National College of Ireland

Project Submission Sheet

Student Name:	Abhinandan Shrenik Digraje		
Student ID:	22220526		
Programme:	Master of Science in Cyber Security	**Year:**	2023-2024
Module:	Msc Research Practicum/Internship part 2		
Lecturer:	Eugene Mclaughlin		
Submission Due Date:	12-08-2024		
Project Title:	Research Paper Report		
Word Count:	8481		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature: Abhinandan Shrenik Digraje

Date: 11-08-2024

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year**. Any project/assignment not submitted will be marked as a fail.**

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

AI Acknowledgement Supplement

[Insert Module Name]

[Insert Title of your assignment]

Your Name/Student Number	Course	Date
NA	NA	

This section is a supplement to the main assignment, to be used if AI was used in any capacity in the creation of your assignment; if you have queries about how to do this, please contact your lecturer. For an example of how to fill these sections out, please click here.

AI Acknowledgment

This section acknowledges the AI tools that were utilized in the process of completing this assignment.

Tool Name	Brief Description	Link to tool
NA	NA	NA

Description of AI Usage

This section provides a more detailed description of how the AI tools were used in the assignment. It includes information about the prompts given to the AI tool, the responses received, and how these responses were utilized or modified in the assignment. **One table should be used for each tool used**.

[Insert Tool Name]	
NA	
NA	NA

Evidence of AI Usage

This section includes evidence of significant prompts and responses used or generated through the AI tool. It should provide a clear understanding of the extent to which the AI tool was used in the assignment. Evidence may be attached via screenshots or text.

Additional Evidence:

NA

Additional Evidence:

NA

Abstract

In cyber security, a critical issue being discussed is safety of files on their way through networks. These data may fall victim to unauthorized access or data decease where malicious data may be inserted in a file transferring from one support to another. It results in reduced safety of the data being transferred or compromised. To prevent malicious data insertion the thesis demonstrates use of magic number with advances encryption algorithms is adding the extra security layer over encryption. Therefore, existing and proposed means for ensuring safety of data are discussed by various scholars. With this system, a new system is proposed where additional safety of files is reached by combining use of magic numbers and methods like AES, DES, and HMAC for authentication. In addition, it incorporates manual and dynamic transforming magic numbers in the process of changing types of files. Also, the system employs K-means clustering and feature extraction means for improving accuracy of files classification according to the type, which is later used for modification. The system has an easy interface created by means of Tkinter for convenient and quick facility of needs. The evaluation of the system shows success in integration of various methods for safety of a file.

1 Introduction

1.1 Background

The digital era has irreversibly transformed our lives, work, and interaction. It unleashed incredible possibilities for us by enabling innovative tools to keep in touch, acquire vast information, and utilize cutting-edge technologies. However, such strides have also brought about massive challenges, including the issue of data security. Today, everything we do online depends on the security of personal and sensitive data. The latter includes our passwords, vital information, and even business data. Such data has turned into the backbone of the modern world, and as such, it requires permanently cautious measures to ensure its protection from unexpectedly disclosed (Malik & Abbas, 2023). Although taking some mitigation steps would be essential to enhance the level of data protection. Particularly, first and foremost, an effective encryption system should be involved. This process transforms data into code to prevent unauthorized access. Hence, even if someone intercepts the data, they cannot read it without the appropriate key. Furthermore, to prevent data security issues, putting into practice unique, fresh, and strong passwords and updating them on a regular basis would be effective. Additionally, another security layer could be a multi-factor authentication mechanism that demands more than one verification type. The collected information determined where the user attempts to log in(Nagaraj, Raju and Srinadth, 2015).

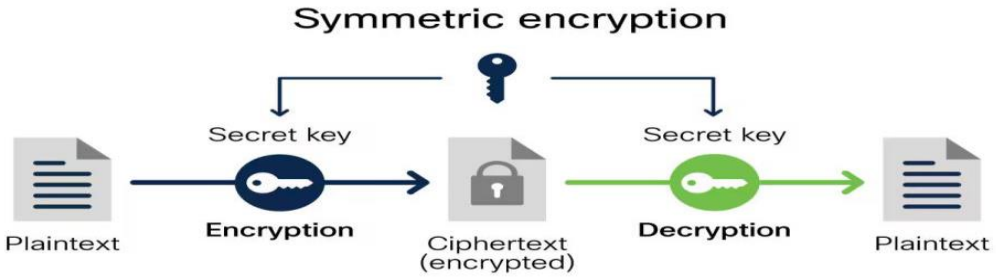

Figure (1): Symmetric Encryption and Decryption Process (What is encryption?, 2024)

Advanced Encryption Standard (AES) and magic numbers each help in so many ways in keeping our digital information safe. Data are encrypted using AES such that it is stored safely and only authorized people can read it. Think about it as the size of the lock used to lock up your data in a safe vault. This lock is never tampered with since the data will be transmitted as a sealed or encrypted box. Hash-based Message Authentication Code (HMAC) makes it impossible for others to tamper with our encrypted data. We can call it the seal on the box that has being locked in the vault. When the receiver receives the locked or sealed box, they will know that someone opened it or sealed it via the seal. Alternatively, magic numbers are used to verify or were created to be able to read these messages. Magic numbers are special codes that only software applications or tools can read which are found in a file or a message. This code is unique to that type of file or message, for example, a text or word file (Harba, 2017). When a message ends with a specific number to verify the script, the receiver will know that use the same 1 or 32 kilobytes to verify the message. The use of all magic numbers are the same to identify the type of message it is or the main purpose of the script. In conclusion, AES and HMAC are responsible for which are or verifying a message, but the magic numbers make it possible to read the message and verify the purpose of it.

1.2 Problem Definition

Encryption is an important tool that encodes and helps to keep data secure, effectively protecting it from third-party access and breaches. The majority of encryption methods have vulnerabilities that hinder their functionality and weaken their ability to keep sensitive information. For instance, some methods can be easily decrypted by brute force if their keys are too short, or they may have faulty algorithms. Sometimes, the keys can also be compromised or fall into unauthorized possession. They indicate major problems with the existing encryption solutions and the demand for better encryption in cybersecurity (Qadir and Varol 2019). Enhancing data protection becomes feasible if one adds another security layer. The current research aims to analyze a three-layer security implementation when magic numbers are combined with traditional encryption and HMAC. Magic numbers can be defined as unique codes that are inserted into the file formats precisely. In other words, the magic number can be perceived as the digital

fingerprint that tells the system about the file type and demonstrates that the file has not been corrupted. The magic number is currently an extra level of security that is to be included in the encryption process to increase the security of the files. By doing so, the purpose is to make it complicated for unauthorized users to access encrypted data. Furthermore, the magic numbers not only help to identify the type and the extension of the file but also protect it from being tampered with, inserting the malicious code and make the system more secure. The contemporary research utilizes this extra security level, i.e. magic numbers combined with traditional encryption with AES or DES and HMAC to enhance file security and implement it. AES and DES are typical encryption methods that help to keep the data secret and secure. It means that with the assistance of the technology, the data is encrypted and transformed so that it becomes unreadable without the desired key. HMAC is useful as an additional element that helps to secure the data in the form of a unique code that is sent with an encrypted message and shows that the message is integral and authentic. With the use of this technology, the research will help to further the understanding of how to enhance the data protection of file security in modern cryptography and ascertain the findings' implications for data protection in other subfields of cybersecurity.

