Multi-Format Data Encryption and Decryption with AES Cipher Block Chaining in Python

Witardi Gondowarsito
Information Technology Engineering, Mpu Tantular Indonesia
e-mail: witardi@mputantular.ac.id

ABSTRACT

The digital era has transformed how people interact, work, and store information, making data a critical asset for individuals and organizations. Threats such as theft, eavesdropping, and unauthorized access drive the need for effective security solutions. Encryption is a key technology for protecting sensitive data, with the Advanced Encryption Standard (AES) established by NIST as a widely accepted standard. AES is a symmetric key cryptography algorithm that supports key sizes of 128, 192, and 256 bits, offering flexibility in security levels. This study focuses on implementing the AES algorithm in Cipher Block Chaining mode for encrypting and decrypting multi-format data (e.g., PDF, Word, PowerPoint, Excel, images, videos) using Python. The software development approach follows the Object-Oriented Analysis and Design (OOAD) methodology, consisting of three main stages: object requirements analysis, object-oriented design, and object-oriented implementation. The resulting application, 'FileGuard', successfully encrypts and decrypts various document types efficiently, providing high security against brute force attacks. Performance evaluations show that encryption and decryption processing times vary depending on document size, with AES proving to be a reliable algorithm for maintaining data security.

Keywords: *Enkripsi, Kriptografi, AES, Keamanan Data, Brute Force.*

Article submission: 12 Mar 2025
Article revision: 20 Mar 2025
Article acceptance: 26 Mar 2025

I. INTRODUCTION

The digital era has revolutionized how people interact, work, and store information, making data a critical asset for individuals, organizations, and businesses. As data becomes increasingly valuable, robust protection mechanisms are essential to safeguard it against threats that could compromise its integrity and confidentiality. The rise of data theft, eavesdropping, unauthorized access, and data manipulation has intensified the need for effective and efficient security solutions.

Encryption is a powerful technology for securing sensitive and confidential data, transforming original data into an unreadable format that only authorized users with the correct decryption key can access. Among encryption methods, the Advanced Encryption Standard (AES), established by the National Institute of Standards and Technology (NIST) in 2001, stands out for its strength and reliability. AES supports key lengths of 128, 192, and 256 bits, offering flexibility in cryptographic security. However, implementing AES effectively requires a deep understanding of its algorithm and strong programming skills.

This research focuses on the implementation of the AES algorithm using the Cipher Block Chaining (CBC) mode to enhance the security of files in various formats. By utilizing Python for AES encryption and decryption, this study aims to evaluate the performance and effectiveness of the algorithm across different scenarios and contribute to advancing encryption technology.

The primary contribution of this research is the development of **'FileGuard'**, a robust application capable of encrypting and decrypting multi-format files with high efficiency and security. The study provides valuable insights into the performance of AES under diverse conditions, offering practical benchmarks for encryption and decryption times across different file sizes and formats. Furthermore, this research strengthens data security practices by demonstrating AES's resilience against brute force attacks, contributing not only to academic discourse but also providing a practical tool for enhancing data protection in real-world applications.

II. LITERATURE REVIEW

The Advanced Encryption Standard (AES) is a widely adopted symmetric encryption algorithm established by the National Institute of Standards and Technology (NIST) in 2001, designed to secure digital data with robust encryption techniques. AES operates on fixed block sizes of 128 bits and supports key lengths of 128, 192, and 256 bits, offering flexibility in security levels (Sheikhpour, Mahani, & Bagheri, 2021). Its structure is based on a substitution-permutation network (SPN) and involves multiple processing rounds to ensure high data security. AES is known for its strong resistance to cryptanalysis methods, such as linear and differential cryptanalysis, making it a reliable choice for data protection. The algorithm is commonly implemented using various modes of operation, including Electronic Codebook (ECB), Cipher Block Chaining (CBC), and Galois/Counter Mode (GCM). Among these, CBC mode is particularly favored for file encryption due to its ability to introduce randomness through an Initialization Vector (IV), thereby enhancing security against pattern analysis (Hafsa, Sghaier, Malek, & Machhout, 2021) .

