ITIS 5250–Computer Forensics
Professor Victor Grose
University of North Carolina at Charlotte

Evaluating Entropy-Based Encryption Detection Methods

Student: David Gary
Student ID: 801325583
Date: December 5, 2023

Contents

1 Abstract

Detecting encrypted files is a crucial step in digital forensics investigations. But current methods often struggle with high false positive rates. In this project, the false positive rate of the encryption detection method used by Autopsy, an industry standard digital forensics tool, is examined. Alternative methods are proposed and tested, with results showing a significant reduction in false positives - all without significant increases in computational cost.

2 Introduction

Forensic examinations of digital media often check for the presence of encrypted files, as they may contain key evidence of criminal activity[26]. However, this detection process does not come easily, and even industry standard approaches have been shown to be ineffective[23]. In this project, the detection approach used by an industry standard tool, Autopsy[28], is thoroughly examined and compared against slight variations in numerous experiments. The results of these experiments show that, while the detection approach used by Autopsy is effective, there is a significant risk of false positives. Additional detection methods used in this project show a significant reduction in false positives.

3 Background

Improving on the current methods of encryption detection requires at least a foundational understanding of cryptography, along with decent intuition on how to break cryptosystems. Additionally, it is crucial to understand the current methods used to detect encryption in the modern field of digital forensics. To this end, this section will provide an overview of the core mathematical cryptography concepts, reveal the current method used to detect encryption in a major digital forensics tool, and highlight recent research in the field.

3.1 Encoding

Encoding methods are used to convert data from one format to another, and while the output may not always appear to be human-readable, it is always easily reversible. The most hidden aspect of an encoded message is the method used to encode it, as this is not always apparent. To show how a typical encoding method works, the method known as Base64[16] will be used as an example.

Base64 takes in binary data and converts it into a string of ASCII[24] characters, producing an output that is 33% larger than the input. This size increase comes from the number of bits used to represent each character: 8 bits for ASCII, and 6 bits for Base64. These 6 bits are used to represent the 64 distinct characters used in the Base64 encoding method.

Similar Base$X$ encoding methods exist, all following the pattern of using an alphabet of size X to represent the input data. The binary representations of the characters can always be recovered and converted back to the desired format, as long as the encoding method is known. This all may seem trivial, but it's important to highlight these basics because two core components of the later analysis steps are beginning to take shape: uniformity and entropy. The frequency of each possible byte variation's occurrence matters greatly, so the different Base encoding methods raise an interesting question: does the alphabet size relate to the entropy of the encoded data?

3.2 Hashing

A hash function is a function that takes in an input and produces a fixed-length output, known as a hash or hash value[14, 20]. Hash values are commonly used in digital forensics to verify the integrity of data, as they can be used to determine if a file has been modified[26] because hash functions deterministic and the hash values are unique to the input. While this sounds similar to an encoding method, the core difference is that hash functions are irreversible – the original input cannot be recovered from a hash value.

3.3 Encryption Methods

Encryption methods are used to secure data by converting it into an indecipherable format, known as ciphertext, that can only be recovered by authorized parties. Recovering the original data from a ciphertext requires a piece of information that is known by the authorized parties, known as a key. The encryption method used to encrypt the data and the key used to encrypt it are both required to decrypt the data. Specifically for this project, the encryption methods used will be symmetric block ciphers, meaning the same key is used to encrypt and decrypt the data, and the data is split into blocks of a fixed size before being encrypted[14, 20].

3.4 Entropy

Entropy is a measure of a given system's disorder, a feature that encryption methods attempt to maximize in order to make the encrypted data as difficult to decipher as possible[20]. The entropy of a system is calculated by determining the probability of each possible state of the system, and then using that probability to calculate the entropy value. To calculate the entropy of a file, the probability of each possible byte value is determined, and the file contents are treated as a sequence of independent and identically distributed random variables[3, 23, 17, 30]. The resulting entropy value gives a relative measure of how random the file contents are. Higher entropy indicates a more random file, and from this, inferences are often made about the file's current state.