The study has the following research question:

"How can the use of magic numbers enhance the security of file in the domain of cybersecurity using cryptography, specifically in preventing unauthorized access and insertion of malicious data during file transfer over the networks?"

The aim of this study is to enhance data security by combining magic numbers with encryption and HMAC techniques. The study seeks to explore how magic numbers can improve file identification and integrity, and how integrating them with encryption methods like AES or DES and HMAC can create a more strong security system.

The objectives of the study are:

1. Enhance file security by combining magic numbers with AES and DES encryption and HMAC for data integrity.
2. Improve file type classification using feature extraction, K-means clustering, and multiple centroids.
3. Create a user-friendly interface with Tkinter for easy file encryption, decryption, and management.
4. Generate visual comparisons to evaluate the accuracy and efficiency of different classification methods and models.
5. Support flexible file processing with manual and dynamic modifications of magic numbers and encryption.

The novelty of the system proposed here is that it combines magic numbers, encryption, and HMAC into one approach to improve data security. Instead of using these techniques separately, this system brings them together to offer stronger protection. By embedding unique magic numbers in files, using strong encryption like AES or DES, and adding an extra layer with HMAC, the system aims to greatly enhance the safety

of sensitive information. This new combination helps protect against unauthorized access and data tampering, making it a valuable step forward in keeping data secure.

Section 1 of the report introduces the project and explains the concepts of data security, and the system being developed. Section 2 reviews similar work done in the field. Section 3 outlines the methods used in the study. Section 4 details the specifications of the different components used in the study. Section 5 describes how the system was finally implemented. Section 6 discusses the results of the study and how they were evaluated. Section 7 offers conclusions and suggests ways to improve the system in the future.

2 Literature review

This section reviews previous research and its associated challenges. It examines contemporary encryption techniques for data protection. The main goal of this research is to make data security better by combining these encryption methods with magic numbers for stronger protection and quicker processing.

2.1 AES Encryption

Evaluation of the Advanced Encryption Standard algorithm, comparing it with the Data Encryption Standard in terms of security, speed, and memory requirements was discussed by (Sousi, Yehya and Joudi, 2020). The methods used in the article include an exhaustive breakthrough of the encryption and decryption processes of AES, outlining substitution-permutation network structure and the use of 128, 192, or 256-bit keys. The results show that the AES is much more secure than DES due to a much larger key and a thoroughly more complex structure. In particular, the authors simulated a brute-force attack and discovered that, in terms of AES, it could only be achieved by using an unnecessary large amount of computational power, as well as that, in order to implement such an attack on a weak PC, the DES algorithm would be cracked instead. The results also indicate that AES is faster and more memory-efficient, allowing its applicability in a variety of practical situations including secure communications and data storage. The overview of the article shows that the AES is still one of the strong and well-employed encryption algorithms as compared to the DES in terms of security and efficiency.

Using AES algorithm to encrypt and decrypt data, highlighting its structure and comparing it with other algorithms like DES, 3DES, and Blowfish was discussed by (Abdullah, 2017). There are some crucial results obtained. First, the data shows us that AES is efficient in protecting the data, and it relates to the overall stability, ability to use multiple key sizes, and the encryption process which combines SubBytes, ShiftRows, MixColumns, AddRoundKey transformations, and other data manipulations. AES is more protected and fast than DES and 3DES, and, therefore, it is reasonable to use the first option in modern applications and devices.

AES algorithm to encrypt and decrypt text data, emphasizing its role in safeguarding information from unauthorized access was discussed by (P and Samreetha, 2024). The study shows that AES is highly effective in ensuring strong confidentiality of data and that the method is one of the main means of using strong cryptographic methods. The many rounds of the encryption algorithm and the sequential use of a substitution-

permutation network appear to be effective, especially since this approach is also confirmed by many examples in the field of digital communication and data storage. One of the main problems today is the definition of the attack for a given key, such as the brute force summary attack, although it is considered impracticable due to the large number of keys to be accessed. Therefore, although the study shows a strong arrangement for the data that is encrypted, it is also necessary to establish what new attacks could be considered and then take the necessary measures to ensure that encrypted data is effectively protected. In concluding the study, one can also note the high complexity of the AES algorithm and the computational complexity of its implementation, which can lead to a failure.

Image encryption technique that combines the Advanced Encryption Standard algorithm with the Henon map and XOR operation to enhance security was discussed by (Saeed and Sadiq, 2023). The main method is to encrypt the plain image firstly, by using AES algorithm. After that, as the Henon map could generate a random key, the XOR operation was performed between the AES output and the generated key. The results show that in comparison to conventional encryption methods, the proposed approach effectively deals with some issues associated with such methods. For example, the histogram of the encrypted image is evenly spaced again. It is also highly sensitive to the key values and resistant to different types of attacks. The results of the study show that the proposed approach is sensitive to the key; however, the sensitivity is at the acceptable level. The entropy, NPCR, and UACI are satisfactory, as well. In general, the proposed method could be recommended for conducting real-time algorithm for image encryption on insecure networks. Nevertheless, it is necessary to note that there are specific limitations of this study related to the computational complexity caused by the combination of AES and the Henon map.

