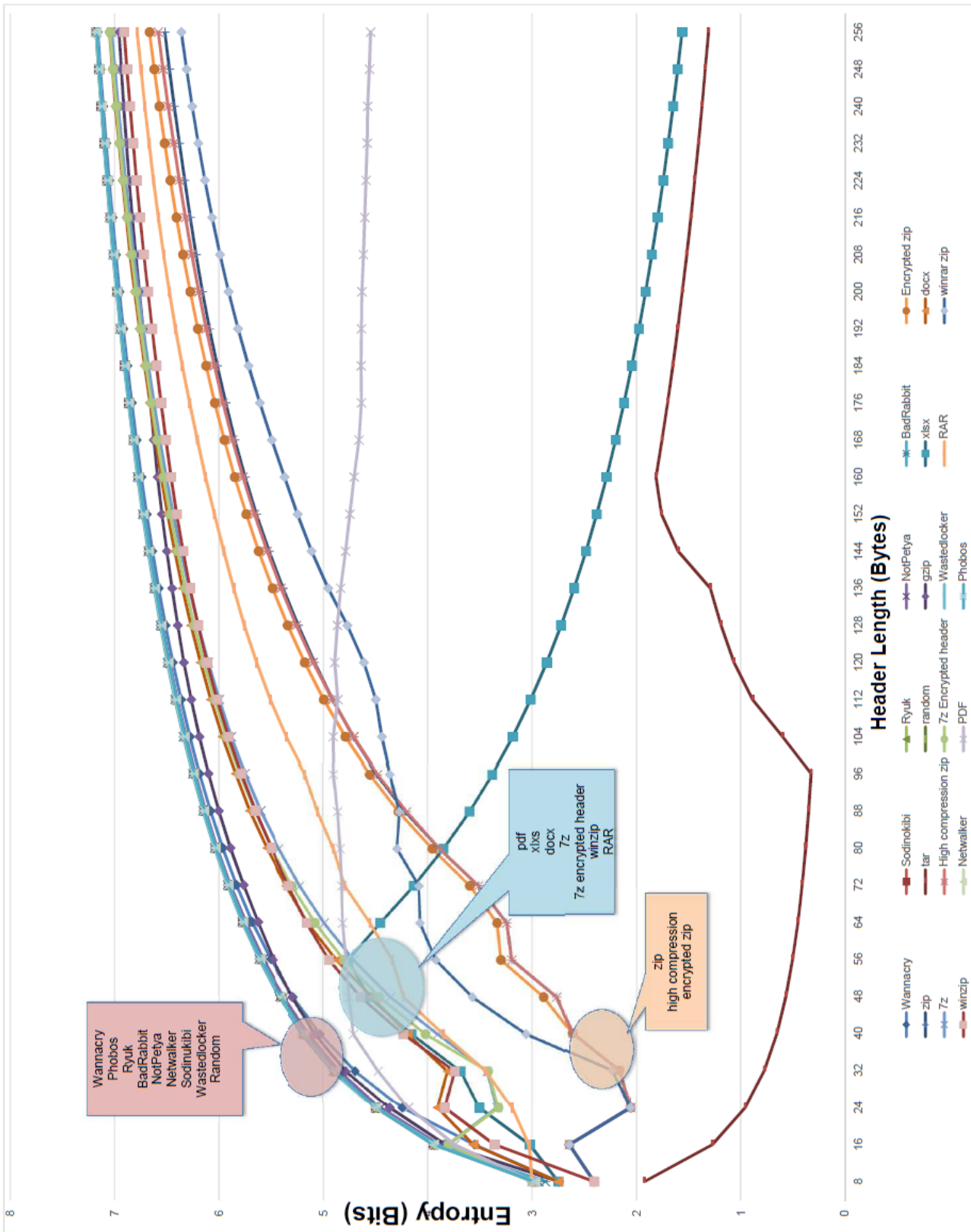


Fig. 5: Entropy Values for Different File Types.



