Why Current Statistical Approaches to Ransomware Detection Fail

Jamie Pont[0000−0003−0969−2464], Budi Arief[0000−0002−1830−1587], and Julio Hernandez-Castro[0000−0002−6432−5328]

University of Kent, Canterbury, UK
{jjp31, ba284, jch27}@kent.ac.uk

Abstract. The frequent use of basic statistical techniques to detect ransomware is a popular and intuitive strategy; statistical tests can be used to identify randomness, which in turn can indicate the presence of encryption and, by extension, a ransomware attack. However, common file formats such as images and compressed data can look random from the perspective of some of these tests. In this work, we investigate the current frequent use of statistical tests in the context of ransomware detection, primarily focusing on false positive rates. The main aim of our work is to show that the current over-dependence on simple statistical tests within anti-ransomware tools can cause serious issues with the reliability and consistency of ransomware detection in the form of frequent false classifications. We determined thresholds for five key statistics frequently used in detecting randomness, namely Shannon entropy, chi-square, arithmetic mean, Monte Carlo estimation for Pi and serial correlation coefficient. We obtained a large data set of 84,327 files comprising of images, compressed data and encrypted data. We then tested these thresholds (taken from a variety of previous publications in the literature where possible) against our dataset, showing that the rate of false positives is far beyond what could be considered acceptable. False positive rates were often above 50% and even above 90% on several occasions. False negative rates were also generally between 5% and 20%, numbers which are also far too high. As a direct result of these experiments, we determine that relying on these simple statistical approaches is not good enough to detect ransomware attacks consistently. We instead recommend the exploration of higher-order statistics such as skewness and kurtosis for future ransomware detection techniques.

Keywords: ransomware · anti-ransomware · statistical tests · randomness · entropy · chi-square.

1 Introduction

Ransomware is a strain of malware which, upon compromising a victim's machine, denies access to a user's resources. Typically, this is achieved through the use of a hybrid cryptosystem, where user data is encrypted using symmetric keys. These keys are then encrypted using asymmetric cryptography, such as RSA, and the private key is held by the attacker on their *Command & Control* (C&C) infrastructure [12]. In this scenario, the attacker would hold the encrypted data for 'ransom', demanding a payment (typically via a cryptocurrency such as

Bitcoin) from the victim in order to restore their data. This type of ransomware is known as *Crypto-Ransomware* [16], and is the main scope of this analysis.

Ransomware is an ever-growing threat and continually causing widespread disruption. Alarmingly, we have recently observed more targeted attacks, i.e. ransomware attacks aimed at specific organisations with the intention of causing maximum damage [2]. For example, the Spanish company Everis was hit by the BitPaymer ransomware in 2019. It was shown that the ransom note deployed was specifically aimed at Everis, and the extension used for encrypted files was `.3v3r1s` [5]. Additionally, Norsk Hydro was hit by ransomware in 2019 and despite huge disruption to their production lines, they refused to pay the ransom. They were able to recover from the attack through use of trusted backup servers, whilst consulting paper documentation to continue business throughout the recovery process. This manual recovery process cost Norsk Hydro £45 million [24], but the company correctly refused to fund the cybercriminal economy.

Thankfully, we have also witnessed an increase in anti-ransomware research. One such avenue of research covers the development of techniques and tools aimed at the early detection and recovery from ransomware attacks. A recurring approach to detecting ransomware is the use of statistical tests for randomness, because properly encrypted data should appear completely random to anyone not in possession of the key. Therefore, the problem of detecting ransomware can be (somewhat simplistically) reduced to the problem of detecting random data being written to the filesystem. However, this assumption can be problematic.

Motivation. Whilst the use of statistical tests to detect ransomware has shown promise in the literature, it also raises issues. Most notably, the processing of random data on a machine does not automatically imply that it is under attack by ransomware. It is common for perfectly benign data on the filesystem to appear random and this happens with various image formats (e.g. JPEG and PNG), as well as frequently after the use of compression tools. Additionally, even if the presence of randomness is the result of encryption taking place, that does not necessarily imply that the encryption is malicious (i.e. the result of a ransomware attack).

In this work, we explore the former of these issues by investigating both popular image formats and types of compression commonly in use today. After collecting a representative dataset of JPEG, PNG and WebP images, and compressing and encrypting files from the Govdocs corpus [4], we ran them through Ent [26], a battery of statistical tests for randomness. We compared their output to various thresholds in use by current state-of-the-art anti-ransomware tools, and show our findings in Section 5. We show that the thresholds in use by these tools often result in too many false positives, which leads to unencrypted data in the filesystem being incorrectly labelled as encrypted by a ransomware attack.

Contributions. Firstly, we highlight the issue that one of the most popular approaches to detecting ransomware is potentially flawed. This weakness could present a serious problem to many anti-ransomware tools. We also provide insights into statistics beyond those that are currently used in this context, in order to illustrate the fact that the approach of using statistics to detect ransomware

needs improving in general. We would like to stress that this *does not* mean that there are underlying problems with the statistical tests themselves. The problem is with using the tests in this context (i.e. for ransomware detection).

Secondly, we provide a number of recommendations for future work in light of our results. We highlight that the statistics which show the most promise in this context are chi-square and serial byte correlation. However, we believe that the research community should also look more carefully to alternatives such as variance, standard deviation, skewness and kurtosis. Finally, in the interests of scientific reproducibility, we have made all of the code we used freely available and open-source, as well as the datasets we used in our analysis.

The rest of the paper is structured as follows. In Section 2, we discuss some of the major uses of statistics to identify ransomware, as well as previous works which analyse this approach. In Section 3, we provide some intuition as to why these tests are used in ransomware detection. Section 4 explains the methodology we followed in our experiments, and Section 5 presents and analyses our results. In Section 6, we discuss what these results mean for anti-ransomware development, and we provide some recommendations for ransomware detection moving forwards. Finally, in Section 7, we conclude our work.