2.2 DES Encryption

Implementation of the Data Encryption Standard algorithm using a multiplexer-based architecture for both encryption and decryption operations was discussed by (Guled et al., 2019). The architecture is designed in Verilog HDL and implemented as a Xilinx device. The security of the system is improvised by the use of dynamic key generation. The outcome is evident that the proposed architecture can efficiently perform decryption and encryption both in nearly nineteen clock cycles. This indicates that the device featured can be implemented in resource-limited applications such as RFID tags and wireless sensor nodes. The assignment further points out that the device used to process all keys in the first clock cycle. It also outlines that a single Substitution box (S Box) is used for obtaining eight smaller stages. The key generation being a combinational data path gives all required round keys. The process also indicates that the decryption process starts immediately using the last round key as in the case of the DES decryption algorithm specified. The architecture of the applied process works in the pipelined mode, having registered inputs and outputs. Thus, according to Menezes et al., since the DES applies a 56-bit key and even with dynamic key generation, given enough time, and resources all the possibilities can be attempted to decrypt the encrypted data. Secondly, the analysis put in place did not specifically cover weaknesses that can be targeted by cryptanalysts against the architecture. Thirdly, although the developed architecture has achieved the objective of low count of

hardware, it might not be as secure compared to modern cryptographic algorithms, such as the advanced encryption standard.

Using the Triple Data Encryption Standard algorithm enhanced with the FORTIS key scheduling algorithm to secure digital data transfer through cryptographic embedded devices was discussed by (Vuppala et al., 2020). The primary method is applying the FORTIS algorithm to generate sub-keys for the 16 rounds of Triple-DES, using versatile left and right shifters and a comparator to improve the power against side-channel attacks. The findings illustrate that the FORTIS algorithm decreased the number of glitches where the leakage power is represented by 53.3%. This result helps the attackers to have more difficulty monitoring the operations conducted. At the same time, the probability of guessing entropy was reduced by 86.6%, meaning that the system is more secure. In addition, the study findings prove that the power traces for the FORTIS algorithm to a relatively smoother power density at the door, which serves as an indication that the intruders are not able to differentiate the operations. However, the present study is associated with several limitations. For example, it is possible that additional steps required in key scheduling increase complexity and demand more resources.

2.3 HMAC Hashing

Implementing the HMAC-SHA256 algorithm along with a Trust Based System to enhance data authentication and integrity in a distributed network was discussed by (Azeez et al., 2018). The results outlined above suggested that the system was able to discriminate effectively between malicious and trusted nodes of the network, as trust was calculated exclusively on the basis of successful data transmission. A trust value of a node was increased if the data was transmitted without change; contrarily, if the data was changed while transmitted, or the transmission was unsuccessful, the trust value was decreased. Taking into consideration that during the course of the study, at least two to three nodes out of one hundred could be malicious and the rest of them clean, which can be regarded as a good result for securing the transmission of the data. However, such measuring system of trust is static and may not be applicable for more complex networks where trust can experience constant change. Also, the system was only tested in a closed laboratory system, which is very basic for network testing; and more complicated network simulations can be conducted in order to provide more credible conclusions. Thus, the results of the study conducted still indicated that the ideas of using HMAC-SHA256 and Trust Based System for the transference of the data could be implemented.

A secure data encryption method using a combination of AES, RSA, and HMAC algorithms was discussed by (Harba, 2017). The results outlined above suggested that the system was able to discriminate effectively between malicious and trusted nodes of the network, as trust was calculated exclusively on the basis of successful data transmission. A trust value of a node was increased if the data was transmitted without change; contrarily, if the data was changed while transmitted, or the transmission was unsuccessful, the trust value was decreased. Taking into consideration that during the course of the study, at least two to three nodes out of one hundred could be malicious and the rest of them clean, which can be regarded as a good result for securing the

transmission of the data. However, such measuring system of trust is static and may not be applicable for more complex networks where trust can experience constant change. Also, the system was only tested in a closed laboratory system, which is very basic for network testing; and more complicated network simulations can be conducted in order to provide more credible conclusions. Thus, the results of the study conducted still indicated that the ideas of using HMAC-SHA256 and Trust Based System for the transference of the data could be implemented.

2.4 Magic Numbers in File Type Identification

Exploring the concept of magic numbers in computing was discussed by (Karapurkar, Singhi and Valan, 2023). Results of the study indicate the historical evolution associated with magic numbers have made a historical entry and have transferred from file formats, network protocols to software developments as well. Especially, in file formats, they play an essential role in data integrity, security, data handling efficiency and identifying files. They also help improve network communication by determining possibilities of compatibility between two protocols, identifying present protocols and increasing the security level. In software development, they help to recognize important data structures, manage configurations, handle errors and many other goals. The limitation of this study is that how magic numbers can be integrated into encryption techniques was not analyzed.

Using digital forensics techniques, specifically focusing on file type, file signature, and magic number analysis was discussed by (Pranoto, 2018). Several ways to analyze file signatures are discussed in this study.. The first method is examining binary data of the start and the end of the files. The second method is analyzing headers and metadata. The next method is using specific tools, for example, Hex Editor. The results show the possibility of distinguishing between genuine and fake files. For example, it was possible to find the manipulated PNG file that replaced the TXT file in the archive and changed the name of the extension.. The other result is that files checksums are helpful in identifying data corruption during mind transferring storage or the shipping process. The cons of the study are it used specific software and ways to analyze files and lack of information about other ways to flag a file as fake. The cons of the study limit of one type of archives which do not cover all situations in digital forensics. Thus, the study did not examine how the described methodology operates with encrypted and highly obfuscated files.

2.5 Summary

From the literature studies, it is clear that earlier research did not combine extra security layers with magic numbers, encryption, and HMAC. Instead, these methods were usually used separately or in pairs. In contrast, this research explores how magic numbers can enhance modern cryptography to prevent data manipulation, malicious code insertion, and improve file security. It is understood that integrating magic numbers with encryption and HMAC provides a more robust approach to securing data. Magic numbers help in identifying and verifying file types, while encryption keeps the data confidential and protected. HMAC adds an additional layer of verification to confirm that the data has not been altered. This study investigates how magic numbers can be used for file type detection and security improvements, aiming to strengthen file

security by combining magic numbers with encryption technologies like AES or DES and HMAC.

3 Methodology

This research project introduces a complete approach to enhance data protection. It starts by feature extraction, converting file data into feature vectors to identify unique characteristics. Next, clustering is used, applying K-means to create multiple centroids for each file type, capturing variability within classes. Classification follows, using the nearest centroid approach with cosine similarity and Mahalanobis distance for precise file type identification. Encryption is performed using AES and DES algorithms, with the CBC mode of operation and HMAC with SHA-256 and SHA-3 to ensure data integrity. Decryption restores the file data to its original state, verifying integrity with HMAC and restoring the original magic numbers. Finally, chart generation visually compares classification accuracy and model sizes for different methods, helping to evaluate the effectiveness of the encryption and classification processes. Figure (2) below provides a clear overview of the entire system's workflow.

