International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0223
e ISSN: 2584-2854
Volume: 03
Issue: 04 April 2025
Page No: 1368 - 1375

Encrypted Algorithm Detector in Cryptography

Vetri Selvan M[1],Gajendran N[2], Gururajan K[3] ,Hari Pranava M D[4]
[1]Assistant Professor, Dept. of AI&DS, Panimalar Engineering College, Chennai, India.
[2,3,4]UG Scholar, Dept. of AI&DS, Panimalar Engineering College, Chennai, India.
Email ID: *vetrinelson7@gmail.com[1], gajendrann2004@gmail.com [2], gurukolanjirajan1603k@gmail.com[3], haripranav@gmail.com[4].*

Abstract
With the increasing reliance on digital communication and data storage, the need for secure cryptographic algorithms has become paramount. However, the rise of encrypted malware and unauthorized encrypted communications poses a significant challenge to cybersecurity. This paper presents an Encrypted Algorithm Detector using Cryptography, a system designed to verifying and identify encryption algorithms embedded within data transmissions or software applications. The proposed system leverages cryptographic fingerprinting techniques, statistical analysis, and machine learning models to detect and classify encryption methods such as AES, RSA, DES, and other cryptographic schemes. The detector functions by analyzing entropy levels, byte distribution, and frequency patterns to distinguish between encrypted and non-encrypted data. This approach can be applied in cybersecurity for malware detection, forensic investigations, and preventing unauthorized encrypted data transmission in secure environments. The implementation of this system contributes to strengthening cybersecurity measures by enabling early detection of encrypted threats and ensuring compliance with security protocols.

Keywords: *Cryptography, Encryption Algorithm Detection, Cybersecurity, Machine Learning, Cryptographic Fingerprinting.*

1. Introduction
Cryptography plays a crucial role in securing digital communications, protecting sensitive data, and ensuring confidentiality, integrity, and authenticity. Modern encryption algorithms such as AES, RSA, and DES are widely used in various applications, including secure messaging, online banking, and cloud computing. However, while encryption enhances security, it also presents challenges in cybersecurity when used maliciously, such as in encrypted malware, ransomware, and covert communication channels. The increasing use of encryption by cybercriminals necessitates the development of sophisticated detection mechanisms to identify encrypted algorithms embedded within data streams, files, or applications. Traditional security solutions often struggle to distinguish between legitimate and maliciously encrypted content, leading to potential data breaches, ransomware attacks, and undetected threats. This paper introduces an Encrypted Algorithm Detector using Cryptography, a system designed to dissect and identify encryption algorithms within digital data. The system employs cryptographic fingerprinting, entropy analysis, and machine learning techniques to classify encryption methods accurately. By examining byte distributions, randomness, and statistical properties of encrypted data, the proposed approach enhances cybersecurity by enabling proactive detection of encrypted threats. By implementing this detection system, security analysts and forensic experts can improve threat identification, prevent data breaches, and ensure regulatory compliance. This research contributes to the evolving field of cryptographic security by bridging the gap between encryption technology and cybersecurity defense mechanisms [1][2].

2. Literature Review
Cryptographic algorithm detection has been a significant area of research in cybersecurity, particularly in identifying encryption techniques used in both legitimate and malicious applications. Various studies have explored cryptographic

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0223
e ISSN: 2584-2854
Volume: 03
Issue: 04 April 2025
Page No: 1368 - 1375

fingerprinting, entropy analysis, and machine learning-based classification for identifying encrypted data. This section reviews existing literature related to encryption detection, cryptographic analysis, and machine learning applications in cybersecurity [3].

2.1 Cryptographic Algorithm Detection

Several studies have focused on detecting encryption algorithms based on statistical and structural properties of encrypted data.
According to M. Bello et al. (2019), cryptographic algorithm identification can be achieved using entropy and byte distribution analysis, as encrypted data exhibits high randomness and uniformity. Sharma et al. (2020) proposed a signature-based approach for identifying encryption techniques, where unique characteristics of algorithms like AES, RSA, and DES were analyzed for detection.

2.2 Entropy-Based Detection Methods

Entropy analysis is a widely used method for detecting encryption, as encrypted data generally has higher entropy than plaintext. Shannon (1948) introduced entropy as a measure of uncertainty in information theory, which later became a foundation for encryption detection. L. Wu et al. (2021) developed an entropy-based model for distinguishing encrypted and compressed files, proving that encrypted data exhibits distinct entropy characteristics.