**Table 5.** Investigated ransomware samples

Name	SHA256 Hash Value
BadRabbit	630325cac09ac3fab908f903e3b00d0dadd5fdaa0875ed8496fcbb97a558d0da
NotPetya	027cc450ef5f8c5f653329641ec1fed91f694e0d229928963b30f6b0d7d3a745
Phobos	a91491f45b851a07f91ba5a200967921bf796d38677786de51a4a8fe5ddeafd2
Netwalker	57cf4470348e3b5da0fa3152be84a81a5e2ce5d794976387be290f528fa419fd
Ryuk	d083ecc1195602c45d9cb75a08c395ad7d2b0bf73d7e70e2fc76101c780cc38f
Sodinokibi	96dde0a25cc6ca81a6d3d5025a36827b598d94f0fca6ab0363bfc893706f2e87
Wannacry	32f24601153be0885f11d62e0a8a2f0280a2034fc981d8184180c5d3b1b9e8cf
Wastedlocker	905ea119ad8d3e54cd228c458a1b5681abc1f35df782977a23812ec4efa0288a

**Table 6.** File entropy values for the first 40 bytes of the file.

File type	Sample Size	Entropy	$\sigma$
Ryuk	969	5.175	0.0785
WastedLocker	979	5.170	0.0809
Sodinokibi	971	5.162	0.0808
BadRabbit	556	5.111	0.0736
Netwalker	3,110	5.176	0.0798
Phobos	889	5.170	0.0804
gzip	1,000	5.075	0.0891
NotPetya	457	5.052	0.0756
Wannacry	754	5.025	0.0799
PDF	25,282	4.708	0.1629
KML	32	4.645	0.0851
PS	1,212	4.616	0.1432
XML	932	4.605	0.2322
DWF	5	4.547	0.1713
EPS	70	4.433	0.1582
SWF	43	4.303	0.5957
winzip	1,000	4.232	0.1256
KMZ	28	4.221	0.4118
DOCX	8,038	4.190	0.0427
TEX	36	4.157	0.3286
XLSX	4,284	4.152	0.0437
PPTX	21	4.136	0.0470
RTF	130	4.096	0.0944
HTML	25,510	4.058	0.6840
7z Encrypted header	1,000	4.020	0.0895
7z	1,000	3.903	0.0742
RAR	1,000	3.847	0.1228
winrar zip	1,000	3.058	0.0118
ZIP	1,000	2.612	0.0271
High compression zip	1,000	2.612	0.0268
Encrypted zip	1,000	2.611	0.0266
TAR	1,000	0.654	0.0245

values of the encrypted files. However, we believe it is significant that while the initial entropy of the encrypted files is relatively high from the beginning, the files that have been compressed have a much lower entropy value at the start of the files. Other researchers have also noted [20, 30] that for some file formats, the entropy of the file fragments changes throughout the file. Some explanations for this could be related to the file's magic number, compression techniques being used and in the use of Huffman tables and other metadata at the start of these files contributing to these fragments having a lower entropy than the remainder of the file. The only exception to this pattern being files created with the *gzip* pro-

gram despite this method also using the LZ77 compression algorithm and Huffman tables, similar to the other compression programs tested [12]. Due to this the dataset for *gzip* files were not analysed during the classification stage.

The one anomaly to this pattern relates to the values recorded for the Phobos strain of ransomware. The relatively low average entropy values for files encrypted with this strain can be explained by the fact that this ransomware sometimes creates files that have large blocks of zeros in the file's header. This has the effect of lowering the average entropy for that file header and the general average for this file type. The entropy calculation algorithm used in the classification was modified to identify these large blocks of zeros and subsequently ignore them when calculating the file header entropy. After implementing this modification the entropy plot for this strain then closely followed the other ransomware samples by also exhibiting higher initial entropy values.

When examining smaller fragments of the file headers it was observed that there was a measurable average entropy difference between encrypted files having a value of five and compressed files having a value of four.

Previous researchers have commented [39] on the difficulty in discerning the differences in file types of high entropy with little pattern, if any, within the data. Noting that compression algorithms are often optimised for speed, they suggest that it may be possible to further compress already compressed data. As such, encrypted data may be more random than compressed data and, thus, compressibility and randomness may enable these file types to be distinguished from one another. Using the above approach of reducing the sample size used for the analysis of these files and restricting it to the beginning of the file, may enhance the accuracy of identification of the type file being analysed.

An example of the values recorded is given in Table 6 and illustrates the entropy values of the examined files when analysing the first 40 bytes of the file's header. Differentiation between compressed and encrypted file types is achievable for most ransomware encrypted files as they tend to have an entropy value of at least unity greater than file archives with the exception of the *gzip* format. The majority of other common file types have significantly lower entropy than compressed files, so these would also be easily distinguishable using this technique.