2 Related Work

The use of statistical approaches to detect ransomware was the second most common approach to detecting ransomware in 2019, according to an analysis of the academic anti-ransomware landscape [20]. Genç et al. classify *measuring entropy inflation* as one of the main *behavioural analysis* approaches to defending against ransomware [7], and Al-Rimy et al. highlight the use of entropy in their analysis of ransomware research [1]. There are some potential reasons as to why statistical approaches may be so popular in this context. For example, it is quite logical to consider using randomness tests to detect encryption (since the process of encryption results in data that is effectively random). The relative ease with which these randomness tests can be implemented may also be a contributing factor. We expand more on this in Section 3.

Several anti-ransomware tools use a statistical approach. ShieldFS calculates the entropy of data written to the filesystem and uses this as a machine learning feature to help detect ransomware attacks [3]. ShieldFS implemented a Windows Filesystem Minifilter Driver [17] to observe filesystem *write* operations, including the data buffer over which the entropy could be calculated. In fact, the approach of computing the entropy over filesystem *write* operations is a popular approach. To evaluate the validity of a detection, UNVEIL calculates the entropy value over the data buffer of both *read* and *write* operations [10]. If there is a significant increase in entropy between a *read* and a *write*, random data has likely been written and so a ransomware attack may have occurred.

Similarly, Redemption looks at the difference in entropy between a *read* and its subsequent *write* [11]. If there was an increase, the "malice score" of the associated process is increased, highlighting that there is a greater chance that this is a ransomware process.

CryptoDrop also uses entropy to help with ransomware detection [21]. This tool relies on the fact that ransomware will continually make highly-entropic *writes* to the filesystem, and takes weighted entropy averages to ensure that the low-entropy *writes* resulting from writing ransom notes do not confuse the system. CryptoDrop also looks at the delta between a *read* and its subsequent *write* to determine the change in entropy to a specific file. Furthermore, a small delta (0.1) is used as the threshold, to help cater for the small entropy increase that occurs when a file that is already highly-entropic (such as compressed data) is encrypted by ransomware. RWGuard also measures entropy as an indicator of a ransomware attack; if the entropy of a *write* request to the filesystem results in a value greater than 6, the metrics recorded for the specific file are analysed further due to the increased possibility of a ransomware attack [15].

To the best of our knowledge, Data Aware Defence (DaD) is the only ransomware detection tool that uses chi-square, rather than entropy, as its detection method [19]. In fact, detection is solely achieved using chi-square. Similarly to the above-mentioned tools, this statistic is computed over the data buffer of filesystem *write* operations, and a sliding median over the last 50 *writes* is used as a basis for this calculation.

Other work has explored the robustness of using statistical approaches to detecting ransomware. McIntosh et al. conclude that the use of entropy to detect ransomware should be stopped altogether in future anti-ransomware work [14]. Two methods by which ransomware could implement encryption in such a way as to avoid entropy-based detection measures are presented, using techniques including Base64 encoding and partial encryption. Interestingly, the work presented tackles the same problem explored in this paper, although primarily from the perspective of false negatives rather than false positives.

3 Randomness for Anti-Ransomware

To provide some explanation of the applicability of using randomness tests to detect ransomware, consider data (such as *writes* made to the filesystem, as well as the content of files) purely as streams of bytes (i.e. values between 0 and 255). Random (or encrypted) data should comprise of an approximately equal number of each byte, distributed across the data in an unpredictable way. 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.

However, some filetypes are comprised of data which, from the perspective of statistical tests such as entropy, actually appears random. Calculating the entropy of a JPEG typically results in values of 7.8 and higher. Considering the highest possible value is 8 bits per byte (i.e. completely uniform), it is clear that an unencrypted file can look as if it has been encrypted. Figure 1 shows the entropy values of 1,004 JPEG files, 1,000 PNG files and 1,000 WebP files that we found in the wild (discussed in more detail in Section 4) compared with a threshold of 7.8 (shown as a red horizontal line). This shows the consistency with which these types of files contain highly entropic data, which raises an issue: files

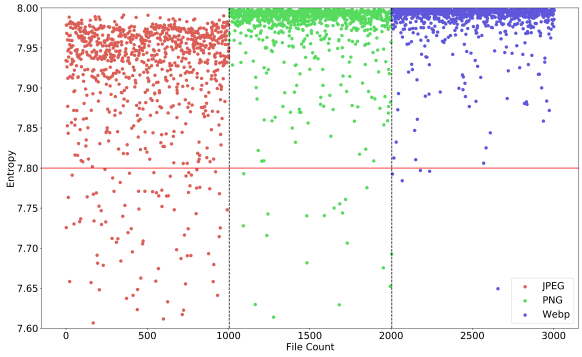


Fig. 1: Entropy values of 3,004 images found in the wild

like these will frequently cause false positives in anti-ransomware tools heavily reliant on this statistic. We explore this in more detail in Section 5.

Below, we expand on some of the main statistical tests used to detect the presence of a ransomware attack. Entropy and chi-square are currently used by the academic state-of-the-art in ransomware detection. However, we analyse three additional tests. Whilst these are not currently used by anti-ransomware tools, the ease with which they can be implemented may make them the next logical step for anti-ransomware developers so we felt it important to preemptively examine their accuracy in this context.