2.3 Machine Learning in Encryption Detection

Machine learning has emerged as an effective tool in detecting encryption patterns and classifying algorithms. Wang et al. (2018) implemented a neural network model for encryption identification, demonstrating that deep learning can achieve high accuracy in recognizing cryptographic schemes. Zhang and Chen (2022) applied Support Vector Machines (SVM) and Random Forest classifiers to detect encrypted malware traffic, showing promising results in identifying encrypted cyber threats.

2.4 Encrypted Malware and Ransomware Detection

The use of encryption in malware, particularly ransomware, has driven research in detecting unauthorized encryption activities. Singh et al. (2021) explored ransomware detection using entropy and machine learning techniques, where abnormal encryption behaviors were flagged. Gupta et al. (2023) further expanded on automated malware analysis using deep learning to identify hidden encryption routines within malicious software.

2.5 Cryptographic Traffic Analysis

In network security, detecting encrypted traffic is a crucial challenge. Bujlow et al. (2017) proposed a classification system for distinguishing encrypted traffic from regular internet data. N. Patel et al. (2020) examined the use of deep packet inspection (DPI) and machine learning to identify encryption protocols in real-time network monitoring.

2.6 Summary of Findings

The reviewed literature highlights several key insights:

- Entropy and statistical analysis are fundamental in detecting encrypted data.
- Machine learning approaches significantly improve the classification of cryptographic algorithms.
- Identifying encrypted malware and ransomware remains a critical challenge in cybersecurity.
- Traffic analysis techniques can help distinguish between encrypted and unencrypted network transmissions.

While existing research has made significant strides in encryption detection, there is still a need for a comprehensive system that integrates multiple techniques for accurate cryptographic algorithm detection. The proposed Encrypted Algorithm Detector using Cryptography builds upon these findings by combining entropy analysis, cryptographic fingerprinting, and machine learning [4].

3. Objectives

The primary objective of this research is to develop an Encrypted Algorithm Detector using Cryptography that can accurately identify encryption algorithms embedded within digital data. The system aims to enhance cybersecurity by distinguishing between various encryption methods and detecting unauthorized or maliciously encrypted data.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0223
e ISSN: 2584-2854
Volume: 03
Issue: 04 April 2025
Page No: 1368 - 1375

3.1 The Specific Objectives of This Research Are

- To develop a cryptographic algorithm detection framework – Implement a system capable of identifying different encryption algorithms, such as AES, RSA, DES, and Blowfish, based on their unique characteristics.
- To analyze encrypted data properties – Utilize entropy analysis, byte distribution, and randomness tests to differentiate encrypted data from plaintext or compressed files.
- To employ machine learning for classification – Train and test machine learning models to recognize patterns in encrypted data and classify different cryptographic algorithms with high accuracy.
- To enhance cybersecurity through proactive detection – Detect unauthorized encrypted data transmissions, encrypted malware, and ransomware to strengthen cybersecurity threat in respective area.
- To provide a forensic tool for cryptographic analysis – Assist cybersecurity professionals and digital forensic analysts in identifying and investigating encrypted content in various environments, including networks and file systems.

By achieving these objectives, the proposed system aims to contribute to modern cybersecurity solutions by providing an efficient method for detecting and analyzing encrypted algorithms.

4. Scope

The **Encrypted Algorithm Detector using Cryptography** focuses on the identification and classification of encryption algorithms embedded in digital data, with applications in cybersecurity, digital forensics, and malware analysis. The scope of this study is outlined as follows:

4.1 Encryption Algorithm Detection

- Identification of commonly used encryption algorithms such as AES, RSA, DES, Blowfish, and ChaCha20.
- Detection of encrypted data by analyzing entropy, randomness, and byte distribution.
- Differentiation between encrypted, compressed, and plaintext data.

4.2 Cybersecurity Applications

- Detection of maliciously encrypted data used in ransomware, encrypted malware, and hidden communication channels.
- Enhancing network security by identifying encrypted traffic patterns.
- Preventing unauthorized encryption in secure environments.

4.3 Machine Learning Integration

- Implementation of machine learning models such as Support Vector Machines (SVM), Random Forest, and Deep Neural Networks to classify encryption algorithms.
- Training the models using datasets containing encrypted and nonencrypted data samples.
- Evaluating performance metrics such as accuracy, precision, recall, and F1-score.

4.4 Digital Forensics & Law Enforcement

- Assisting forensic investigators in identifying encrypted files and detecting illicit data-hiding techniques.
- Providing insights into encryption usage in cybercrime investigations.
- Supporting compliance with legal and regulatory frameworks regarding encryption.