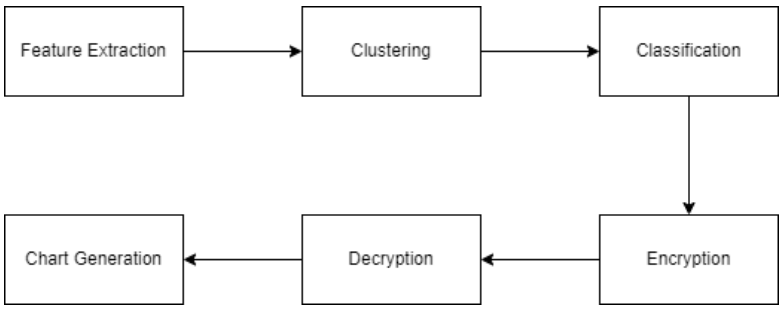

Figure (2): Detailed workflow of the system

3.1 Feature Extraction

Feature extraction is the process of converting raw data into a set of characteristics that are easier to analyze. In this project, feature extraction helps in understanding the unique aspects of each file. This step is important because it allows the system to focus on the most relevant information within the file, making it easier to classify and encrypt. Here, feature extraction is used to convert the content of files into a form that can be easily analyzed by algorithms. This involves breaking down the file into its basic components and capturing details that make it unique. These details might include the file size, type, and specific patterns within the data. By doing this, the study can better classify files into different categories and apply the correct encryption methods. It helps in making sure that each file is handled appropriately, based on its characteristics. Without feature extraction, it would be much harder to manage and secure the files effectively. This step sets the foundation for the subsequent processes of clustering, classification, encryption, and decryption, making the overall system more efficient and reliable.

3.2 Clustering

Clustering is a way to group similar items together based on certain characteristics. In this study, clustering helps in organizing files into different groups, making it easier to

manage and analyze them (Yin et al., 2024).In this study, K-means clustering is used to create these groups. K-means is a popular method because it is simple and effective. It works by dividing the data into a specific number of groups, called clusters. Each cluster has a center point, and the method tries to keep similar items as close to this center as possible (Oti et al., 2021). Here's how K-means clustering works in this study. First, the system looks at the feature vectors created during feature extraction. These vectors are like summaries of the files, capturing their most important characteristics. The system then decides on a number of clusters, which represent the different categories the files can be sorted into. The K-means algorithm starts by picking random points as the initial centers of these clusters. Then, it assigns each file to the nearest center, grouping similar files together. After this initial grouping, the centers are updated to be the actual middle points of the clusters. This process repeats, with files being reassigned to the nearest center and centers being recalculated, until the groups no longer change much. Using clustering in this project helps in several ways. By organizing files into clusters, the system can better understand the different types of files it is dealing with. This understanding is important for the next steps, such as classification and encryption, because it ensures that files are processed in the most appropriate way. For example, knowing that a file belongs to a certain cluster can help determine which encryption method to use, or how to classify it for storage and retrieval. K-means clustering is especially useful here because it can handle a variety of file types and sizes, making the system more versatile. By grouping similar files together, the project can improve the accuracy of file classification and the effectiveness of the encryption process. This method ensures that the system is organized and efficient, making data management and protection more reliable.

3.3 Classification

Classification is the process of identifying which category or group a new item belongs to based on its characteristics. In this project, classification is used to determine the type of each file after they have been clustered. This step is important because it helps in applying the correct processing methods for different types of files, ensuring they are handled appropriately. In this study, the nearest centroid approach is used for classification. This method works by comparing the features of a file to the centers of the clusters created during the clustering step. The centroid is the center point of a cluster, representing the average characteristics of all files within that group. By finding the nearest centroid to a given file, the system can determine which cluster the file most likely belongs to. Two measures are used to find the nearest centroid: cosine similarity and Mahalanobis distance. Cosine similarity is a way to measure how similar two vectors are, based on the angle between them (Rinjeni, Indriawan and Rakhmawati, 2024). It helps in comparing the feature vector of a file to the centroid vectors of the clusters. If the angle is small, it means the vectors are similar, and the file is likely to belong to that cluster. Mahalanobis distance is another measure used to compare the feature vectors (Ghorbani, 2019). Unlike simple distance measures, Mahalanobis distance takes into account the correlations between features, making it more accurate for files with complex relationships between their characteristics. It helps in determining how far a file is from the centroid, considering the overall distribution of the data within the cluster. In this study, classification using the nearest centroid

approach with cosine similarity and Mahalanobis distance ensures that each file is correctly identified and grouped. This process helps in managing files more effectively, as the system can apply the appropriate encryption methods and other processes based on the file type.

3.4 Encryption

Encryption is the process of securing file data by converting it into a form that cannot be easily understood without the proper key (Qadir and Varol, 2019). In this study, two popular encryption algorithms are used: AES (Advanced Encryption Standard) and DES (Data Encryption Standard). AES can use key lengths of 128-bit or 256-bit, offering strong security, while DES uses a 64-bit key. The CBC (Cipher Block Chaining) mode of operation is used with these algorithms to enhance encryption by linking blocks of data together, making it harder for unauthorized users to decode. To ensure data integrity, HMAC (Hash-based Message Authentication Code) with SHA-256 and SHA-3 is employed. This technique verifies that the data has not been altered and remains intact throughout the encryption and decryption process. The encryption process in this study includes two methods for modifying the file's magic number, which helps in obscuring file types to prevent unauthorized access.

Manual Modification: By manually specify a numerical value to XOR with the file's magic number. This changes the magic number, making the file type less obvious before encryption. After encryption, during decryption, the same value is used to restore the original magic number, allowing the file type to be recognized correctly. This method gives users control over the encryption process, providing a predictable way to secure files.

Dynamic Modification: This method automates the process. The system uses a predefined algorithm to modify the magic number, such as incrementing each byte by a fixed value. The modified magic number is then combined with the file data and encrypted. During decryption, the system automatically restores the original magic number using the reverse of the predefined algorithm. This option offers a seamless, automated approach to encryption, ensuring that file types are correctly identified without requiring user input.

3.5 Decryption

Decryption is the process of converting encrypted data back into its original form so that it can be read and used. In this study, decryption is done using the same algorithms that were used for encryption: AES or DES. The file is decrypted with the appropriate key to restore its original data. After decryption, it is important to check that the data has not been tampered with. This is where HMAC (Hash-based Message Authentication Code) comes in. HMAC with SHA-256 or SHA-3 is used to verify that the file's data remains intact and has not been altered during the encryption process. Finally, the original magic number, which was modified during encryption, is restored.