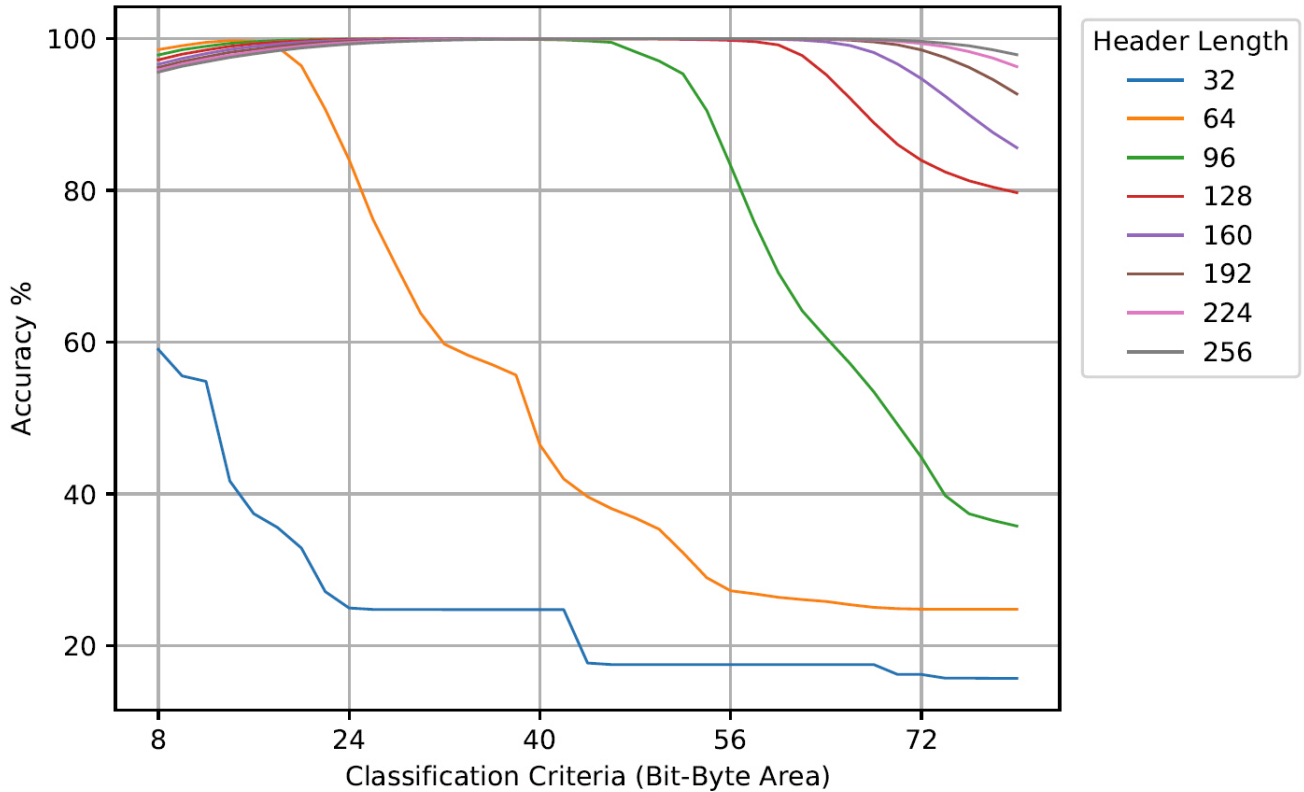


Fig. 6. Overall Classification Accuracy.

Table 7. Classification Accuracy Results (%)

		Classification Criteria (Bit-Bytes)				
		8	24	40	56	72
Header Length (Bytes)	32	63.290	20.097	19.439	14.061	11.767
	64	99.010	87.851	46.472	24.227	19.879
	96	98.546	99.914	99.842	87.077	51.326
	128	98.118	99.903	99.944	99.794	89.289
	160	96.736	99.825	99.960	99.934	96.439
	192	97.452	99.744	99.957	99.952	98.959
	224	97.234	99.631	99.954	99.957	99.517
	256	97.055	99.505	99.944	99.962	99.713

### 5.1. Classification model validation

The outcome from the second experiment was also successful in identifying encrypted files amongst other files of high entropy and the results are presented in Figure 6 with the most interesting portion being highlighted in Figure 7. It can be seen that for header lengths between 128 and 256 bytes and, a classification value of between 32 and 56 Bit-Bytes, high rates of success for ransomware encrypted files classification were achieved. Table 7 shows that in some cases a success classification rate of greater than 99.96% was achieved with the highest accuracy being highlighted in grey. Related calculated values for this high accuracy region are also provided, with Precision being shown in Table 8, Recall in Table 9 and *F1* in Table 10

Errors in classification occurred in circumstances where normal files that had a high header entropy were classified

Table 8. Classification Precision Results (%)

		Classification Criteria (Bit-Bytes)			
		24	40	56	72
Header Length (Bytes)	96	99.369	98.503	44.651	17.640
	128	100	99.462	98.058	49.324
	160	100	99.656	99.371	74.537
	192	100	100	99.553	90.923
	224	100	100	99.621	95.596
	256	100	100	99.690	97.343

as ransomware encrypted (False Positive) and where files resulting from ransomware encryption that had a low header entropy were classified as normal files (False Negative). For example, for a header length of 256 bytes and a classification criteria of 56 Bit-Bytes there were 32 classification errors. Details of which are show in Table 11.