3.1 Shannon Entropy

In the context of ransomware detection, entropy can be seen as a measure of a given input's level of uncertainty. In this case, the input would be a series of bytes typically representing either a file's contents or the contents of a *write* request made to the filesystem. The equation for Shannon Entropy ($H(X)$) can be seen below for a random variable X [22]:

$$H(X) = -\sum_{i=0}^{255} P(x_i) log_b P(x_i)$$

In this case, the summation is between 0 and 255 as there are 256 possible values. Additionally, $b = 2$ allows the representation of bits, and $P(x_i)$ is $\frac{F_i}{totalbytes}$ where F_i is the observed frequency of byte i. This equation returns a value between 0 and 8, where 0 represents totally predictable data and 8 highly uncertain data.

3.2 Chi-Square

The chi-square (χ^2) test is a popular statistical test generally used to determine if an observed distribution is statistically similar to an expected distribution [6]. In the case of perfectly random data on the filesystem, we would expect an equal occurrence of each byte value. Therefore, for ransomware detection, we use equal frequencies of byte values for the expected distribution, and use the following formula to check if the observed distribution is similar:

$$\chi^2 = \sum_{i=0}^{255} \frac{(F_i - f_i)^2}{f_i}$$

Here, there are again 256 possible values for a given byte. F_i and f_i represent the observed and expected frequency of byte i, respectively.

As there are 256 possible categories that a given byte could be, the degrees of freedom for our chi-square test is set to 255 (i.e. $number\,of\,categories - 1$). This allows us to refer to a chi-square distribution table and find what results we should expect at a given significance level. Researchers often use significance levels of 1%, 5% or 10% [23], and 5% is used in Data Aware Defence [19]. We state as the null hypothesis that our observed input is random. After computing the chi-square, we compare it with a distribution table at a 5% significance level. If our value is higher than the value in the table, this situation would only occur 5% of the time for a perfectly random distribution. Our observed distribution is therefore unlikely to be random, so we reject the null hypothesis.

3.3 Other Statistical Tests

In our experiments we used Ent, a Pseudorandom Number Sequence Test Program [26] which quickly calculates various statistics. In addition to entropy and chi-square, Ent provides the following statistics which we have not yet seen used to detect ransomware:

- **Arithmetic Mean.** This metric is calculated by summing all of the individual byte values and dividing by the total number of bytes. In the event of random data, or a ransomware attack, we would expect a result close to 127.5, i.e. half way between 0 and 255.
- **Monte Carlo value for Pi.** For this statistic, every sequence of six bytes is used to calculate X and Y coordinates inside a square. For a circle inscribed within this square, the percentage of generated points that fall within the circle can be used to calculate the value of Pi. With sufficiently long and random data streams (for example due to a ransomware attack), the result will be close to the value of Pi.
- **Serial Correlation Coefficient.** Considering a byte stream of length n, it is possible to compare byte 0 with byte 1, byte 1 with byte 2 and so on up until byte n in order to calculate the correlation coefficient of this data. This is typically measured as a value between -1 and 1. The closer the value is to one of the extremes (i.e. -1 or 1), the stronger that type of correlation is. Random data (for example from a ransomware attack) should be highly uncorrelated. This approach was explored by Pont et al. [20] as a potential ransomware detection technique.

4 Methodology

In the following section, we first detail our data collection process. This covers the collection of JPEGs, WebPs, PNGs, compressed and encrypted data. Following this, we detail how we prepared our dataset for graph generation, and finally discuss the creation of our threshold values.

4.1 Dataset Creation

After noticing the popularity of statistical tests to detect ransomware [20], we began by collecting a large dataset on which we could calculate these statistics for ourselves. Other works have noted that in the case of some filetypes, such as

images and compressed data, their data is naturally highly entropic [11,13,21]. We therefore focused on these filetypes as we believed they would cause false positives in state-of-the-art anti-ransomware tools. Most of our data came from Digital Corpora's Govdocs [4]. This is a large corpus of real files from .gov domains that are freely available for research purposes. We first downloaded the entire JPEG image corpus, containing 109,282 JPEG images. This was downloaded as a single compressed directory approximately 36.8 GB in size (about 37.5 GB when decompressed). Within this directory were 961 subdirectories, each containing a number of JPEGs. In order to build a dataset of around 1,000 JPEGs in a way that is easily reproducible, we selected the first 15 of these subdirectories, providing us with a dataset of 1,004 JPEGs (about 145.4 MB).

We then used ImageAssisstant Batch Image Downloader (a Firefox addon which unfortunately seems to have been removed from the addon marketplace) to collect 1,000 WebP images using the Google search engine. Hurley-Smith et al. show that WebP files are frequently reported as random by Ent and the FIPS 140-2 randomness tests [9], so we felt it an important filetype to include in our experiments. We have yet to come across an anti-ransomware tool that includes this filetype as part of their dataset, which is concerning due to its rising popularity. In fact, WebP is in use by approximately 20,000 of all websites on Alexa website rankings, with uptake steadily on the increase [25].

We obtained these files by searching for a keyword followed by the `filetype:` operator. As an example, we searched for `mountains filetype:webp`, then used the Firefox addon to download a selection of the results in bulk. After repeating this process for 15 arbitrarily chosen keywords, we completed our collection of 1,000 WebP images (at approximately 77.1 MB in size). We include these keywords (along with the images themselves) in our dataset, although using different keywords for future experiments may be a good way to corroborate our findings. We repeated this process to collect 1,000 PNG images (at approximately 778 MB in size). This time, only 14 keyword searches were required to reach the desired quantity of 1,000 images. These two lists of keywords were kept mutually exclusive to ensure as diverse a dataset as possible.

To complete our image collection, we took our JPEGs and PNGs and used the command line utility `cwebp` to convert them to WebP at various quality levels. This was achieved using a bash script which takes every JPEG and PNG, runs `cwebp` at quality levels 0, 25, 50, 75, 80 (the typical quality level used according to the tool itself) and 100, and stores the output in our WebP directory. We repeated this process with the `-lossless` option to include lossless WebPs.