3.5 Autopsy

Autopsy is a software tool for digital forensics analysis, and it is commonly used by law enforcement agencies and private investigators[28]. It is an open-source project that is supported by the non-profit organization, The Sleuth Kit[28], and it allows for third party modules to be added into the core framework. Due to its ease of use and open-source nature, Autopsy is highly popular.

Its core functionality includes a tool for detecting encrypted files, which upon further examination of the source code[3], calculates Shannon entropy[14, 20] for each file and compares it to a defined threshold. If the entropy value is greater than the threshold, the file is marked as encrypted. However, this method has been shown to have some issues[23, 17, 30], and it is the goal of this project to examine these flaws and find new methods to reduce error rates.

3.6 Relevant Research

Accurately detecting encrypted or malicious files is a complicated task, as there are rarely deterministic identification methods that can be used with full certainty. Research in this area is typically focused on malware detection, not necessarily encryption detection, but the two are closely related. To this end, this section will highlight the approach and findings of relevant recent research works and explain how they influenced the design of this project.

Lee et al.[17] proposed a method for detecting ransomware in cloud storage systems. Their method utilized an entropy calculation, recovery of file format information, Markov chain analysis, compression ratio analysis, and collision test estimation. Although, their entropy calculation did not use a standard method of full file entropy calculation, instead using focusing only on the most common byte value in the file. Their method was able to detect ransomware-infected files with 100% accuracy and a false positive rate of 0.0%. But this comes at an increased computational cost[18], along with the need for contextual information that may not always be available. Improving this approach's efficiency would require quasi-newton methods[8] to be used in the Markov chain analysis and would be too complicated for the scope of this project.

Pont, Arief, and Hernandez-Castro[23] reviewed the failings of common statistical approaches for detecting ransomware, finding false positive rates in some contexts as high as 90% and false negative rates up to 20%. The statistical approaches they reviewed included Shannon entropy, chi-square, arithmetic mean, and Monte Carlo estimation. Their recommendation was to use higher order statistics (kurtosis and skew) to improve the accuracy of the detection method. To this end, this project paid special attention to these higher order methods in the analysis phase.

In 2021, Davies, Macfarlane, and Buchanan[11] proposed a method for detecting ransomware attacks in mixed file datasets using differential area analysis. Their experiments revealed a unique characteristic for the Shannon entropy of encrypted files, specifically in their file header fragments. They proposed a detection method using classification model that analyzes differential area values derived from Shannon entropy calculations – with an accuracy rate up to 99.96%. But again, the computational cost of this method is high, and the reported characteristics of the file header fragments for encrypted files needed further investigation and verification before being used in this project.

The 2011 work of Manjula and Anitha[19] used a decision tree to detect the type of encryption used on a file. This work came earlier than other research works discussed, and the lack of decision tree usage in more recent studies may be, in part, a result of the findings of this work, as their method could only determine the encryption method used 70 to 75% accuracy. Decision trees are, similar to other approaches discussed, highly computationally expensive, and they require a large amount of data to be trained on. Seeing the pitfalls in a decision tree approach, this project opted to emphasize computational efficiency.

4 Design

4.1 Testing Data Used

Since different file types utilize a variety of encodings, character sets, and compression methods (among other things), it is important to test the detection methods on a variety of different file types. To accomplish this, a collection of 27 different files was assembled, including text files, SQLite databases, PDFs, and compressed files (zip and tar.gz). This file collection is a good starting point for testing the efficacy of entropy-based encryption detection methods, but it is by no means exhaustive. For detailed descriptions of each file in the collection, see the Appendix 8.1 section of this report.

4.2 Obfuscation Methods

A mixture of methods were used to obfuscate the contents of the testing data files, including four encoding methods (Base16, Base32, Base64, and Base85)[16], four hashing methods (MD5, SHA-1, SHA-256, and SHA-512)[25, 22], and four encryption methods (AES, Blowfish, ChaCha20, and Salsa20)[21, 7, 6, 27] The three categories of obfuscation method used (encoded, hashed, and encrypted) provide a range of file complexity and set the stage for potential false positives for the entropy-based detection methods. Python libraries were used to ensure that the obfuscation methods used were up to industry standard and incldued `base64` for the encoding methods, `hashlib` for the hashing methods, and `cryptography` for the encryption methods.