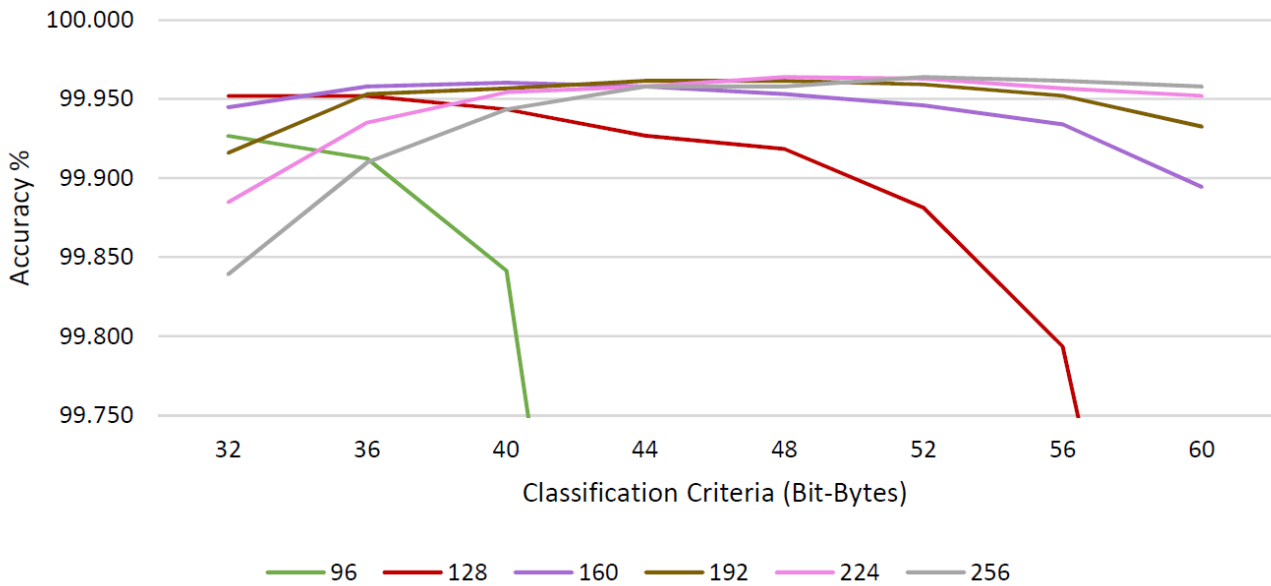


Fig. 7. Enlarged section of classification accuracy.

Table 9. Classification Recall Results (%)

		Classification Criteria (Bit-Bytes)			
		24	40	56	72
Header Length (Bytes)	96	99.804	100	100	100
	128	99.286	100	100	100
	160	98.480	99.965	100	100
	192	97.640	99.850	100	100
	224	96.546	99.793	99.965	99.977
	256	95.337	99.632	99.942	99.977

Table 10. Classification F1 Results (%)

		Classification Criteria (Bit-Bytes)			
		24	40	56	72
Header Length (Bytes)	96	99.586	99.246	61.736	29.990
	128	99.533	99.730	99.019	66.063
	160	99.154	99.810	99.684	85.411
	192	98.760	99.793	99.770	95.246
	224	98.202	99.781	99.793	97.738
	256	97.572	99.729	99.816	98.642

Table 11. Example of Classification Errors

Classification	Errors	File types
False Positives	27	PDF(1) PS(14) SWF(12)
False Negatives	5	BadRabbit(1) Phobos(2) Sodinokibi(1) WasterLocker(1)

Finally, a performance comparison test was conducted to determine the difference in execution time between file header and full file entropy calculations. A sample set of 200 PDF files with sizes varying from 2 KB to 16 MB was used. A performance improvement of three orders of magnitude was achieved by restricting the entropy calculation to the file's header as opposed to calculating the entropy of the entire file.