4.5 Limitations

- The system focuses primarily on identifying encryption algorithms and does not aim to decrypt encrypted data.
- Detection may be affected by obfuscation techniques used by attackers to disguise encryption.
- Accuracy depends on the quality and diversity of the training dataset for machine learning models.

Conclusion

The proposed system provides a powerful tool for detecting and classifying encryption algorithms, with applications in cybersecurity, digital forensics, and malware analysis [5]. By leveraging cryptographic analysis and machine learning, it enhances security measures against maliciously encrypted threats and unauthorized data encryption.

5. Proposed System

The **Encrypted Algorithm Detector using Cryptography** is designed to identify and classify encryption algorithms embedded in digital data. The

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0223
e ISSN: 2584-2854
Volume: 03
Issue: 04 April 2025
Page No: 1368 - 1375

system leverages cryptographic analysis techniques and machine learning models to enhance detection accuracy. It is intended to be used in cybersecurity, digital forensics, and malware analysis to identify encrypted threats, unauthorized encryption usage, and cryptographic signatures [7].

5.1 System Architecture

The proposed system consists of the following major components:

5.1.1 Data Preprocessing Module

- Collects and processes input data, including encrypted files, network traffic, and malware samples.
- Performs feature extraction by analyzing entropy, randomness, and byte frequency distributions.

5.1.2 Cryptographic Analysis Module

- Uses statistical techniques to detect encrypted data based on high entropy and randomness.
- Identifies cryptographic fingerprints associated with different encryption algorithms.

5.1.3 Machine Learning-Based

5.1.3.1 Classifier

- Trains models such as Support Vector Machine (SVM), Random Forest, and Deep Neural Networks to classify encryption types.
- Uses labeled datasets containing encrypted and non-encrypted data for model training and validation.

5.1.3.2 Detection and Reporting Module:

- Compares extracted features with known encryption patterns.
- Generates reports detailing detected encryption algorithms and potential threats.

5.2 Working Mechanism

- **Data Collection**: The system collects input data from files, network traffic, or system memory.
- **Feature Extraction**: Entropy, byte frequency, and cryptographic signatures are analyzed.
- **Algorithm Detection**: The extracted features are compared against known cryptographic patterns.
- **Machine Learning Classification**: The trained model predicts the encryption algorithm used.
- **Result Interpretation**: The system provides a detailed report on detected encryption algorithms and potential risks.

5.3 Advantages of the Proposed System

- **High Accuracy**: Combines statistical and machine learning approaches for precise encryption detection.
- **Automation**: Reduces the need for manual cryptographic analysis.
- **Real-Time Detection**: Can be integrated with cybersecurity tools for real-time monitoring.
- **Versatility**: Detects various encryption algorithms used in secure communication, ransomware, and malware. Figure 1 shows Flowchart for Encrypted Algorithm.