We then moved towards compressed data, whose tendency to generate false positives has been noted in the literature [7,15]. We compressed data from the Govdocs *threads*, which are mutually exclusive sets of approximately 1,000 files. We chose Thread 4 and Thread 5 due to their larger size (containing 986 files (311 MB) and 989 files (295 MB) respectively). We used a bash script to call the Gzip, BZip2 and LZMA command line utilities to compress each file separately at each compression level (0 through 9 for LZMA, otherwise 1 through 9). A detailed breakdown of our dataset is presented in Table 1.

Table 1: The breakdown of our image files dataset

Filetype	Quantity	Size (MB)
JPEG	1,004	145.4
WebP	25,048	6100.1
PNG	1,000	778.0
LZMA	19,750	5,772.0
Gzip	17,775	5,527.0
Bzip2	17,775	5,351.0
AES Encrypted	1,975	951.7
Total	84,327 Files	24,625.2 MB

Table 2: Statistical thresholds to identify randomness

Statistic	Randomness Threshold
Entropy	≥ 7.99
Chi-Square	≤ 293.25
Arithmetic Mean	$126.23 \leq$ value ≤ 128.78
Monte Carlo Value for Pi	$3.11 \leq$ value ≤ 3.17
Serial Correlation Coefficient	$-0.01 \leq$ value ≤ 0.01

Baseline Dataset. In order to provide an idea of the statistics we would expect for data that really has been encrypted, we encrypted each file in Thread 4 and Thread 5 separately using `openssl`, a command line utility on Linux allowing the use of encryption algorithms [18]. We used the AES symmetric encryption algorithm with a 256 bit key and the CBC mode, as ransomware variants typically implement a hybrid encryption model where user data is encrypted with symmetric encryption (such as AES), and the symmetric keys are encrypted with asymmetric encryption (such as RSA) [16].

4.2 Dataset Preparation

To calculate the statistics described in Section 3, we wrote a Python script to call `ent` on the command line for each file in our dataset. Using the terse mode (`-t`) option, we were able to generate output in .csv format for easy processing. These steps provided us with the five statistics we needed for each file. We acknowledge that anti-ransomware tools often process individual filesystem operation buffers [3,10], however by processing entire files we are providing the tests with more data in order to increase accuracy. We then created the graphs in this work using Matplotlib [8], a data visualisation library for Python.

4.3 Threshold Creation

Of the five statistics we have looked at in this paper, to the best of our knowledge, only two (entropy and chi-square) are actively being used by anti-ransomware tools. Whilst, for entropy, the absolute threshold values used by the current state-of-the-art in anti-ransomware do not seem to be widely reported, an overall indication is given as to what can be considered as highly entropic data. For example, in ShieldFS, an entropy value of 0.948 (recorded on a scale between 0 and 1 – when scaled up to a scale of 0 to 8, this becomes 7.584) is considered as "very high" [3]. We set our entropy threshold as 7.99 to ensure that only the most uncertain of data is considered as encrypted. The threshold for chi-square was taken based on consulting a chi-square value table at 255 degrees of freedom with a significance level of 5%, as discussed in Section 3. This gives us a threshold of 293.25, the same value that was used in Data Aware Defence [19].

For the three remaining tests (arithmetic mean, Monte Carlo value for Pi, and serial correlation coefficient), we defined thresholds based on a 1% error margin.

We consider any values calculated that fall within 1% of the baseline values to be random. The baseline values for arithmetic mean, Monte Carlo value for Pi and serial correlation coefficient are: 127.5, 3.14 and 0.00, respectively. Values within this range are treated as cases that would be detected as ransomware (i.e. a false positive for our images and compressed data, and a true positive for our encrypted data). Table 2 summarises the thresholds we used in our experiments.

5 Results and Analysis

We break the analysis of our results down into two main parts: an analysis of 'false classifications', and general observations. Many academic anti-ransomware tools are not open-source or available for use, so in our experiments we were unable to determine false classification rates of the tools themselves. Self-reported results of these anti-ransomware tools are summarised in [20], although interestingly this work highlights that many *false positive rates* (FPRs) are not reported. Below, we investigate the statistics introduced in Section 3.

Minimal FPRs are vitally important for real-time ransomware detection tools that always run in the background to prevent a user from instinctively dismissing alerts (putting them at risk of dismissing a real attack) or stopping using the tool altogether. Where the thresholds are not made available, we set our own, as described in Section 4. We analyse the percentage of our dataset that falls above and below these thresholds, providing an indication into the FPRs summarised in Table 3 and Table 4.

We also included an analysis of *false negative rates* (FNRs), as shown in Table 5. This would mean truly encrypted data that is incorrectly classified as being unencrypted. We acknowledge that these are devastating to a user, however minimising FNRs is beyond the scope of this paper – it is something that the authors of the respective tools themselves aim to minimise.

5.1 False Classification Analysis

False Positives Rates (FPRs). Table 3 and Table 4 summarise the FPRs we saw in our experiments across our images and compressed data, respectively. For each quality level used (shown as a percentage on the left), we include the number of files detected as a false positive, alongside the percentage of our dataset that this number equates to (i.e. the FPR). We also highlight the highest FPRs from our experiments in bold. We note that in Table 4, we only show the results for the three compression algorithms (BZip2, GZip and LZMA) at three levels of compression rate (1, 5 and 9). This was in the interest of creating a more succinct and readable table. Our complete set of results and graphs are available on GitHub (https://github.com/anti-ransomware/stats-tools-research).