## 6. Conclusion

The findings from these experiments support the hypothesis that it is possible to classify the type of file being created by only analysing the first few bytes of that file's header. The results from the first experiment demonstrated that, on average, there is a noticeable entropy difference at the beginning of the file between most files and encrypted files generated by ransomware. The paper also introduces the concept of Bit-Byte area and how it can be used as a criteria on which to base file classification decisions. The second experiment highlighting the high success rates achieved when using these criteria. The model was tested against a mixed data set developed by the researchers that, while based on the govdocs1 corpus, also included additional files that more reflect the types of files found on modern systems such as archives. These additional files are included in the data set as they have similar attributes, and were used to determine if the model was able to identify the encrypted files. The results from the experiments confirming that the model was sufficiently robust enough to cope successfully with these more difficult file classifications. In some cases achieving a classification accuracy of greater than 99.96%. While [11, 40] claim that entropy is not a good measure of encryption, these researchers have only considered the file's entire entropy and not a file fragment as we have done in these experiments.

The proposed technique is also ransomware agnostic as no knowledge of the specific ransomware strain is required. The technique is also resilient to the simple obfuscation techniques [35] of altering magic numbers and file extensions. Finally, as only a relatively small portion of the file is being analysed, successful classification is achieved in a fixed amount of time regardless of file size. Consequently reducing the performance impact of such a check and achieving Continella et al.'s goal of minimising the time to decision [8].

As the specific type of file being created is not the focus of the technique, rather only trying to determine the probability that the file is a result of a ransomware encryption event or not, this technique could possibly be incorporated into a ransomware detection system that intercepts the file writing process. Then, based on the results of the analysis a decision could be made on whether to allow the file write operation or not. If it is determined that the file may be being created by a ransomware process, further analysis of the process creating the file could be performed, possibly using live forensics to discover the encryption keys being used [10].

### 6.1. Further research

Some further areas of research could be to investigate why files generated using the *gzip* program has a higher initial entropy value than other compression programs using the LZ77 algorithm and Huffman tables. Investigation into the behaviour of the Phobos ransomware strain may prove beneficial in discovering why it places large blocks of zeros at the beginning of some of the files it creates.

While in these experiments, Shannon entropy was used, it is thought that in the future other calculations maybe be researched. These could include Hamming distance [21], NIST statistical test suit [45], Higuchi fractal dimension [24], n-grams [30], byte frequency analysis [44] or techniques based on machine learning.