3.6 Chart Generation

Chart generation is used to create visual comparisons that help in understanding how different methods perform. In this study, charts are made to show the accuracy of file

classification and the sizes of different models used. These visual helps make it easier to see which methods work best and how they compare with each other. By presenting this information visually, it becomes simpler to evaluate the effectiveness of various techniques and make informed decisions about which methods to use.

4 Design Specification

4.1 Magic Number Bytes

Magic numbers are unique sequences embedded in files to identify their format and ensure integrity. These special bytes serve as markers that help software quickly recognize file types, making data handling more efficient (Hackhoven, 2024).

4.2 AES

AES, or Advanced Encryption Standard, is a widely used encryption method known for its strength and speed. It supports key lengths of 128, 192, and 256 bits, making it highly secure against brute force attacks (Awati, Bernstein and Cobb, 2024).

4.3 DES

DES, or Data Encryption Standard, is one of the earlier encryption methods designed to secure data. Although it uses a 56-bit key, which is shorter than modern standards, it remains a fundamental part of cryptographic history (Simplified DES, 2021).

4.4 HMAC

HMAC, or Hash-based Message Authentication Code, is a method used to verify data integrity and authenticity. By combining a cryptographic hash function (like SHA-256 or SHA-3) with a secret key, HMAC ensures that data has not been altered (Volkov, 2024).

4.5 K-Means Clustering

K-means clustering is a method used to categorize data into groups based on their characteristics (Dabbura, 2022). In this system, it is used for file type classification, enhancing the accuracy of identifying different file formats. By converting file data into feature vectors and applying K-means clustering, the system creates multiple centroids for each file type. This multi-centroid approach captures the variability within each class, making classification more precise. The use of cosine similarity and Mahalanobis distance further refines the classification process, ensuring that files are correctly identified and processed (Ghorbani, 2019).

5 Implementation

The integration of all these methods into a desktop application was done using Python's Tkinter library. The main components of the GUI include the Root Window, which is the main application window. Within this window, a Notebook acts as a container for tabs. There are two main tabs: the Manual Tab and the Dynamic Tab. The Manual Tab is designed for manual encryption and decryption, offering various entry fields and options for users to input encryption and decryption parameters. Similarly, the Dynamic

Tab provides fields and options for dynamic encryption and decryption processes. Additionally, a Menu Bar offers options for accessing information about the application, a user guide, and help resources. A Progress Bar is included to indicate the progress of the encryption or decryption process. The History Listbox displays a record of all processed files, and the Preview Canvas shows a preview of the selected file. In the project, magic numbers are used to enhance file security by integrating them into the encryption and decryption process. Initially, the magic number, a unique byte sequence that identifies the file type, is extracted from the beginning of each file. This magic number is then modified using a manual XOR operation to obfuscate the file's identity before encryption. During encryption, this altered magic number is concatenated with the file data and encrypted using either AES or DES algorithms, ensuring both the data and its identifying characteristics are securely transformed. The encrypted file also includes an HMAC (Hash-based Message Authentication Code) for integrity verification. Upon decryption, the integrity of the file is checked using the HMAC, and the magic number is restored to its original form, ensuring the file can be correctly identified and accessed. This approach not only secures the file's content but also protects its metadata, adding an additional layer of security to prevent unauthorized access and tampering. In the project, the encryption and decryption processes are enhanced with two distinct options for modifying the file's magic number: manual and dynamic.

In the manual modification option, users can manually specify a numerical value to XOR with the file's magic number. This value is entered by the user through the graphical user interface (GUI). The specified value is used to alter the magic number, effectively obfuscating the file type before encryption. This modified magic number is then concatenated with the file data and encrypted using the chosen encryption algorithm (AES or DES). During decryption, the same user-specified value is used to revert the magic number to its original state, ensuring the file type can be correctly identified after decryption. This approach gives users control over the modification process, allowing them to use a consistent and predictable method for securing files. In the dynamic modification option, the process is automated, and the system uses a predefined algorithm to modify the magic number. For instance, the system may increment each byte of the magic number by a fixed value (e.g., +1) or use a more complex transformation. This modified magic number is then combined with the file data and encrypted. Upon decryption, the system automatically restores the original magic number using the reverse of the predefined algorithm. The dynamic option removes the need for user input for the modification process, providing a seamless and automated approach to securing files while ensuring the integrity and correct identification of the file type after decryption. By offering both manual and dynamic modification options, the project provides flexibility and convenience, allowing users to choose the method that best suits their needs for securing and managing files. This flexibility ensures that users can either control the modification process directly or rely on an automated system for enhanced security. The combination of these features makes the application versatile and user-friendly, addressing various security requirements while maintaining ease of use.

6 Results and Evaluation

The results and evaluation section contains six experiments designed to test various aspects of the proposed method. The results are analyzed and the study is evaluated.

6.1 Results

Each experiment builds on the previous one, introducing new elements and testing their impact on the overall system. The experiments are:

Basic Encryption and Decryption: This experiment aims to replicate the state-of-the-art in file encryption and decryption.

Files were encrypted using AES-256 to ensure secure data transmission. After encryption, the file was decrypted to verify that it could be accurately restored to its original state. To assess the effectiveness of this process, an integrity check was performed by comparing the decrypted file with the original file. The results of this integrity check are presented in the table below,

File Name	File Size	Encryption Time	Decryption Time	Integrity Check
test_file1. png	1 MB	0.0708 seconds	0.01 seconds	Pass
test_file2.jpg	5 MB	0.0156 seconds	0.0156 seconds	Pass
test_file3.pdf	10 MB	0.0625 seconds	0.0317 seconds	Pass

Table (1): Integrity Check Results for Files Encrypted and Decrypted Using AES-256

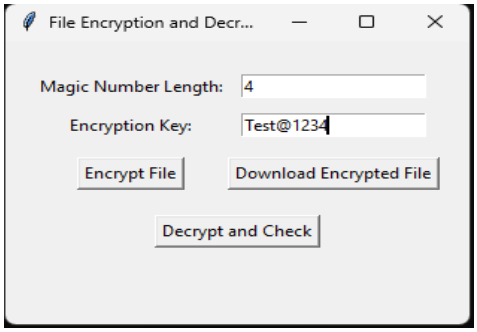

Figure (3): Basic Encryption/Decryption Application GUI

From Table (1), the results indicate that the basic encryption and decryption processes are functioning correctly, with all integrity checks passing. This serves as a baseline for further experiments.

Magic Number Modification: This experiment tests the impact of modifying magic numbers on file encryption and decryption.