Focusing first on the entropy and chi-square of our image dataset, we see a range in FPRs from 0% with chi-square to 83.90% using entropy. At first glance, this reinforces the idea that chi-square is better at distinguishing between encryption and JPEG compression [13]. However, analysing further, it quickly becomes clear that using chi-square is not necessarily the complete solution to the problem. For example, we see FPRs in the range of 43.13% to 76.69% when analysing lossy WebP files which have been converted from JPEGs at various

Table 3: False positive analysis for images

Data			False Positives									
			Entropy		Chi-Square		Mean		Pi		Correlation	
			Count	%	Count	%	Count	%	Count	%	Count	%
JPEG			3	0.30%	0	0%	178	17.73%	231	23.01%	92	9.16%
PNG			468	46.80%	2	0.20%	519	51.90%	478	47.80%	74	7.40%
WebP			677	67.70%	454	45.40%	839	83.90%	726	72.60%	668	66.80%
WebP (from JPEG)	Lossless	0%	193	19.22%	0	0%	241	24.00%	342	34.06%	4	0.40%
		25%	187	18.63%	0	0%	226	22.51%	312	31.08%	5	0.50%
		50%	397	39.54%	3	0.30%	396	39.44%	483	48.11%	9	0.90%
		75%	403	40.14%	5	0.50%	391	38.94%	477	47.51%	8	0.80%
		80%	411	40.94%	5	0.50%	398	39.64%	481	47.91%	8	0.80%
		100%	417	41.53%	3	0.30%	389	38.75%	484	48.21%	4	0.40%
	Lossy	0%	18	1.79%	433	43.13%	373	37.15%	300	29.88%	286	28.49%
		25%	267	26.59%	759	75.60%	736	73.31%	505	50.30%	582	57.97%
		50%	383	38.15%	764	76.10%	839	83.57%	583	58.07%	669	66.63%
		75%	458	45.62%	770	76.69%	878	87.45%	641	63.84%	729	72.61%
		80%	505	50.30%	749	74.60%	873	86.95%	654	65.14%	782	77.89%
		100%	798	79.48%	569	56.67%	916	91.24%	742	73.90%	895	89.14%
WebP (from PNG)	Lossless	0%	335	33.50%	2	0.20%	450	45.00%	474	47.40%	25	2.50%
		25%	357	35.70%	5	0.50%	424	42.40%	482	48.20%	34	3.40%
		50%	546	54.60%	21	2.10%	555	55.50%	586	58.60%	84	8.40%
		75%	569	56.90%	18	1.80%	566	56.60%	605	60.50%	96	9.60%
		80%	571	57.10%	18	1.80%	559	55.90%	593	59.30%	96	9.60%
		100%	609	60.90%	25	2.50%	619	61.90%	644	64.40%	93	9.30%
	Lossy	0%	39	3.90%	202	20.20%	392	39.20%	374	37.40%	294	29.40%
		25%	408	40.80%	501	50.10%	761	76.10%	600	60.00%	586	58.60%
		50%	543	54.30%	546	54.60%	839	83.90%	671	67.10%	659	65.90%
		75%	620	62.00%	515	51.50%	863	86.30%	732	73.20%	696	69.60%
		80%	665	66.50%	507	50.70%	884	88.40%	737	73.70%	710	71.00%
		100%	839	83.90%	417	41.70%	928	**92.80%**	826	82.60%	850	85.00%

Table 4: False positive analysis for compressed data

Data		False Positives									
		Entropy		Chi-Square		Mean		Pi		Correlation	
		Count	%	Count	%	Count	%	Count	%	Count	%
BZip2	1	373	37.83%	56	5.68%	395	40.06%	377	38.24%	113	11.46%
	5	382	38.74%	62	6.29%	394	39.96%	405	41.08%	143	14.50%
	9	384	38.95%	63	6.39%	395	40.06%	403	40.87%	142	14.40%
GZip	1	418	42.39%	235	23.83%	446	45.23%	348	35.29%	409	41.48%
	5	401	40.67%	250	25.35%	464	47.06%	356	36.11%	428	43.41%
	9	400	40.57%	259	26.27%	471	47.77%	377	38.24%	446	45.23%
LZMA	1	526	53.35%	913	**92.60%**	868	88.03%	667	67.65%	731	74.14%
	5	511	51.83%	910	92.29%	863	87.53%	664	67.34%	730	74.04%
	9	509	51.62%	907	91.99%	853	86.51%	663	67.24%	726	73.63%

quality levels. To top this off, when analysing WebPs found in the wild, we still see an FPR of 45.40%, indicating that almost half of this part of our dataset would cause a false positive. We believe this is a serious issue due to the rising popularity of WebP images in the wild [25].

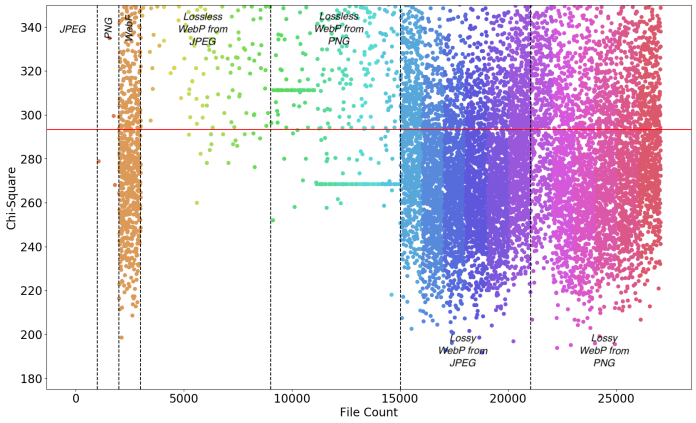
Looking at entropy and chi-square overall, it does appear, however, to be the case that chi-square is in general a better indicator of ransomware (at least for images). Chi-square outperformed entropy (i.e. achieved a lower FPR) in almost all of our batches of data. Interestingly, however, entropy outperformed chi-square for the case of Lossy WebPs which had been converted from JPEGs.