4.3 Detection Methods

All of the detection methods used in this project were based on entropy calculations, but the way in which this calculation was performed varied. Specifically, these methods included Shannon entropy, Min entropy, Reyni entropy, and Collision entropy. Mathematical definitions of these entropy calculation methods are provided in the Appendix 8.2 section of this report. Every detection method was implemented from scratch in Python and used to analyze each file in both the original and obfuscated states. The results of these analyses were then compared to determine the efficacy of each detection method.

To reflect the settings used in Autopsy[28], the resultant entropy values were considered to be indicative of encryption if they were greater than 6. This is the default threshold used in Autopsy's source code[3], and it is the minimum value that can be set in the user interface. Further explorations int the impacts of this threshold are still needed, but for the purposes of these experiments, matching industry standard defaults was the goal.

5 Results

In this section, we present the results of the experiments as described in the Design section 4. The two primary areas of analysis are the false positive rates from each entropy calculation method and the higher order statistics of the entropy values themselves. It is important to note that no false negatives were found in any of the experiments, and as such, is not considered a part of the trade-offs inherent to each method in this project.

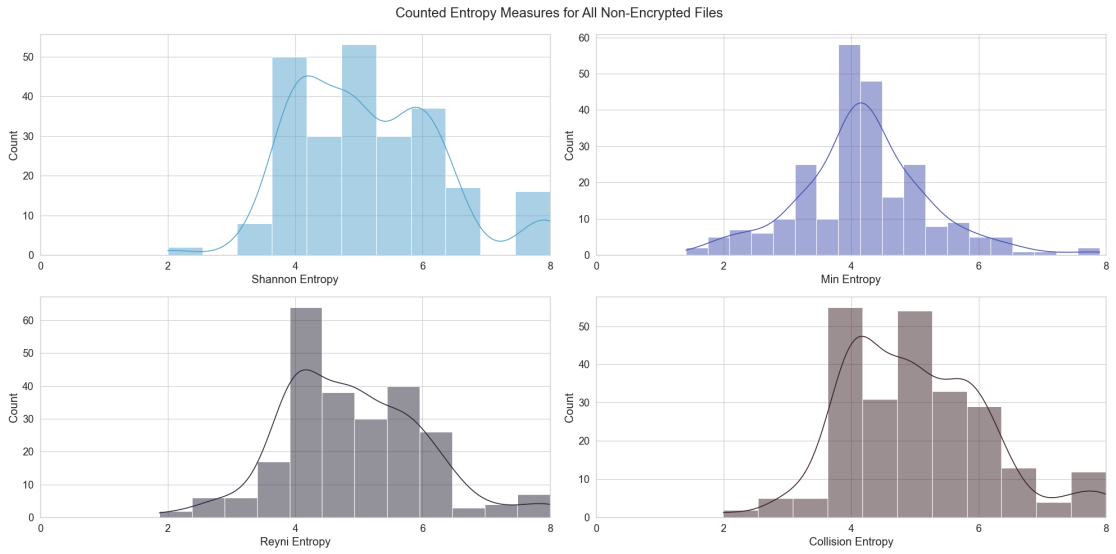

Figure 1: Counted entropy measures for all files

In figure 1, the distribution of entropy values for each non-encrypted file is shown by entropy calculation method. The presence of false positives for each method is immediately apparent, but this will be discussed in more detail in the following subsection. What is more interesting is that all calculation methods except for Min entropy show a similar shape to their distribution. This shape should be considered when later comparing the false positive rates of each method, as it's specific structure may be indicative of common pitfalls in the detection methods.