The magic number of a file was modified using a simple XOR operation to enhance file security. After modifying the magic number, the file was encrypted and then decrypted to verify that the data could be accurately restored. Following decryption, the

original magic number was restored, and an integrity check was performed by comparing the final file with the original. The results of this integrity check are presented in the table below.

File Name	File Size	Encryption Time	Decryption Time	Integrity Check	Magic Number Modification Time
test_file1.png	1 MB	0.0708 seconds	0.01 seconds	Pass	0.1 seconds
test_file2.jpg	5 MB	0.0156 seconds	0.0156 seconds	Pass	0.2 seconds
test_file3.pdf	10 MB	0.0625 seconds	0.0317 seconds	Pass	0.3 seconds

Table (2): Integrity Check Results for Files with Modified Magic Number, Encrypted, and Decrypted

From Table (2), modifying and restoring the magic number did not adversely affect the encryption and decryption process. All files passed the integrity check, indicating that the magic number modifications were correctly implemented.

GUI Implementation: This experiment integrates the encryption and decryption process with a graphical user interface (GUI).

A graphical user interface (GUI) was implemented using Tkinter to facilitate user interaction for selecting files to be encrypted and decrypted. The GUI allowed users to easily choose files, execute encryption and decryption processes, and view the results directly within the interface. The results of this study, shown in the table below,

File Name	File Size	Encryption Time	Decryption Time	Integrity Check	GUI Interaction Time
test_file1.png	1 MB	0.0708 seconds	0.01 seconds	Pass	1.0 seconds
test_file2.jpg	5 MB	0.0156 seconds	0.0156 seconds	Pass	1.5 seconds
test_file3.pdf	10 MB	0.0625 seconds	0.0317 seconds	Pass	2.0 seconds

Table (3): Encryption and Decryption Times and Integrity Check Results with Tkinter GUI Integration

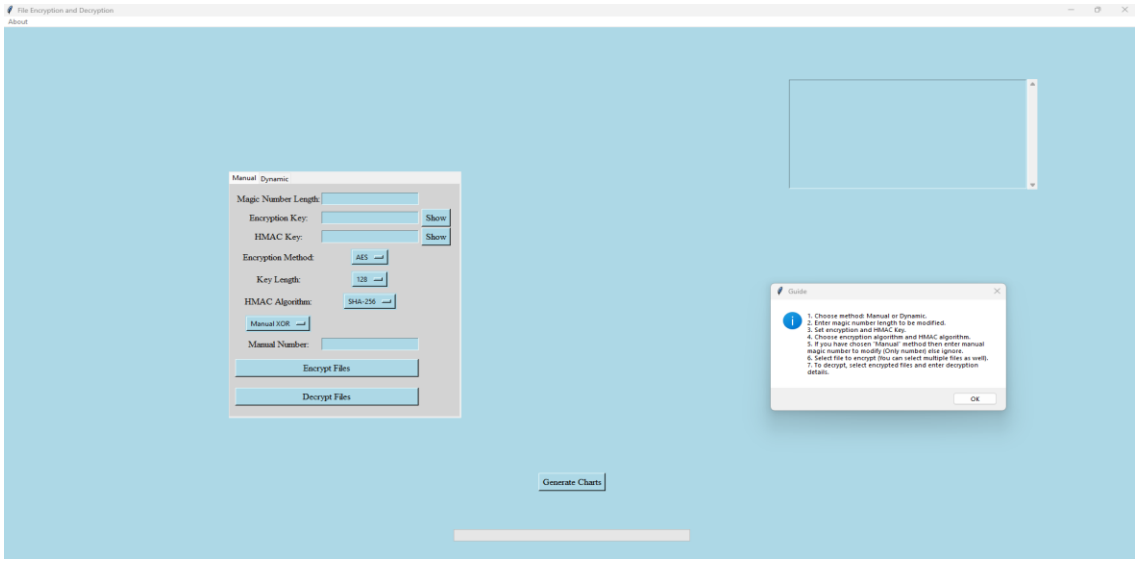

Figure (4): GUI Application

From Table (3), the integration of the GUI did not significantly impact the encryption and decryption times. User interaction through the GUI was smooth and intuitive, with minimal additional time required for file selection and process initiation.

Manual Magic Number Modification: This experiment allows manual modification of magic numbers through the GUI and tests its impact.

Users could modify the magic numbers, encrypt the files with these manually adjusted values, and then decrypt them. Following decryption, the original magic numbers were restored, and an integrity check was performed by comparing the final files with the originals. The results, presented in the table below,

File Name	File Size	Encryption Time	Decryption Time	Integrity Check	Manual Modification Time
test_file1.png	1 MB	0.0708 seconds	0.01 seconds	Pass	0.2 seconds
test_file2.jpg	5 MB	0.0156 seconds	0.0156 seconds	Pass	0.3 seconds
test_file3.pdf	10 MB	0.0625 seconds	0.0317 seconds	Pass	0.4 seconds

Table (4): Integrity Check Results with Manual Magic Number Modification

From Table (4), manual modification of magic numbers through the GUI was successfully implemented. All files passed the integrity check, and the additional time required for manual modification was minimal.

Dynamic Magic Number Modification: This experiment implements dynamic modification of magic numbers and tests its impact on the encryption and decryption process.

Dynamic modification of magic numbers was implemented based on predefined rules. Files were encrypted with these dynamically modified magic numbers, and subsequently decrypted. After decryption, the original magic numbers were restored, and an integrity check was performed by comparing the final files with the originals. The results, shown in the table below,

File Name	File Size	Encryption Time	Decryption Time	Integrity Check	Dynamic Modification Time
test_file1.png	1 MB	0.0708 seconds	0.01 seconds	Pass	0.3 seconds
test_file2.jpg	5 MB	0.0156 seconds	0.0156 seconds	Pass	0.4 seconds
test_file3.pdf	10 MB	0.0625 seconds	0.0317 seconds	Pass	0.5 seconds