The remaining FPRs were calculated based on our own thresholds, as discussed in Section 4. Arithmetic mean in general seems to be a poor indicator based on the fact that at its best, it still had an FPR of 17.73% and at its worst it had an FPR of 92.80%. This FPR level would be absolutely unacceptable in any context. The situation is similar for the value of Pi, which at its best achieved

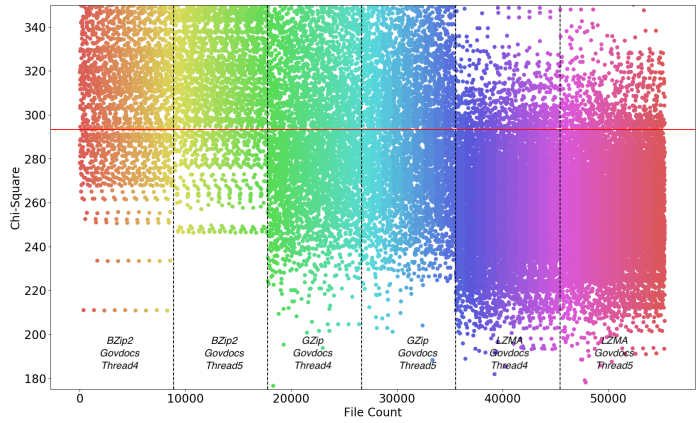
an FPR of 23.01% and at its worst, 82.60%. Interestingly, for both arithmetic mean and Pi, the best cases were achieved for JPEG, and the worst cases were on Lossy WebPs converted from PNGs at 100% quality.

Serial correlation coefficient, at its best, achieved an FPR of just 0.40%. This was for lossless WebPs converted from JPEGs at 0% quality. We believe this kind of FPR would be more palatable to the average end user. However, at its worst, it had an FPR of 89.14%, higher than the worst case of Pi.

Figure 2a and Figure 2b show the chi-square distributions of our image and compressed dataset, respectively. The threshold of 293.25 is also included for reference (shown as a red horizontal line). For clarification, we have divided the graph into each of the major sub-divisions of our dataset. Within these sub-divisions are further divisions represented by a change in the corresponding point's colour. These separations represent the different quality levels used in the conversion process (0%, 25%, 50%, 75%, 80% and 100%, respectively). The same is true for the graphs representing the other statistics we calculated, for example those which can be found on GitHub.



(a) Chi-square of 27,052 images



(b) Chi-square of 55,300 compressed files

Fig. 2: Chi-square of our dataset

Table 5: False negative analysis for encrypted data

	Data	False Negatives									
		Entropy		Chi-Square		Mean		Pi		Correlation	
		Count	%	Count	%	Count	%	Count	%	Count	%
AES	Thread4	230	23.33%	48	4.87%	51	5.17%	184	18.66%	115	11.66%
	Thread5	241	**24.37%**	49	4.95%	50	5.06%	174	17.59%	133	13.45%

We consider data points that fall *below* this line as false positives. We also note that some points are not visible on the graph due to chi-square values much higher than our axis limits. The only data type to achieve zero false positives is JPEG. The false positive count of PNG is low (i.e. 2), but all other data types have a high number of false positives. An interesting point to note is the huge number of false positives received for WebP when lossy compression is used. Our experiments show much lower FPRs when lossless compression is used. However, the FPR of our WebPs "from the wild" (i.e. 45.40%) suggest that the most common type of WebP compression in use is lossy. In fact, when cross-referencing the FPR of all WebPs from Table 3, it seems the most common type of WebP are those from PNGs and using lossy conversion.

Shifting to look at the compressed data, FPRs for both entropy and chi-square unfortunately do not look very promising. As discussed in Section 4, compressed data is often highlighted as a potential cause of false positives, so we hope our results reaffirm this serious issue. Looking at Table 4, the FPR for entropy is consistently within the range of 37.83% and 53.35%. Whilst these rates are generally more promising than those of our image dataset, we still deem them to be far beyond the realms of acceptability. Kharraz et al. conduct usability testing in [11], which may be a crucial step going forward to identify an acceptable FPR from a user's perspective. Interestingly, the range of FPRs for chi-square is much larger. At its best, chi-square had an FPR of 5.68%, which is closer (although still not satisfyingly enough) to what could be considered acceptable. However, at its worst, we observe an FPR of 92.60%. This is almost the highest FPR observed across all of our experiments, topped only by using arithmetic mean on lossy WebPs from PNGs at 100% quality.

Moving on to the three remaining statistics, FPRs are again far too high to be considered acceptable. The best performance we see is for BZip2 at a level 1 compression rate. Using correlation, we see an FPR of 11.46%. However, FPRs for these statistics are generally in the range of 40 to 60%, even reaching 88.03% in the case of using arithmetic mean for data compressed using LZMA at level 1 compression rate.

False Negative Rates (FNRs). Table 5 summarises the FNRs we saw in our experiments for the individual statistics. As with the above tables, we provide the number of files detected as a false negative, alongside the FNRs, whilst highlighting the highest FNR in bold. In this context, this represents data that has been encrypted but has incorrectly been classified as not encrypted. In a real life scenario, this would represent ransomware encrypting a user's files without any active protection mechanisms alerting the user to some form of malicious activity.

Whilst this is not the focus of our paper, we thought it would still be relevant to report our findings. The best case we saw was an FNR of 4.87% when using

the chi-square statistic over Thread 4. Conversely, the worst case we saw was an FNR of 24.37% when using entropy over Thread 5. Thankfully, we recorded no FNR higher than this, although we still believe that these rates are too high to be acceptable. In this case, almost a quarter of encrypted files go undetected.