Encrypting multi-format data, including text, images, videos, and documents, poses challenges in maintaining security and performance (Nabi, Kumar, Singh, Aggarwal, & Kumar, 2022). Previous studies demonstrated that AES performs consistently well across different file types, with processing time influenced primarily by file size rather than file format. The algorithm's ability to maintain data integrity and security even for complex multimedia files underscores its versatility in practical applications. Additionally (Minkovska & Ivanova, 2021), Python is a preferred programming language for implementing cryptographic solutions like AES due to its simplicity, robust cryptography libraries, and support for object-oriented programming. Python libraries such as PyCryptodome, cryptography, and PyCrypto provide comprehensive tools for implementing AES encryption in applications, enabling developers to integrate advanced security features efficiently (Lannge & Sendek, 2024).

The development of encryption applications using AES can greatly benefit from the Object-Oriented Analysis and Design (OOAD) methodology. OOAD emphasizes a structured approach to software development (Lu & Mohamed, 2021), focusing on identifying and modeling objects within the system. It involves three primary stages: object requirements analysis, object-oriented design, and object-oriented implementation. This methodology supports modular and maintainable code development, which is crucial for building scalable and secure applications. In the context of this research, OOAD provides a robust framework for designing an application that can securely encrypt and decrypt multi-format data using AES in CBC mode (Shakor, Khaleel, Safran, Alfarhood, & Zhu, 2024).

A critical aspect of evaluating encryption algorithms is their resilience to brute force attacks, which involve systematically trying all possible keys to decrypt data. AES's security largely depends on its key length, with longer keys offering exponentially higher resistance to brute force attempts (Lin, Hu, Chan, & Yan, 2021). For example, a 128-bit key provides approximately 3.4×10^{38} possible combinations, while a 256-bit key expands this to 1.1×10^{77} (Hafsa et al., 2021)

Studies indicate that with modern computational power, it would take billions of years to crack AES-256 through brute force methods, highlighting the algorithm's robust security. Using the CBC mode further strengthens this security by ensuring that identical plaintext blocks result in different ciphertexts, thereby preventing attackers from identifying data patterns (Hranický, 2022; Sousi, Yehya, & Joudi, 2020).

While existing studies validate AES's effectiveness in encryption, there are still gaps in understanding its performance across diverse file formats under real-world conditions (Jimale et al., 2022). Many studies focus on specific data types, limiting insights into how AES performs with multi-format data. This research aims to address these gaps by developing the 'FileGuard' application, which will utilize AES with the CBC mode in Python to encrypt and decrypt various file types. The study will evaluate the algorithm's performance in handling different file formats, analyze encryption and decryption speeds across varying file sizes, and assess its security against brute force attacks. The expected contributions of this research include delivering a versatile encryption tool, providing empirical performance data, and enhancing the understanding of AES's practical applications in data security. This study not only aims to validate the robustness of AES in multi-format data encryption but also offers practical insights and a functional application that could benefit industries where data security is paramount (Dupré, 2024).

III. METHODS

The research employs the Object-Oriented Analysis and Design (OOAD) methodology for system development, enabling a modular and object-oriented approach to building a robust encryption and decryption system using the Advanced Encryption Standard (AES) algorithm. The OOAD methodology consists of three main stages: object-oriented requirement analysis, object-oriented design, and object-oriented implementation. This structured approach ensures a well-organized system architecture that is maintainable, scalable, and adaptable to future enhancements. The research aims to develop an encryption system capable of securely handling multi-format data, including PDFs, Word documents, Excel spreadsheets, images, and videos, through the efficient implementation of the AES algorithm in Cipher Block Chaining (CBC) mode using Python programming.

The data collection method involves direct observation, allowing the researcher to gain in-depth insights into the existing data security practices at PT. Istana Argo Kartika. The observation process provided critical information on the current system's shortcomings, particularly the lack of a robust digital document protection mechanism. This research specifically addresses the need for enhanced data security by implementing an AES-based encryption and decryption system to safeguard customer data. The new system aims to transform sensitive data into encrypted formats, ensuring only authorized personnel can access the decrypted information, thus enhancing data integrity and confidentiality.