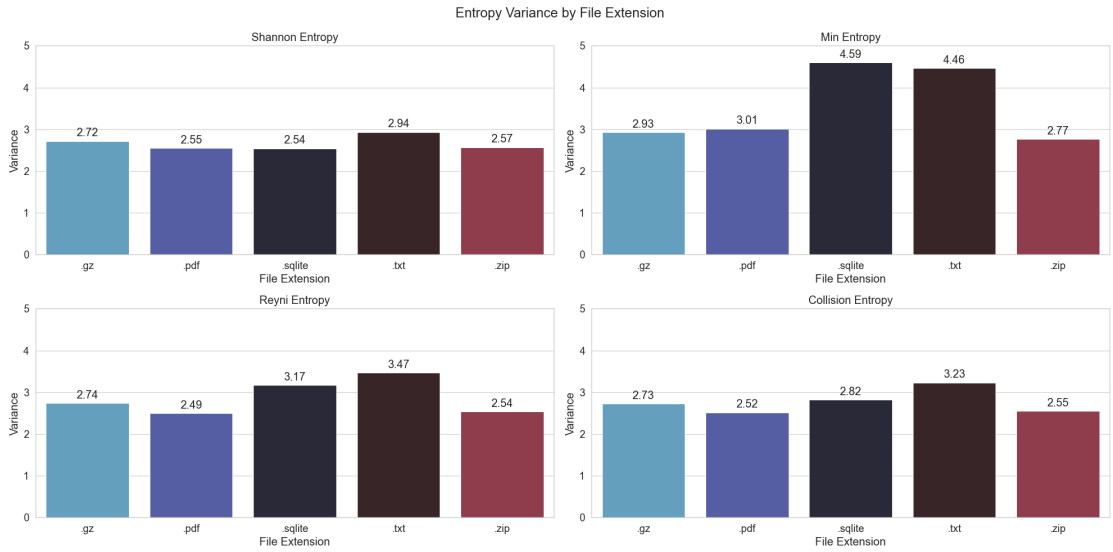

Figure 2: Entropy variance by file extension

Figure **??** shows how much each entropy calculation varies for each file type. This is inclusive of both encrypted and non-encrypted files, which helps to reveal how the byte-level reflected characteristics of the file format impact efficacy of a given entropy calculation. Generally, the range of variances is consistent across all file types and entropy calculation methods, except for a significant increase in variance for the Min entropy of .txt and .sqlite files. Further analysis is needed to determine the cause of this increase, but it is likely due to the fact that these file types are more likely to contain human-readable text, which is more likely to have a higher Min entropy value.

5.1 False Positives

The false positive counts for each entropy calculation method are shown in figure 3. Shannon entropy has the highest false positive count, and Min entropy has the lowest. Reyni and Collision have similar false positive rates and are both closer to Shannon entropy than Min entropy.

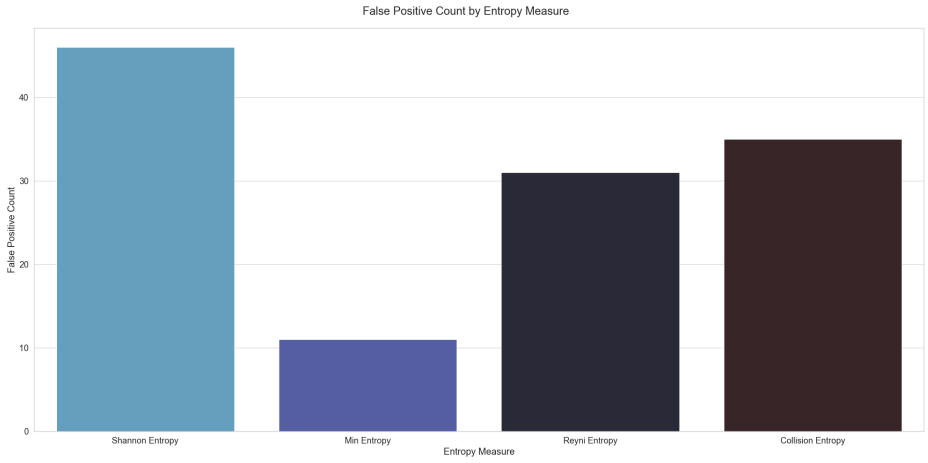

Figure 3: False positive counts for all entropy measures

Figure 4 shows the false positive counts for each file type, broken down by entropy calculation method. In line with the intuitive interpretation of figure **??** from earlier, Min entropy does not have any false positives for .txt or .sqlite files. Each calculation method saw similar struggles with .pdf, .zip, and .tar.gz files, but Shannon entropy was the only method that struggled with .sqlite files.