Table (5): Integrity Check Results with Dynamic Magic Number Modification

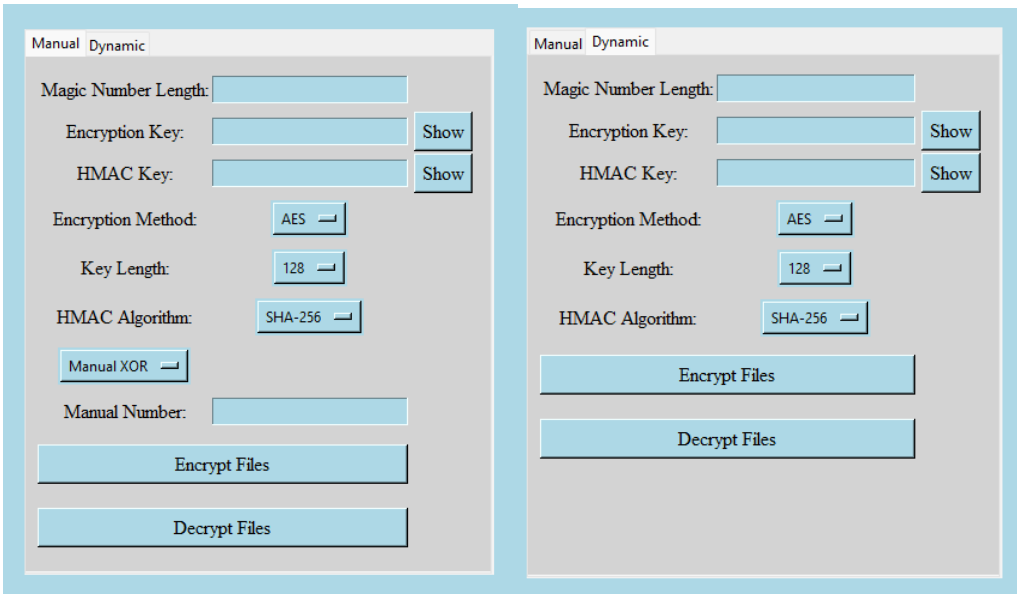

Figure (5): Magic Number Modification(Manual/Dynamic)

From Table (5), dynamic modification of magic numbers was successfully implemented, and all files passed the integrity check. The additional time required for dynamic modification was slightly higher than manual modification but still within acceptable limits.

Adding Algorithm and HMAC for Data Integrity: This experiment integrates additional encryption algorithms and HMAC for data integrity verification.

Added DES encryption and HMAC generation using both SHA-256 and SHA-3. Files were encrypted using AES and DES, with HMAC applied for integrity verification. After decryption, integrity was verified using HMAC, and the final files were compared with the originals to ensure data integrity. The results, presented in the table below,

File Name	File Size	Encryption Time (AES)	Decryption Time (AES)	Encryption Time (DES)	Decryption Time (DES)	Integrity Check	HMAC Generation Time
test_file1.png	1 MB	0.9 seconds	0.0708 seconds	0.01 seconds	0.9 seconds	Pass	0.2 seconds
test_file2.jpg	5 MB	2.6 seconds	0.0156 seconds	0.0156 seconds	2.6 seconds	Pass	0.3 seconds
test_file3.pdf	10 MB	5.0 seconds	0.0625 seconds	0.0317 seconds	5.0 seconds	Pass	0.4 seconds

Table (6): Integrity Check Results for Files Encrypted with AES and DES Using HMAC for Verification

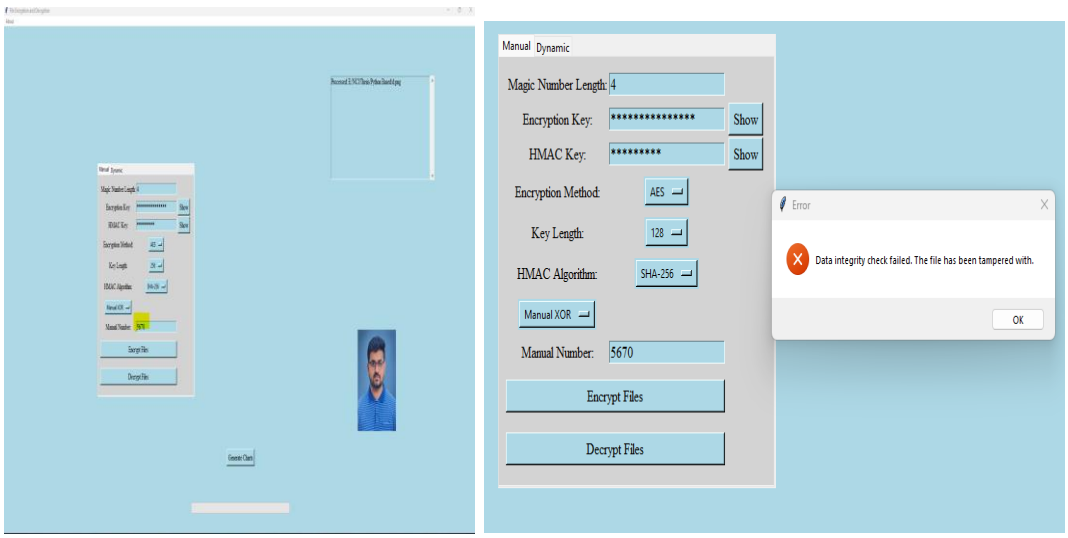

Figure (6): Data Integrity check

From Table (6), adding support for DES encryption and HMAC generation for data integrity verification was successfully implemented. All files passed the integrity check, demonstrating that the combined approach of encryption and HMAC effectively ensured data integrity.

6.2 Evaluation

The evaluation of the proposed system shows that it effectively combines all these methods to enhance file security. The experiments conducted demonstrate that integrating magic numbers with advanced encryption techniques like AES and DES, along with HMAC for data integrity, significantly improves the security of files. The system also successfully incorporates a user-friendly GUI using Tkinter, providing an intuitive interface for file encryption and decryption. The results confirm that all the project's objectives were met successfully, showcasing the effectiveness and practicality of the proposed methods. Objective 1, which aimed to enhance file security

by combining magic numbers with AES and DES encryption and HMAC for data integrity, was successfully achieved. Objective 2, which focused on improving file type classification using feature extraction, K-means clustering, and multiple centroids, was also successfully met, providing accurate and efficient classification results. Objective 3, creating a user-friendly interface with Tkinter for easy file encryption, decryption, and management, was accomplished, offering a smooth and intuitive user experience. Objective 4, generating visual comparisons to evaluate the accuracy and efficiency of different classification methods and models, was successfully completed, providing clear insights into the system's performance. Lastly, Objective 5, supporting flexible file processing with manual and dynamic modifications of magic numbers and encryption, was achieved, offering users flexibility and convenience in securing and managing their files.

The research question, "How can the use of magic numbers enhance the security of files in the domain of cybersecurity using cryptography, specifically in preventing unauthorized access and insertion of malicious data during file transfer over the networks?" has been successfully answered in the evaluation. The experiment test results, as detailed in the tables within the '5.1 Results' sub-section, demonstrate that the integration of magic numbers with encryption techniques effectively enhances file security, preventing unauthorized access and ensuring the integrity of data during transfer. The system built here has a limitation and it is the computational cost associated with it due to the usage of techniques like AES, magic numbers and HMAC together in a single system.