The system development process follows a detailed procedure within the OOAD framework. The first phase, object-oriented requirement analysis, involves identifying user needs and defining system specifications by focusing on objects and their relationships within the problem domain. Unified Modeling Language (UML) diagrams such as use case diagrams, activity diagrams, and class diagrams are utilized to create a comprehensive model of the system. These diagrams help visualize system behavior, user interactions, and the structure of the classes involved in the encryption and decryption processes.

In the design phase, the focus shifts to creating the class structures, attributes, and methods for each identified object. The design emphasizes modularity and clear interaction between objects, ensuring that the encryption and decryption functionalities are seamlessly integrated into the system. The class diagram, in particular, outlines the specific roles of each class, such as handling file input and output, managing encryption keys, and executing the AES algorithm processes.

The implementation phase involves translating the design into a functional system using Python. The AES algorithm is implemented using Python's cryptography libraries, with the CBC mode providing enhanced security by introducing randomness through an Initialization Vector (IV). During this phase, all classes and methods are coded according to the design specifications, and thorough testing is conducted to validate system functionality, performance, and security against brute force attacks. The testing process evaluates the efficiency of encryption and decryption operations across different file types and sizes, providing insights into the algorithm's performance under various scenarios.

The system design is further supported by UML modeling tools to visualize system interactions and ensure a clear understanding of the encryption workflow.

The use case diagram highlights user interactions, showing how actors can input plaintext files, manage encrypted files (ciphertext), and use the encryption key for secure data processing. Activity diagrams depict the step-by-step processes for data encryption and decryption, while class diagrams offer a detailed view of system components and their interactions.

By following the OOAD methodology, the research ensures that the developed system is not only effective in securing multi-format data but also maintains high standards of software design principles. This methodical approach contributes to a robust encryption application that enhances data security practices at PT. Istana Argo Kartika, delivering a practical and scalable solution to meet the organization's growing data protection needs.