An important point we would like to mention is that it was our choice to consider this encryption as malicious. We wanted to include this data to represent what data would look like from a statistical point of view if it had been encrypted by ransomware. However, the presence of encryption on a system does not automatically imply that a ransomware attack is underway. For example, benign applications may use encryption for communication, or a user may wish to encrypt their files for privacy.

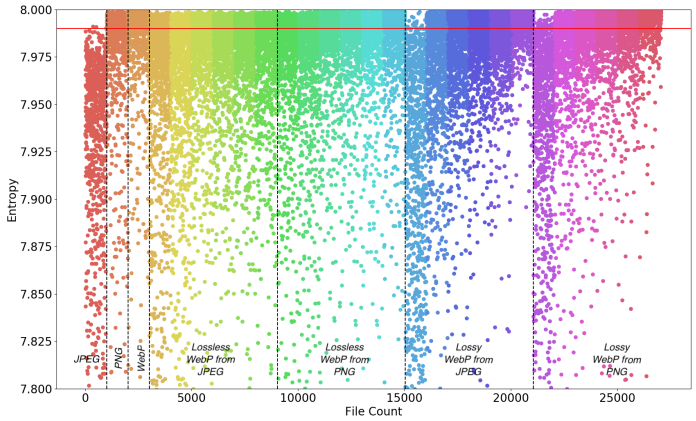


Fig. 3: Entropy of 27,052 images

5.2 General Observations

Figure 3 shows the entropy values calculated for our image dataset, which accurately summaries the patterns we saw for the majority of our other statistics. Within each major sub-division of the dataset, it is clear how – as we progress from left to right through the different conversion quality levels – the dispersion of entropy decreases. In other words, as both JPEGs and PNGs are converted to WebPs at higher quality levels, the resulting entropy of the data is increased. For this reason and that a similar pattern can be observed for the other statistics, we believe investigating the variance, standard deviation and higher-order statistics such as skewness and kurtosis could be a step towards detecting consistently random data, but this would require more experimentation.

Due to the uncertainty as to which filetypes any given user may have on their computer, we do not believe the solution is as simple as picking the statistics that achieve the lowest FPR and FNR. It may be the case that whilst this works well for some users, it does not work at all for others. For example, the obvious choice would be using chi-square or serial correlation coefficient due to their lower FPRs and FNRs in general, but they don't always perform well.

We believe that these results highlight a serious flaw in the current state-of-the-art in ransomware detection. Whilst the tools are generally excellent at detecting ransomware, more effort needs to be put into reducing FPRs. We

acknowledge that these tools often use statistics as part of a wider detection mechanism, for example behavioural analysis such as in ShieldFS [3]. Despite this, it is very common for these tools to place heavy reliance on the results of statistical tests. We believe this is worrying due to their susceptibility to errors, as shown by our results. We would again like to stress that this *does not* mean that there are underlying problems with these tests; the problem is with using them for ransomware detection.

6 Recommendations and Future Work

As discussed in Section 2, McIntosh et al. recommended that future research should avoid the use of entropy for ransomware detection [14]. This was due to the relative ease by which ransomware could implement encryption without triggering any entropy thresholds in place. We come to a similar conclusion from the perspective of false positives rather than false negatives. The immediately obvious recommendation would be to avoid the *sole* use of entropy to reliably and consistently detect a ransomware attack. The frequency of false positives in our results show that an average user would be plagued by false alarms, ending in a practically unusable system. However, our results also show that the problem is not just with entropy but for all of the statistics we tested. Whilst some statistics (entropy, chi-square and serial correlation coefficient) performed extremely well in certain cases (e.g. FPRs of around 0% to 0.5%), they did not perform this well consistently across our experiments. We are therefore unable to recommend a single statistic as the optimal way of detecting ransomware reliably.

Due to their lower FPRs whilst still achieving the lowest FNRs, we believe chi-square and serial correlation coefficient deserve the most attention going forwards. To the best of our knowledge, Data Aware Defence is the only tool so far that has used chi-square for ransomware detection [19].

The statistics we calculated were for single files at any one given time. An improved approach would be to identify deltas in these statistical values over time for a given set of files. This idea has been explored by Redemption [11] and CryptoLock [21]. It should be immediately obvious when ransomware writes to a file by identifying a significant change in (for example) the entropy value of the *read*, followed by the subsequent *write*. This approach may still be susceptible to false positives, for example if a file is highly structured before the encryption takes place (like much of the data used in our dataset).

As discussed in Section 5.2, we believe it would be worthwhile for the anti-ransomware community to investigate variance and standard deviation of the previously discussed statistics. We would expect highly random data to be written to the filesystem *consistently* during a ransomware attack. By calculating the statistics of these *writes* for a given time window, the variance and standard deviation could be calculated in order to determine spread and dispersion from central tendency. We expect a low variance and standard deviation in the case of consistent highly random *writes* to the filesystem, but a high variance and standard deviation in the case of normal system usage. It may also be possible to apply this technique using higher-order statistics such as skewness and kurtosis, although this would require further experimentation.

7 Conclusion

In this paper, we have highlighted the very serious issue that in the context of ransomware detection, popular file formats in use by typical computer users can cause frequent false positive alerts when analysed with various statistical tests for randomness. We analysed a dataset of 84,327 files (at 24.6 GB) consisting of JPEG images, PNG images, WebP images, compressed data (using BZip2, GZip and LZMA), and encrypted data (using AES in CBC mode with a 256 bit key). On this dataset, we calculated values for entropy, chi-square, arithmetic mean, Monte Carlo value for Pi and serial correlation coefficient (using the command line tool Ent). We compared these values against thresholds that were both found in and based on the literature (using a 1% error margin where no thresholds were available) to determine their false classification rates.