## References

- [1] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions," *Computers and Security*, vol. 74, pp. 144–166, 2018. doi: 10.1016/j.cose.2018.01.001. <https://doi.org/10.1016/j.cose.2018.01.001>
- [2] I. V. Alekseev and V. V. Platonov, "Detection of encrypted executable files based on entropy analysis to determine the randomness measure of byte sequences," *Automatic Control and Computer Sciences*, vol. 51, no. 8, pp. 915–920, 2017. doi: 10.3103/S0146411617080041. <https://dx.doi.org/10.3103/S0146411617080041>
- [3] Asecuritysite, "AsecuritySite," 2020. <https://asecuritysite.com/encryption/ent> (Last Accessed 2020-09-14).
- [4] K. E. Atkinson, *An Introduction to Numerical Analysis (2nd ed.)*. New York: John Wiley & Sons, Inc, 1989.
- [5] J. Aumasson, *Serious Cryptography*. No Starch Press, 2018.
- [6] P. Bajpai and R. Endoby, "An Empirical Study of Key Generation in Cryptographic Ransomware," in *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2020. doi: 10.1109/CyberSecurity49315.2020.9138878. <https://dx.doi.org/10.1109/CyberSecurity49315.2020.9138878>
- [7] G. Cleary, "Digital Evidence Detection using Bitwise Approximate Matching Gabrielle Cleary RD5 Report Edinburgh Napier University School of Computing," Edinburgh Napier University, Tech. Rep. January, 2018.
- [8] A. Continella, A. Guagnelli, G. Zingaro, G. De Pasquale, A. Barengi, S. Zanero, and F. Maggi, "ShieldFS: A self-healing, ransomware-aware file system," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016. doi: 10.1145/2991079.2991110 pp. 336–347. <https://dl.acm.org/doi/10.1145/2991079.2991110>
- [9] Coveware, "Q2 2020 Ransomware Marketplace Report," 2020. <https://www.coveware.com/blog/q2-2020-ransomware-marketplace-report> (Last Accessed 2020-08-03).
- [10] S. R. Davies, R. Macfarlane, and W. J. Buchanan, "Evaluation of live forensic techniques in ransomware attack mitigation," *Forensic Science International: Digital Investigation*, vol. 33, 2020. doi: 10.1016/j.fsidi.2020.300979. <https://doi.org/10.1016/j.fsidi.2020.300979>
- [11] F. De Gaspari, D. Hitaj, G. Pagnotta, L. De Carli, and L. V. Mancini, "EnCoD: Distinguishing Compressed and Encrypted File Fragments," in *14th International Conference, Network and System Security*, 2020. doi: 10.1007/978-3-030-65745-1\_3 pp. 1–19. [https://dx.doi.org/10.1007/978-3-030-65745-1\\_3](https://dx.doi.org/10.1007/978-3-030-65745-1_3)
- [12] P. Deutsch, "RFC 1952 - GZIP File Format Specification," 1996. <https://doi.org/10.17487/RFC1952>
- [13] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Computing Surveys*, vol. 44, no. 2, 2012. doi: 10.1145/2089125.2089126. <https://doi.org/10.1145/2089125.2089126>
- [14] S. Fitzgerald, G. Mathews, C. Morris, and O. Zhulyn, "Using NLP techniques for file fragment classification," *Digital Investigation*, vol. 9, no. SUPPL., pp. 44–49, 2012. doi: 10.1016/j.diin.2012.05.008. <https://doi.org/10.1016/j.diin.2012.05.008>
- [15] S. Garfinkel, P. Farrell, V. Rousev, and G. Dinolt, "Bringing science to digital forensics with standardized forensic corpora," *Digital Investigation*, vol. 6, no. SUPPL., pp. 2–11, 2009. doi: 10.1016/j.diin.2009.06.016. <http://dx.doi.org/10.1016/j.diin.2009.06.016>
- [16] S. L. Garfinkel, "Digital forensics research: The next 10 years," *Digital Investigation*, vol. 7, no. SUPPL., pp. 64–73, 2010. doi: 10.1016/j.diin.2010.05.009. <http://dx.doi.org/10.1016/j.diin.2010.05.009>
- [17] —, "Govdocs1," 2020. <http://downloads.digitalcorpora.org/corpora/files/govdocs1/> (Last Accessed 2020-09-14).
- [18] C. Grajeda, F. Breiting, and I. Baggili, "Availability of datasets for digital forensics And what is missing," *Digital Investigation*, vol. 22, pp. S94–S105, 2017. doi: 10.1016/j.diin.2017.06.004. <https://dx.doi.org/10.1016/j.diin.2017.06.004>
- [19] M. Haar, "RANDOM.ORG - True Random Number Service," 1998. <https://www.random.org/> (Last Accessed 2020-10-14).
- [20] G. A. Hall and W. P. Davis, "Sliding Window Measurement for File Type Identification," 2006. doi: 10.1.1.113.8439. <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.113.8439&rep=rep1&type=pdf>
- [21] R. Hamming, "Error Detecting and Error Correcting Codes," *The Bell System Technical Journal*, no. 2, pp. 147–160, 1950. doi: 10.1002/j.1538-7305.1950.tb00463.x. <https://dx.doi.org/10.1002/j.1538-7305.1950.tb00463.x>
- [22] M. Held, "Detecting Ransomware," Masters Thesis, University Konstanz, 2018. <https://kops.uni-konstanz.de/handle/123456789/43396>
- [23] S. Jung and Y. Won, "Ransomware detection method based on context-aware entropy analysis," *Soft Computing*, vol. 22, no. 20, pp. 6731–6740, 2018. doi: 10.1007/s00500-018-3257-z. <https://doi.org/10.1007/s00500-018-3257-z>
- [24] S. Kesić and S. Z. Spasić, "Application of Higuchi's fractal dimension from basic to clinical neurophysiology: A review," *Computer Methods and Programs in Biomedicine*, vol. 133, no. January, pp. 55–70, 2016. doi: 10.1016/j.cmpb.2016.05.014. <https://doi.org/10.1016/j.cmpb.2016.05.014>
- [25] A. Kharraz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda, "UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware," in *Usenix 25th USENIX Security Symposium*, Austin, TX, 2016. doi: 10.1109/SANER.2017.7884603 pp. 757–772. <https://doi.ieeecomputersociety.org/10.1109/SANER.2017.7884603>
- [26] E. Kolodenker, W. Koch, G. Stringhini, and M. Egele, "Pay-Break : Defense against cryptographic ransomware," *ASIA CCS 2017 - Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security*, pp. 599–611, 2017. doi: 10.1145/3052973.3053035. 10.1145/3052973.3053035
- [27] J. Kornblum, "findaes," 2017. <http://jessekornblum.com/tools/findaes/> (Last Accessed 2019-08-10).