6. Flowchart

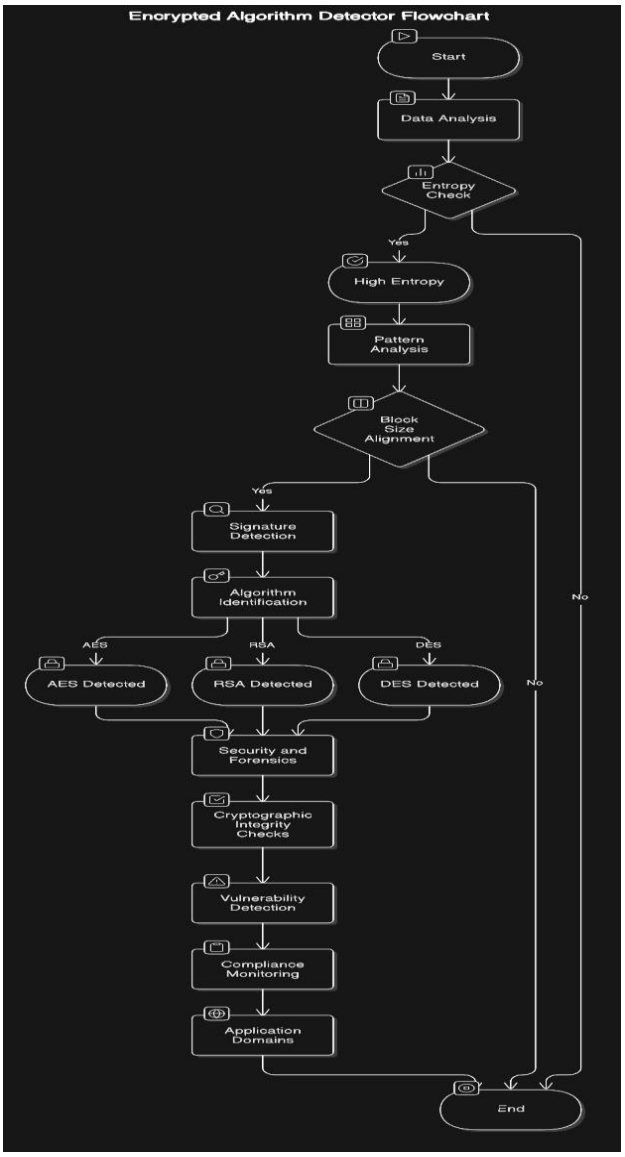

Figure 1 Flowchart for Encrypted Algorithm

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0223
e ISSN: 2584-2854
Volume: 03
Issue: 04 April 2025
Page No: 1368 - 1375

7. Modules

7.1 Cryptography Module

Purpose: Provides simple and secure methods to handle encryption algorithms, and can help in detecting or analyse encryption patterns.

7.1.1 Features

- Implement encryption schemes like AES, RSA, DES, and more.
- Provides secure key management and decryption mechanisms.
- Can detect known encryption algorithms by analyse ciphertext structures.

7.1.2 Usage

- Encrypt and decrypt data with different algorithms (AES, RSA, etc.).
- Check block sizes, padding methods, and initialization vectors (IVs) to detect encryption.

7.2 Pycryptodome Module

Purpose: A Python library for cryptography that includes a variety of encryption algorithms like AES, RSA, DES, and more.

7.2.1 Features

- Implements common cryptographic algorithms.
- Allows you to analyze cipher-text patterns.
- Supports encryption and decryption using symmetric and asymmetric methods.

7.2.2 Usage

- Detect encryption type by testing ciphertexts against common algorithm parameters (block sizes, padding, etc.).

7.3 Hashlib Module

Purpose: A built-in Python library for hashing algorithms (MD5, SHA-256, etc.), useful for detecting hashed data.

7.3.1 Features

- Detects hashed data which may have been used for integrity checking or digital signatures.
- Analyzes hash lengths and characteristics to identify the hash type.

7.3.2 Usage

- Check for hashed data to detect encryption or verify data integrity.

7.4 Binwalk Module

Purpose: A tool for analyzing binary files and detecting embedded encryption or compression algorithms.

7.4.1 Features

- Detects file structures, such as known compression and encryption signatures.
- Scans binary files to find encryption signatures or even hidden data [9].

7.4.2 Usage

- Analyze binary files for embedded encryption, compressed data, or other cryptographic schemes.

7.5 Zlib (Compression Detection)

Purpose: While not strictly for encryption, compression algorithms (like ZIP) are often used before encryption, making compression detection important.

7.5.1 Features

- Helps in detecting compression formats (e.g., GZIP, ZIP).
- Can be used to identify if compression was applied before encryption.

7.5.2 Usage

- Detect common compression schemes used before encryption.

7.6 Numpy & Scipy (Statistical Analysis)

Purpose: Libraries for numerical analysis that can help in detecting the entropy and randomness of encrypted data.

7.6.1 Features

- Helps in calculating entropy, which can be used to detect randomness (a typical characteristic of encrypted data).
- Can be used to analyze byte distributions and detect deviations from normal patterns.

7.6.2 Usage

- Calculate entropy to detect encrypted or compressed data.

7.7 Scikit-Learn (Machine Learning for Encryption Detection)

Purpose: Machine learning can be used for detecting encrypted algorithms based on feature extraction from data, which may include entropy, byte frequency, and statistical analysis.

7.7.1 Features

- Can train models to detect encrypted algorithms based on features like byte distributions, entropy, and more.
- Useful for creating an automated detector of various cryptographic algorithms.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0223
e ISSN: 2584-2854
Volume: 03
Issue: 04 April 2025
Page No: 1368 - 1375

7.7.2 Usage

- Train a machine learning model to classify encrypted data based on pre-extracted features.

7.8 Pycrypto (Deprecated)

Purpose: **pycrypto** is a library that provides various cryptographic functionalities, but it is no longer actively maintained. Use **pycryptodome** as a more modern replacement.

7.8.1 Features

- Cryptographic algorithms like AES, RSA, DES, and more.
- Can still be used to identify encryption schemes if you're working with older systems.