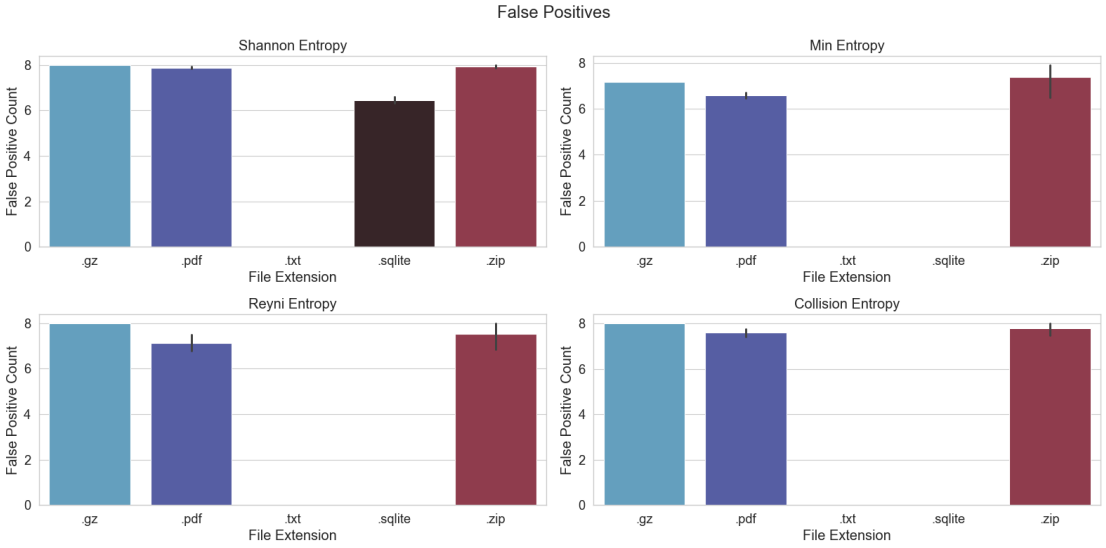

Figure 4: False positive counts by file extension

5.2 Higher Order Statistics

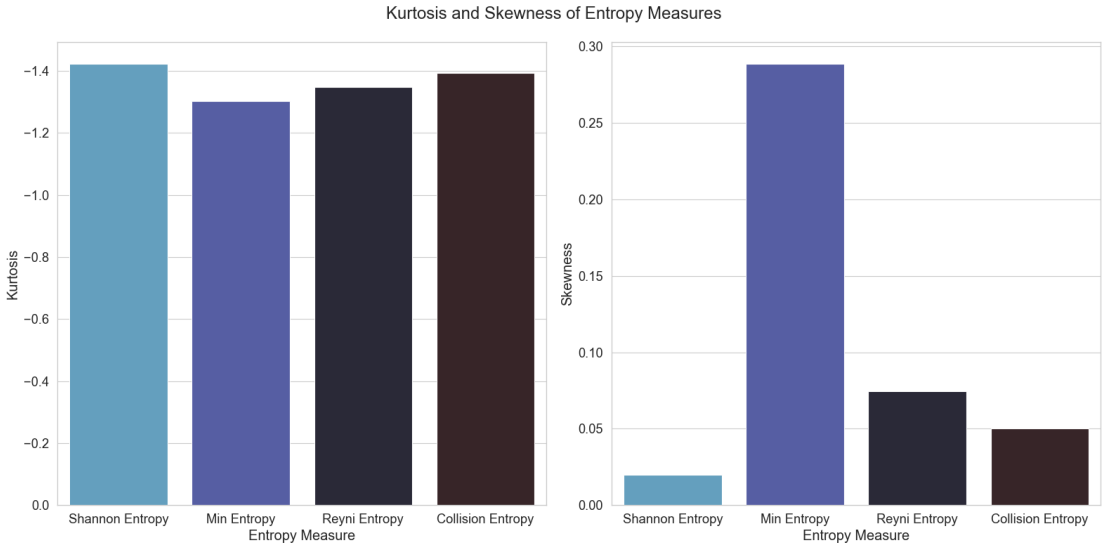

Figure 5: Kurtosis and skewness of entropy measures

In figure 5, the kurtosis and skewness of each entropy calculation method is shown. All entropy calculation methods show similar kurtosis values, and all but Min entropy show similar skewness values. The kurtosis values are all negative, which indicates that the distributions are platykurtic,

or flatter than a normal distribution[20]. The skewness values are all positive, which indicates that the distributions are positively skewed, or skewed to the right[20]. While Shannon entropy's skewness is significantly lower than the other methods, it is still positive. The high skewness for Min entropy is worth further investigation, but this is outside the scope of this project.

6 Conclusion

The results of this project reiterate two common themes on the topic of encryption detection: (1) current methods are not perfect, and (2) there is no one-size-fits-all solution. But considering the frequent use of combined approaches in other studies[23, 17, 30, 11], the methods shown in this project may be useful in combination with one another – all without introducing significant computational overhead. The results of this project also show that the entropy calculation method used by Autopsy[3] produces a higher false positive rate than the other methods tested. Finally, the emphasis on higher order statistics in this project shows the need for further research into their use in encryption detection methods.

7 Glossary

- **Advanced Encryption Standard (AES)** - A symmetric block cipher that is used to encrypt data
- **Autopsy** - A digital forensics platform
- **Base64** - An encoding method that converts binary data into a string of ASCII characters
- **Blowfish** - A symmetric block cipher that is used to encrypt data
- **ChaCha20** - A symmetric stream cipher that is used to encrypt data
- **Collision Entropy** - A method of calculating entropy that is based on the number of collisions in a file