- [28] K. Lee, S. Y. Lee, and K. Yim, "Effective Ransomware Detection Using Entropy Estimation of Files for Cloud Services," in *I-SPAN 2019: Pervasive Systems, Algorithms and Networks*, vol. 1080 CCIS. Springer International Publishing, 2019. doi: 10.1007/978-3-030-30143-9\_11 pp. 133–139. [http://dx.doi.org/10.1007/978-3-030-30143-9\\_11](http://dx.doi.org/10.1007/978-3-030-30143-9_11)
- [29] K. Lee, S.-Y. Lee, and K. Yim, "Machine Learning Based File Entropy Analysis for Ransomware Detection in Backup Systems," *IEEE Access*, vol. 7, pp. 110205–110215, 2019. doi: 10.1109/access.2019.2931136. <https://dx.doi.org/10.1109/ACCESS.2019.2931136>
- [30] W. J. Li, K. Wang, S. J. Stolfo, and B. Herzog, "Fileprints: Identifying file types by n-gram analysis," *Proceedings from the 6th Annual IEEE System, Man and Cybernetics Information Assurance Workshop, SMC 2005*, vol. 2005, pp. 64–71, 2005. doi: 10.1109/IAW.2005.1495935. <https://dx.doi.org/10.1109/IAW.2005.1495935>
- [31] C. Maartmann-Moe, S. E. Thorkildsen, and A. Årnes, "The persistence of memory: Forensic identification and extraction of cryptographic keys," *Digital Investigation*, vol. 6, pp. 132–140, 2009. doi: 10.1016/j.diin.2009.06.002. <https://doi.org/10.1016/j.diin.2009.06.002>
- [32] A. M. Maigida, S. M. Abdulhamid, M. Olalere, J. K. Alhassan, H. Chiroma, and E. G. Dada, "Systematic literature review and metadata analysis of ransomware attacks and detection mechanisms," *Journal of Reliable Intelligent Environments*, vol. 5, no. 2, pp. 67–89, 2019. doi: 10.1007/s40860-019-00080-3. <https://doi.org/10.1007/s40860-019-00080-3>
- [33] M. McDaniel and M. H. Heydari, "Content based file type detection algorithms," *Proceedings of the 36th Annual Hawaii International Conference on System Sciences, HICSS 2003*, p. 10, 2003. doi: 10.1109/HICSS.2003.1174905. <https://dx.doi.org/10.1109/HICSS.2003.1174905>
- [34] T. McIntosh, J. Jang-Jaccard, and P. Watters, "Large Scale Behavioral Analysis of Ransomware Attacks," in *International Conference on Neural Information Processing*, vol. 2. Springer International Publishing, 2018. doi: 10.1007/978-3-030-04224-0\_19 pp. 217–229. [http://dx.doi.org/10.1007/978-3-030-04224-0\\_19](http://dx.doi.org/10.1007/978-3-030-04224-0_19)
- [35] T. McIntosh, J. Jang-Jaccard, P. Watters, and T. Susnjak, "The Inadequacy of Entropy-Based Ransomware Detection," in *International Conference on Neural Information Processing*, vol. 1. Springer International Publishing, 2019. doi: 10.1007/978-3-030-36802-9\_20 pp. 181–189. [https://dx.doi.org/10.1007/978-3-030-36802-9\\_20](https://dx.doi.org/10.1007/978-3-030-36802-9_20)
- [36] K. Nguyen, D. Tran, W. Ma, and D. Sharma, "A proposed approach to compound file fragment identification," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8792, pp. 493–500, 2014. doi: 10.1007/978-3-319-11698-3\_38. [https://doi.org/10.1007/978-3-319-11698-3\\_38](https://doi.org/10.1007/978-3-319-11698-3_38)
- [37] L. O'Donnell, "University of Utah Pays 457K After Ransomware Attack." 2020. <https://threatpost.com/university-of-utah-pays-457k-after-ransomware-attack/158564/> (Last Accessed 2020-08-31).
- [38] K. Pearson, "X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, 1900. doi: 10.1080/14786440009463897. <https://doi.org/10.1080/14786440009463897>
- [39] P. Penrose, R. MacFarlane, and W. J. Buchanan, "Approaches to the classification of high entropy file fragments," *Digital Investigation*, vol. 10, no. 4, pp. 372–384, 2013. doi: 10.1016/j.diin.2013.08.004. <http://dx.doi.org/10.1016/j.diin.2013.08.004>