7.8.2 Usage

- Detect and break basic encryption algorithms, especially older ones like DES.

7.9 Methodology for the Detection System

- **Step 1**: **Input Data Analysis** – Check if the data is in binary format, which is often indicative of encryption.
- **Step 2**: **Statistical Tests** – Calculate the entropy of the data. Encrypted data typically shows high entropy, while plaintext data is often more predictable.
- **Step 3**: **Compression Check** – Use **zlib** or **binwalk** to check if the data is compressed, which is commonly done before encryption.
- **Step 4**: **Pattern Matching** – Check for known encryption algorithm signatures (AES, DES, RSA, etc.).
- **Step 5**: **Decryption Attempts** – Use libraries like **pycryptodome** or **cryptography** to attempt decryption with common algorithms.
- **Step 6**: **Machine Learning** – Optionally, use **scikit-learn** to train a model on various features and classify the encryption algorithm.

8. Performance Analysis

The effectiveness and efficiency of such a detector can be measured through various performance metrics. Below, I will outline the key performance metrics, factors to consider, and methods for performance analysis in this context.

8.1 Key Performance Metrics

8.1.1 Accuracy

Definition: Measures the percentage of correctly identified encryption algorithms compared to the total number of encrypted data samples.

Importance: High accuracy is critical to ensure the detector correctly identifies the encryption algorithm in most cases, minimizing false positives and negatives.

Evaluation Method:

- Use a labeled dataset of encrypted data (e.g., AES, RSA, DES) and test the detector on this dataset [10].
- Calculate accuracy using the formula: Accuracy=Correctly Identified SamplesTotal Samples×100\text {Accuracy} = \frac {\text {Correctly Identified Samples}} {\text {Total Samples}} \times 100Accuracy=Total SamplesCorrectly Identified Samples×100
- **Example**: If the detector correctly identifies 90 out of 100 encrypted samples, its accuracy is 90%.

8.1.2 Detection Time (Latency)

Definition: Measures how quickly the detector can identify the encryption algorithm or detect if the data is encrypted.

Importance: Low latency is important in real-time applications, such as network security monitoring, where fast detection is crucial.

Evaluation Method:

- Measure the time taken from input to detection for each sample or file.
- Calculate the average detection time over multiple samples.

Example: If the detector takes 2 seconds to analyze a 1 MB encrypted file, it is important to see if this performance holds for larger files or in a batch processing scenario.

8.1.3 False Positive Rate

Definition: Measures the percentage of non-encrypted data that is incorrectly identified as encrypted.

Importance: Minimizing false positives is crucial to avoid unnecessary processing or alerts.

Evaluation Method:

- Test the detector with non-encrypted data and calculate the percentage of samples incorrectly flagged as encrypted.

Formula:

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0223
e ISSN: 2584-2854
Volume: 03
Issue: 04 April 2025
Page No: 1368 - 1375

False Positive Rate=False PositivesTotal Non-encrypted Samples×100\text{False Positive Rate} = \frac{\text{False Positives}}{\text{Total Non-encrypted Samples}} \times 100False Positive Rate=Total Non-encrypted SamplesFalse Positives×100

Example: If the detector flags 3 out of 100 non-encrypted files as encrypted, the false positive rate would be 3%.

8.1.4 False Negative Rate

Definition: Measures the percentage of encrypted data that is incorrectly identified as non-encrypted.

Importance: A low false negative rate ensures that the detector doesn't miss encrypted data, which is important in security applications.

Evaluation Method:

- Test the detector with encrypted data and calculate the percentage of samples that are not flagged as encrypted.
- **Formula:** False Negative Rate =False Negatives Total Encrypted Samples × 100 \ text{False Negative Rate} = \frac{\text{False Negatives}}{\text{Total Encrypted Samples}} \times 100False Negative Rate=Total Encrypted SamplesFalse Negatives×100

Example: If 2 out of 100 encrypted files are mistakenly identified as non-encrypted, the false negative rate is 2%.

8.1.5 Precision and Recall

Precision: The percentage of correctly detected encrypted data out of all the data flagged as encrypted.

Precision=True PositivesTrue Positives+False Positives\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}Precision=True Positives+False Positives True Positives

Recall: The percentage of correctly detected encrypted data out of all the encrypted data samples. Recall = True Positives True Positives + False Negatives \text {Recall} = \frac {\text{True Positives}}{\text{True Positives} + \text{False Negatives}}Recall=True Positives+False Negatives True Positives

Importance: Precision and recall give a more balanced view of the model's effectiveness, especially when working with imbalanced datasets.