- **Entropy** - A measure of a given system's disorder
- **Hash** - A fixed-length output produced by a hash function
- **Hash Function** - A function that takes in an input and produces a fixed-length output
- **MD5** - A hashing method that produces a 128-bit hash value
- **Min Entropy** - A method of calculating entropy that is based on the minimum probability of a byte value occurring
- **Quasi-Newton Method** - A method of finding the roots of a function that uses an approximation of the inverse of the function's Hessian matrix
- **Ransomware** - A type of malware that encrypts a victim's files and demands a ransom payment in exchange for the decryption key
- **Reyni Entropy** - A method of calculating entropy that is based on the probability of a byte value occurring and a user-defined parameter
- **Salsa20** - A symmetric stream cipher that is used to encrypt data
- **Secure Hash Algorithm (SHA)** - A family of hashing methods that produce hash values of varying lengths
- **Shannon Entropy** - A method of calculating entropy that is based on the probability of a byte value occurring
- **Symmetric Block Cipher** - An encryption method that uses the same key to encrypt and decrypt data, and splits the data into blocks of a fixed size before encrypting it
- **Symmetric Stream Cipher** - An encryption method that uses the same key to encrypt and decrypt data, and encrypts the data one byte at a time

References

[1] Administration, S. S. Baby names 2015. https://www.ssa.gov/oact/babynames/limits.html. Accessed: 2023-11-14.

[2] Aeberhard, S., and Forina, M. Wine. UCI Machine Learning Repository, 1991. DOI: https://doi.org/10.24432/C5PC7J.

[3] Autopsy. Autopsy source code. https://github.com/sleuthkit/autopsy/blob/develop/Core/src. Accessed: 2023-09-05.

[4] Becker, B., and Kohavi, R. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[5] Bell, T. Canterbury corpus. http://corpus.canterbury.ac.nz/. Accessed: 2023-11-14.

[6] Bernstein, D. J. Chacha, a variant of salsa20. https://cr.yp.to/chacha/chacha-20080128.pdf, 2008. Accessed: 2023-10-12.