- [40] J. Pont and J. Hernandez-Castro, "Why Current Statistical Approaches to Ransomware Detection Fail," in *International Conference on Information Security*. 23rd Information Security Conference., 2020. doi: 10.1007/978-3-030-62974-8\_12 pp. 199–218. [https://dx.doi.org/10.1007/978-3-030-62974-8\\_12](https://dx.doi.org/10.1007/978-3-030-62974-8_12)
- [41] J. Pont, O. Abu Oun, C. Brierley, B. Arief, and J. Hernandez-Castro, "A Roadmap for Improving the Impact of Anti-ransomware Research," in *Nordic Conference on Secure IT Systems*. Springer International Publishing, 2019. doi: 10.1007/978-3-030-35055-0\_9 pp. 137–154. [http://dx.doi.org/10.1007/978-3-030-35055-0\\_9](http://dx.doi.org/10.1007/978-3-030-35055-0_9)
- [42] J. Porter, "Garmin reportedly paid multimillion-dollar ransom after suffering cyberattack," 2020. <https://www.theverge.com/2020/8/4/21353842/garmin-ransomware-attack-wearables-wastedlocker-evil-corp> (Last Accessed 2020-08-04).
- [43] Rosetta, "Entropy," 2020. <http://rosettacode.org/wiki/Entropy> (Last Accessed 2020-09-14).
- [44] V. Roussev and S. L. Garfinkel, "File fragment classification - The case for specialized approaches," *4th International Workshop on Systematic Approaches to Digital Forensic Engineering, SADFE 2009*, pp. 3–14, 2009. doi: 10.1109/SADFE.2009.21. <https://dx.doi.org/10.1109/SADFE.2009.21>
- [45] A. Rukhin, J. Soto, and J. Nechvatal, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," *Nist Special Publication*, vol. 22, no. April, pp. 1/1—G/1, 2010. <http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22rev1a.pdf>
- [46] N. Scaife, H. Carter, P. Traynor, and K. R. Butler, "CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data," *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pp. 303–312, 2016. doi: 10.1109/ICDCS.2016.46. <https://doi.org/10.1109/ICDCS.2016.46>
- [47] B. Schneier, *Applied Cryptograph, Second Edition: protocols, Algorithms and source Code in C*. John Wiley & Sons, Inc, 1996.
- [48] T. Seals, "The REvil ransomware and savvy phone scammers have exposed sensitive information." 2020. <https://threatpost.com/jack-daniels-ritz-london-cyberattacks/158409/> (Last Accessed 2020-08-31).
- [49] C. Shannon, "A Mathematical Theory of Communication," *Bell System Technology*, vol. 27, no. 1, pp. 379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x. <https://dx.doi.org/10.1002/j.1538-7305.1948.tb01338.x>
- [50] K. M. Ting, *Confusion Matrix*. Boston, MA: Springer US, 2017, p. 260. [https://doi.org/10.1007/978-1-4899-7687-1\\_50](https://doi.org/10.1007/978-1-4899-7687-1_50)
- [51] S. Trenholme, "The AES encryption algorithm," 2014. <https://www.samiam.org/rijndael.html> (Last Accessed 2020-09-01).
- [52] R. VandenBrink, "Using File Entropy to Identify 'Ransomware' Files," 2016. <https://isc.sans.edu/forums/diary/Using+File+Entropy+to+Identify+Ransomware+Files/21351/> (Last Accessed 2020-09-05).
- [53] VirusTotal, "VirusTotal," 2020. [www.VirusTotal.com](http://www.VirusTotal.com) (Last Accessed 2020-09-12).
- [54] A. Young and M. Yung, "Cryptovirology: extortion-based security threats and countermeasures," in *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, 1996. doi: 10.1109/SECPRI.1996.502676 pp. 129–140. <https://doi.org/10.1109/SECPRI.1996.502676>
- [55] A. L. Young and M. Yung, "Cryptovirology: The birth, neglect, and explosion of ransomware: Recent attacks exploiting a known vulnerability continue a downward spiral of ransomware-related incidents," *Communications of the ACM*, vol. 60, no. 7, pp. 24–26, 2017. doi: 10.1145/3097347. <https://doi.org/10.1145/3097347>
- [56] B. Zhao, Q. Liu, and X. Liu, "Evaluation of encrypted data identification methods based on randomness test," *Proceedings - 2011 IEEE/ACM International Conference on Green Computing and Communications, GreenCom 2011*, pp. 200–205, 2011. doi: 10.1109/GreenCom.2011.41. <https://doi.org/10.1109/GreenCom.2011.41>