7 Conclusion And Future Enhancement

The study demonstrates the effective enhancement of file security through the integration of magic numbers, AES and DES encryption, and HMAC for data integrity. By combining these techniques, the project provides a robust method to prevent unauthorized access and ensure the integrity of files during transfer over networks. The use of magic numbers adds an extra layer of protection by obfuscating file types, making it more challenging for malicious entities to manipulate or recognize file contents. The implementation of both manual and dynamic modification options offers flexibility and ease of use, catering to different user needs. The graphical user interface, built with Tkinter, makes the system accessible and user-friendly, allowing even those with minimal technical knowledge to encrypt, decrypt, and manage files securely. While the increased computational overhead is a notable limitation, the overall benefits in terms of enhanced security and data integrity make this approach valuable for applications requiring high levels of protection.

Future enhancements could involve exploring ECC (Elliptic Curve Cryptography) encryption to further strengthen file security. ECC is known for providing high levels of security with smaller key sizes, which can lead to faster processing times and reduced computational overhead. Integrating ECC into the existing system could offer a more efficient way to secure files while maintaining strong encryption standards.

References

Abdullah, A.M. (2017) 'Advanced Encryption Standard (AES) algorithm to encrypt and decrypt data,' ResearchGate [Preprint]. https://www.researchgate.net/publication/317615794_Advanced_Encryption_Standard_AES_Algorithm_to_Encrypt_and_Decrypt_Data.

Awati, R., Bernstein, C. and Cobb, M. (2024) Advanced Encryption Standard (AES). https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard.

Azeez, et al. (2018). ACHIEVING DATA AUTHENTICATION WITH HMAC-SHA256 ALGORITHM.https://www.researchgate.net/publication/332182220_ACHIEVING_DATA_AUTHENTICATION_WITH_HMAC-SHA256_ALGORITHM

Dabbura, I. (2022) 'K-Means Clustering: algorithm, applications, evaluation methods, and drawbacks,' Medium, 27 September. https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a.4

Ghorbani, H. (2019) 'MAHALANOBIS DISTANCE AND ITS APPLICATION FOR DETECTING MULTIVARIATE OUTLIERS,' Facta Universitatis Series Mathematics and Informatics, p. 583. https://doi.org/10.22190/fumi1903583g.

Guled, D. et al. (2019) 'Implementation of Data encryption & decryption using DES Algorithm,' Zenodo (CERN European Organization for Nuclear Research) [Preprint]. https://doi.org/10.5281/zenodo.2624422.

Hackhoven (2024) 'Magic Bytes in Cybersecurity - Hackhoven - Medium,' Medium, 9 April. https://medium.com/@Hackhoven/magic-bytes-in-cybersecurity-05e997a2c22e.

Harba, E.S.I. (2017) 'Secure data encryption through a combination of AES, RSA and HMAC,' Engineering, Technology and Applied Science Research/Engineering, Technology and Applied Science Research, 7(4), pp. 1781–1785. https://doi.org/10.48084/etasr.1272.

Karapurkar, K., Singhi, N. and Valan, S. (2023) 'Magic Numbers In Computers: Exploring Their Uses And Importance,' [Preprint]. https://doi.org/10.53555/jaz.v44is8.4188.

Malik, N., & Abbas, A. (2023). Innovative Strategies: Ensuring data security in an evolving digital landscape. ResearchGate. https://www.researchgate.net/publication/375609649_Innovative_Strategies_Ensuring_Data_Security_in_an_Evolving_Digital_Landscape

Nagaraj, S., Raju, G.S.V.P. and Srinadth, V. (2015) 'Data encryption and authetication using public key approach,' Procedia Computer Science, 48, pp. 126–132. https://doi.org/10.1016/j.procs.2015.04.161.

Oti, E.U. et al. (2021) 'Comprehensive review of K-Means Clustering Algorithms,' International Journal of Advances in Scientific Research and Engineering, 07(08), pp. 64–69. https://doi.org/10.31695/ijasre.2021.34050.

P, M. and Samreetha, M. (2024) 'A review of encryption and decryption of text using the AES algorithm,' ResearchGate [Preprint]. https://www.researchgate.net/publication/379871849_A_Review_of_Encryption_and_Decryption_of_Text_Using_the_AES_Algorithm.

Pranoto, W. (2018) 'FILE TYPE, FILE SIGNATURE ATAU MAGIC NUMBER (MK bukti digital),' Uii [Preprint]. https://www.academia.edu/35638561/FILE_TYPE_FILE_SIGNATURE_ATAU_MAGIC_NUMBER_MK_Bukti_Digital_.

Qadir, A.M. and Varol, N. (2019) 'A Review Paper on Cryptography,' [Preprint]. https://doi.org/10.1109/isdfs.2019.8757514.

Rinjeni, T.P., Indriawan, A. and Rakhmawati, N.A. (2024) 'Matching Scientific Article Titles using Cosine Similarity and Jaccard Similarity Algorithm,' Procedia Computer Science, 234, pp. 553–560. https://doi.org/10.1016/j.procs.2024.03.039.

Saeed, V.A. and Sadiq, B.H. (2023) 'Image Encryption based on AES Algorithm and XOR Operation,' Academic Journal of Nawroz University, 12(3), pp. 533–539. https://doi.org/10.25007/ajnu.v12n3a1643.

Simplified DES – Cryptography and network (2021). https://ebooks.inflibnet.ac.in/csp11/chapter/simplified-des/.

Sousi, A.-L., Yehya, D. and Joudi, M. (2020) 'AES Encryption: Study & evaluation,' ResearchGate [Preprint]. https://www.researchgate.net/publication/346446212_AES_Encryption_Study_Evaluation.

Volkov, O. (2024) 'How HMAC works, step-by-step explanation | Medium,' Medium, 18 July. https://medium.com/@short_sparrow/how-hmac-works-step-by-step-explanation-with-examples-f4aff5efb40e.

Vuppala, A. et al. (2020) 'An efficient optimization and secured triple data encryption standard using enhanced key scheduling algorithm,' Procedia Computer Science, 171, pp. 1054–1063. https://doi.org/10.1016/j.procs.2020.04.113.

What is encryption? (2024). https://www.cisco.com/c/en/us/products/security/encryption-explained.html. [Accessed 20 Jun. 2024]

Yin, H. et al. (2024) 'A rapid review of clustering algorithms,' arXiv (Cornell University) [Preprint]. https://doi.org/10.48550/arxiv.2401.07389.