[7] Bernstein, D. J. *The Salsa20 Family of Stream Ciphers*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 84–97.

[8] Boyd, S., and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, March 2004.

[9] Bureau, U. C. Census 2000 surnames. https://www.census.gov/topics/population/genealogy/data/2000_surnames.html. Accessed: 2023-11-14.

[10] California, T. California peninsula city official salaries. https://publicpay.ca.gov/. Accessed: 2023-11-14.

[11] Davies, S. R., Macfarlane, R., and Buchanan, W. J. Differential area analysis for ransomware attack detection within mixed file datasets. *CoRR abs/2106.14418* (2021).

[12] empty. 20 newsgroups dataset, empty.

[13] Fisher, R. Iris. UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C56C76.

[14] Hoffstein, J., Pipher, J., and Silverman, J. *An Introduction to Mathematical Cryptography*, 1 ed. Springer Publishing Company, Incorporated, 2008.

[15] Janosi, A., Steinbrunn, W., Pfisterer, M., and Detrano, R. Heart Disease. UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C52P4X.

[16] Josefsson, S. The Base16, Base32, and Base64 Data Encodings. RFC 4648, Oct. 2006.

[17] Lee, K., Lee, J., Lee, S.-Y., and Yim, K. Effective ransomware detection using entropy estimation of files for cloud services. *Sensors 23*, 6 (2023).

[18] Li, W. On the relationship between complexity and entropy for markov chains and regular languages. *Complex Syst. 5* (1991).

[19] Manjula, R., and Anitha, R. Identification of encryption algorithm using decision tree. In *Advanced Computing* (Berlin, Heidelberg, 2011), N. Meghanathan, B. K. Kaushik, and D. Nagamalai, Eds., Springer Berlin Heidelberg, pp. 237–246.

[20] Menezes, A. J., Vanstone, S. A., and Oorschot, P. C. V. *Handbook of Applied Cryptography*, 1st ed. CRC Press, Inc., USA, 1996.

[21] N.I.S.T. Fips pub 197-upd1 advanced encryption standard (aes). Tech. Rep. FIPS PUB 197-upd1, National Institute of Standards and Technology, November 2001.

[22] N.I.S.T. Fips pub 180-4 secure hash standard (shs). Tech. Rep. FIPS PUB 180-4, National Institute of Standards and Technology, August 2015.

[23] Pont, J., Arief, B., and Hernandez-Castro, J. Why current statistical approaches to ransomware detection fail. In *Information Security* (Cham, 2020), W. Susilo, R. H. Deng, F. Guo, Y. Li, and R. Intan, Eds., Springer International Publishing, pp. 199–216.

[24] Postel, J. ASCII format for network interchange. RFC 20, Oct. 1969.

[25] Rivest, R. The MD5 Message-Digest Algorithm. RFC 1321, Apr. 1992.

[26] Sammons, J. *The Basics of Digital Forensics: The Primer for Getting Started in Digital Forensics*, 2nd ed. Syngress, 2014.

[27] Schneier, B. Description of a new variable-length key, 64-bit block cipher (blowfish). In *Fast Software Encryption, Cambridge Security Workshop* (Berlin, Heidelberg, 1993), Springer-Verlag, p. 191–204.

[28] The Autopsy Project. Autopsy.

[29] USGS. Earthquake data. https://earthquake.usgs.gov/earthquakes/search/. Accessed: 2023-11-14.

[30] Zainodin, M., Zakaria, Z., Hassan, R., and Abdullah, Z. Entropy based method for malicious file detection. *JOIV : International Journal on Informatics Visualization 6* (12 2022), 856.