8.2 Factors Affecting Performance

8.2.1 Size and Type of Data

Larger Files: The time taken to detect encryption increases with the size of the files being analyzed. Larger datasets may take longer to process, which impacts latency.

File Formats: Encrypted data embedded within common file formats (e.g., PDFs, images, or archives) may require different processing steps compared to simple binary files.

8.2.2 Encryption Algorithm Complexity

Symmetric vs Asymmetric Encryption: Algorithms like AES (symmetric) generally have faster decryption times compared to asymmetric algorithms like RSA, which require more complex computations (especially with large keys).

Padding and Block Cipher Modes: The padding mechanism and block cipher mode (e.g., CBC, ECB) used can affect the patterns detected during the analysis, adding complexity to the detection process.

8.2.3 Statistical and Cryptographic Features

Entropy Analysis: Encrypted data typically shows high entropy, but some encryption algorithms (like weak ciphers) may not produce uniformly high entropy. False positives may arise when analyzing entropy as a feature.

Compression Detection: Compression algorithms like ZIP or GZIP may obscure the encrypted data's structure, leading to challenges in detecting encryption if compression is detected first.

8.2.4 Detection Methodology

Signature-Based Detection: This approach relies on known patterns of encryption (e.g., AES block size, RSA key size). It may be fast but can be limited by the detector's ability to handle novel or unknown encryption algorithms.

Statistical and Entropy-Based Detection: Analyzing randomness and entropy levels can be effective but may not always be conclusive, especially in cases where the encryption is weak or uses a non-standard method.

Machine Learning Approaches: If using machine learning to detect encryption, the complexity of the model (e.g., neural networks, decision trees) will

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0223
e ISSN: 2584-2854
Volume: 03
Issue: 04 April 2025
Page No: 1368 - 1375

affect both training time and prediction time.

Reference

[1]. Menezes, A., van Oorschot, P., & Vanstone, S. (1996). Handbook of Applied Cryptography. CRC Press. **A foundational text on applied cryptography, providing detailed descriptions of various encryption algorithms and their characteristics.**

[2]. Python Software Foundation. (2021). Cryptography – Cryptographic Recipes. Retrieved from https://cryptography.io/en/latest/. **The official documentation for the cryptography library in Python, which provides a comprehensive guide on cryptographic algorithms and usage.**

[3]. Halevi, S., & Ristenpart, T. (2016). Detecting AES-like Encryption from Ciphertext. Journal of Cryptology, 29(2), 241–275. **A research paper exploring methods for detecting AES encryption patterns based on ciphertext analysis.**

[4]. Lowe, S. (2019). pycryptodome Documentation. Retrieved from https://www.pycryptodome.org. **Official documentation for the pycryptodome library, which provides implementations for several cryptographic algorithms, useful for encryption detection.**

[5]. Ferguson, N., Schneier, B., & Kohno, T. (2010). Cryptography Engineering: Design Principles and Practical Applications. Wiley. **A textbook that discusses the design and implementation of cryptographic systems, which can help in understanding how encryption algorithms behave and can be detected.**

[6]. Kaufman, C., Perlman, R., & Speciner, M. (2014). Network Security: Private Communication in a Public World. Prentice Hall. **This book provides an overview of encryption protocols used in network security and touches on detection techniques.**

[7]. Schneier, B. (2000). Secrets and Lies: Digital Security in a Networked World. Wiley. **A general book on security and cryptography, providing background information that can help understand the context of encryption algorithm detection.**

[8]. Song, D., & Wagner, D. (1999). Analysis of Block Cipher Modes of Operation. In Proceedings of the 3rd ACM Conference on Computer and Communications Security. ACM. **This paper explores different modes of operation for block ciphers (like AES), which can help in understanding how to detect different encryption schemes.**

[9]. Zhang, Y., & Xie, W. (2015). Automated Detection of Encrypted Traffic using Machine Learning Algorithms. International Journal of Network Security, 17(4), 397-405. **This paper investigates machine learning techniques to detect encrypted traffic, which can be analogous to detecting encrypted algorithms in data.**

[10]. Nielsen, A. (2013). Binwalk: A Tool for Analyzing Binary Files. Retrieved from https: // github.com / ReFirmLabs / binwalk. **Binwalk is a popular tool for analyzing binary files. It can detect compression and encryption signatures and can be used in the detection of encryption algorithms.**