We observed FPRs of up to 92.80%, with a large proportion of our dataset attaining FPRs of over 80%. Only an extremely small proportion achieved rates that could be considered acceptable (i.e. below 0.5%). In addition, the lowest FNR we saw was still 5.06%, with the highest being 24.37%. This shows that even in the best case for our dataset, approximately five out of 100 files could be maliciously encrypted without being recognised by these tests.

Some of these tests are in use by many of the state-of-the-art tools in ransomware detection. We believe our results indicate that testing of these anti-ransomware tools has not been sufficient. We therefore believe future anti-ransomware tools should be tested on much larger and representative datasets, particularly including lots of images (especially WebP files) and compressed data, to ensure FPRs and FNRs are as accurate and realistic as possible. This, combined with a detailed analysis and visualisation of these results, would help to highlight the true accuracy of these tools. Finally, experimenting with the use of variance, standard deviation and higher-order statistics such as skewness and kurtosis may help to classify ransomware attacks more accurately and consistently.

References

1. Al-rimy, B.A.S., Maarof, M.A., Shaid, S.Z.M.: Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. Computers & Security **74**, 144–166 (2018)
2. Constantin, L.: More targeted, sophisticated and costly: Why ransomware might be your biggest threat (February 2020), https://www.csoonline.com/article/3518864/more-targeted-sophisticated-and-costly-why-ransomware-might-be-your-biggest-threat.html
3. Continella, A., Guagnelli, A., Zingaro, G., De Pasquale, G., Barenghi, A., Zanero, S., Maggi, F.: Shieldfs: a self-healing, ransomware-aware filesystem. In: Procs. 32nd Annual Conference on Computer Security Applications. pp. 336–347 (2016)
4. Digital Corpora: (2018), https://digitalcorpora.org
5. Esparza, J.M., Blueliv: Spanish consultancy everis suffers bitpaymer ransomware attack: a brief analysis (November 2019), https://www.blueliv.com/cyber-security-and-cyber-threat-intelligence-blog-blueliv/research/everis-bitpaymer-ransomware-attack-analysis-dridex/

6. F.R.S., K.P.: 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 **50**(302), 157–175 (1900), https://doi.org/10.1080/14786440009463897
7. Genç, Z.A., Lenzini, G., Ryan, P.Y.: Next generation cryptographic ransomware. In: Nordic Conference on Secure IT Systems. pp. 385–401. Springer (2018)
8. Hunter, J.D.: Matplotlib: A 2d graphics environment. Computing in Science & Engineering **9**(3), 90–95 (2007), https://doi.org/10.1109/MCSE.2007.55
9. Hurley-Smith, D., Patsakis, C., Hernandez-Castro, J.: On the unbearable lightness of FIPS 140-2 randomness tests. IEEE Trans on Inf Forensics and Security (2020)
10. Kharraz, A., Arshad, S., Mulliner, C., Robertson, W., Kirda, E.: UNVEIL: A large-scale, automated approach to detecting ransomware. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 757–772 (2016)
11. Kharraz, A., Kirda, E.: Redemption: Real-time protection against ransomware at end-hosts. In: International Symposium on Research in Attacks, Intrusions, and Defenses. pp. 98–119. Springer (2017)
12. Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., Kirda, E.: Cutting the gordian knot: A look under the hood of ransomware attacks. In: Intl Conf on Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 3–24. Springer (2015)
13. Mbol, F., Robert, J.M., Sadighian, A.: An efficient approach to detect torrentlocker ransomware in computer systems. In: International Conference on Cryptology and Network Security. pp. 532–541. Springer (2016)
14. McIntosh, T., Jang-Jaccard, J., Watters, P., Susnjak, T.: The inadequacy of entropy-based ransomware detection. In: International Conference on Neural Information Processing. pp. 181–189. Springer (2019)
15. Mehnaz, S., Mudgerikar, A., Bertino, E.: Rwguard: A real-time detection system against cryptographic ransomware. In: RAID2018. pp. 114–136. Springer (2018)
16. Micro, T.: Ransomware (September 2016), https://www.trendmicro.com/vinfo/us/security/definition/ransomware
17. Microsoft: Kernel-mode driver architecture design guide (June 2017), https://docs.microsoft.com/en-gb/windows-hardware/drivers/kernel/
18. OpenSSL Software Foundation: Openssl, https://www.openssl.org
19. Palisse, A., Durand, A., Le Bouder, H., Le Guernic, C., Lanet, J.L.: Data aware defense (dad): towards a generic and practical ransomware countermeasure. In: Nordic Conference on Secure IT Systems. pp. 192–208. Springer (2017)
20. Pont, J., Oun, O.A., Brierley, C., Arief, B., Hernandez-Castro, J.: A roadmap for improving the impact of anti-ransomware research. In: Nordic Conference on Secure IT Systems. pp. 137–154. Springer (2019)
21. Scaife, N., Carter, H., Traynor, P., Butler, K.R.: Cryptolock (and drop it): stopping ransomware attacks on user data. In: 36th International Conference on Distributed Computing Systems (ICDCS). pp. 303–312. IEEE (2016)
22. Shannon, C.E.: A mathematical theory of communication. Bell system technical journal **27**(3), 379–423 (1948)
23. Stat Trek: Chi-square test for independence (2020), https://stattrek.com/chi-square-test/independence.aspx
24. Tidy, J.: How a ransomware attack cost one firm £45m (June 2019), https://www.bbc.co.uk/news/business-48661152
25. W3Techs: Usage statistics of webp for websites (2020), https://w3techs.com/technologies/details/im-webp
26. Walker, J.: Ent (2008), https://www.fourmilab.ch/random/