8 Appendix

8.1 Testing Data File Descriptions

- **20news-18828.tar.gz**: a collection of around 20,000 different news documents from Ken Lang at Carnegie Mellon University[12].
- **adult.zip**: a dataset from the UCI machine learning repository containing information about adults from the 1994 census[4].
- **applied_cryptography.pdf**: a PDF copy of the book Applied Cryptography by Bruce Schneier[20].
- **alice29.txt**: a text file containing the first chapter of Alice's Adventures in Wonderland, collected from the Canterbury Corpus[5].
- **asyoulik.txt**: a text file containing the first chapter of As You Like It, collected from the Canterbury Corpus[5].
- **bible.txt**: a text file of the King James Bible, collected from the Canterbury Corpus[5].
- **boyd_optimization.pdf**: a PDF version of Boyd and Vandenberghe's book on Convex Optimization[8].
- **canterbury.zip**: a zipfile collection of 11 text files from the Canterbury Corpus[5].
- **census2000names.sqlite**: an SQLite database containing the 2000 US Census data on names[9].
- **chacha-2008128.pdf**: a PDF version of the paper that introduced the ChaCha stream cipher[6].
- **differential_area_analysis.pdf**: a PDF copy of a research study on the use of differential area analysis for malware detection[11].
- **E_coli.txt**: a text file containing the DNA sequence of the E. Coli bacteria, collected from the Canterbury Corpus[5].
- **encryption_identification_decision_tree.pdf**: a PDF copy of a research study on the use of decision trees for encryption identification[19].
- **entropy_based_malicious_file_detection.pdf**: a PDF copy of a research study on the use of entropy for malware detection[30].
- **heart.zip**: a dataset from the UCI machine learning repository containing information about patients with heart disease[15].
- **iris.zip**: a dataset from the UCI machine learning repository containing information about different species of iris flowers[13].
- **lab2.pdf**: a PDF copy of my lab2 report for this class. Citation not needed.
- **lab4.pdf**: a PDF copy of my lab4 report for this class. Citation not needed.
- **lab5.pdf**: a PDF copy of my lab5 report for this class. Citation not needed.

- **lcet10.txt**: a text file containing the first chapter of Les Misérables, collected from the Canterbury Corpus[5].
- **peninsula_publicpay.sqlite**: an SQLite database containing information about public employee salaries in the state of California[10].
- **plrabn12.txt**: a text file containing the first chapter of Pride and Prejudice, collected from the Canterbury Corpus[5].
- **ssa-babynames-for-2015.sqlite**: an SQLite database containing the 2015 US Census data on names[1].
- **usgs-lower-us.sqlite**: an SQLite database containing earthquake data from the US Geological Survey[29].
- **why_statistical_approaches_fail.pdf**: a PDF copy of a research study on the use of statistical approaches for encryption detection[23].
- **wine.zip**: a dataset from the UCI machine learning repository containing information about different types of wine[2].
- **world192.txt**: a text file containing the first chapter of The World Set Free, collected from the Canterbury Corpus[5].

8.2 Entropy Calculation Methods

8.2.1 Shannon Entropy

Shannon entropy is the most common entropy calculation method, and is defined as follows[14, 20]:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i) \tag{1}$$

For the scope of this project, the base of the logarithm was always 2.

8.2.2 Min Entropy

Min entropy is defined as follows[14, 20]:

$$H_\infty(X) = -\log_b \max_{i=1}^{n} p(x_i) \tag{2}$$

Again, the base of the logarithm was always 2 for caclulations performed in this project.

8.2.3 Reyni Entropy

Reyni entropy is defined as follows[14, 20]:

$$H_\alpha(X) = \frac{1}{1-\alpha} \log_b \sum_{i=1}^{n} p(x_i)^\alpha \tag{3}$$

This form of entropy is a generalization of Shannon entropy, and the use of the parameter α allows for this calculation to become equivalent to the other entropy calculation methods. When α is equal to 1, Reyni entropy is equivalent to Shannon entropy. When α is equal to 2, Reyni

entropy is equivalent to Collision entropy. When α is equal to ∞, Reyni entropy is equivalent to Min entropy. When α is equal to 0, Reyni entropy is equivalent to Max entropy.

For this project, the value of α used was always 3.

8.2.4 Collision Entropy

Collision entropy is defined as follows[14, 20]:

$$H_2(X) = -\log_b \sum_{i=1}^{n} p(x_i)^2 \tag{4}$$

As in other methods, the base of the logarithm was always 2.