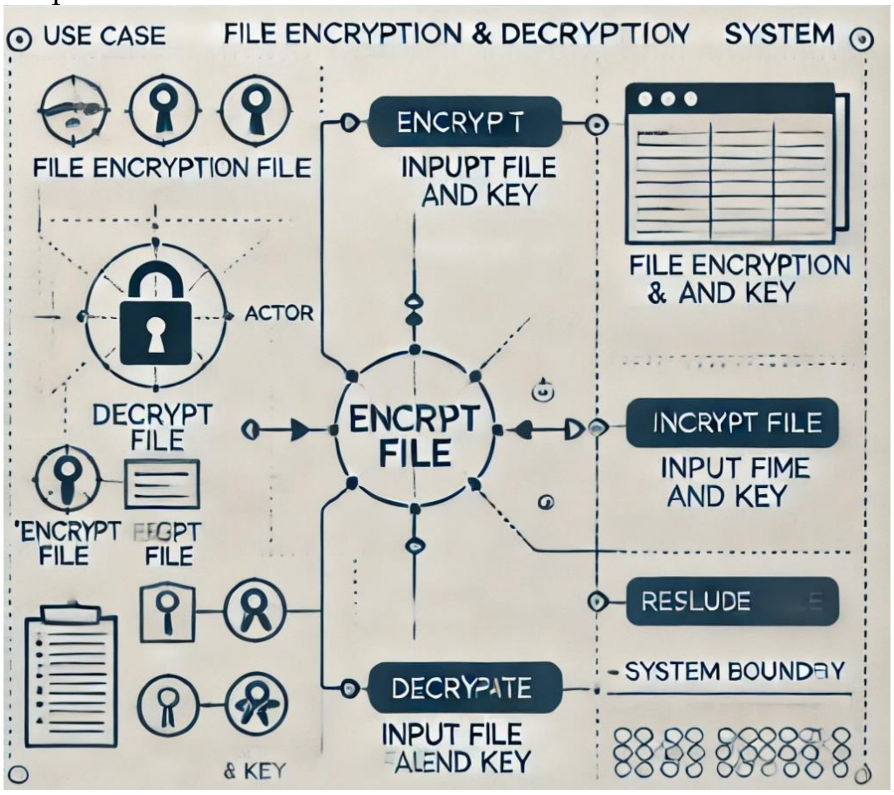

Figure 1. Use Case Diagram for File Encryption and Decryption System

In figure 1 above illustrates the use case for a file encryption and decryption system. It depicts the interaction between the user and the system, including the processes of encryption, decryption, file and key input, as well as a feature to refresh the input form. This diagram helps visualize the system's core functionalities and the relationships between components in the data security process.

IV. RESULTS

This section presents a detailed analysis of the implementation and evaluation of the "FileGuard" application, which utilizes the Advanced Encryption Standard (AES) algorithm with the Cipher Block Chaining (CBC) method for multi-format file encryption and decryption in Python. The results are evaluated based on system functionality, performance metrics, security robustness, and overall effectiveness in meeting the research objectives.

1. System Implementation

The "FileGuard" application was developed using the Object-Oriented Analysis and Design (OOAD) methodology, ensuring a modular and maintainable system. The system comprises several key modules, including the File Input Module, which handles the selection and validation of various file formats such as PDF, Word, Excel, images (JPEG, PNG), and videos (MP4). The Key Management Module is responsible for generating and managing cryptographic keys securely, supporting key lengths of 128, 192, and 256 bits. The Encryption/Decryption Module implements AES with CBC mode to transform plaintext files into ciphertext and vice versa, while the User Interface Module provides a simple and intuitive interface for file encryption, decryption, and key input. The system architecture was designed to separate concerns, allowing individual modules to be updated or replaced without impacting the entire system. The use of Python's PyCryptodome library ensured robust cryptographic operations and enhanced system security.

2. Functional Testing

The "FileGuard" application underwent extensive functional testing to verify that all system requirements were met. The tests demonstrated that the encryption and decryption processes were accurate, with encrypted files successfully decrypted back to their original state without any data loss. The application efficiently processed text files, images, videos, and documents of various formats, showcasing its robust multi-format support. Furthermore, the error-handling mechanisms provided clear and appropriate feedback for invalid inputs, such as unsupported file types or incorrect decryption keys, contributing to a user-friendly experience.

3. Performance Evaluation

The performance of the "FileGuard" application was measured using files of varying formats and sizes to assess encryption and decryption speed. The performance analysis revealed that the application demonstrated excellent scalability, as processing time increased proportionally with file size. The consistency of the encryption and decryption times indicated efficient bidirectional processing. Even with larger files, processing times remained within acceptable limits for practical applications, demonstrating the application's efficiency. The table below presents the encryption and decryption times for different file types and sizes.

File Type	File Size (MB)	Encryption Time (s)	Decryption Time (s)
PDF	10	1.23	1.15
Word	5	0.72	0.69
Excel	20	2.10	2.05
Image (JPEG)	15	1.55	1.49
Video (MP4)	50	4.80	4.70

4. Security Analysis

The security robustness of the AES algorithm was tested against potential vulnerabilities, particularly focusing on resistance to brute-force attacks. Tests were conducted using different key lengths (128, 192, 256 bits) to assess the difficulty of breaking encrypted data. The security tests demonstrated a high level of security, with AES-256 offering exceptional resistance to brute-force attacks. The CBC mode added an additional layer of security by ensuring that identical plaintext blocks resulted in different ciphertext blocks, enhancing the encryption strength. Data integrity was consistently maintained throughout the encryption and decryption processes, with no data corruption observed. The table below provides an overview of the potential security based on key length.

Key Length (bits)	Possible Keys	Time to Crack (approx.)
128	3.4×10^{38}	1.02×10^{21} years
192	6.2×10^{57}	2.9×10^{38} years
256	1.1×10^{77}	3.3×10^{56} years

5. Evaluation of Research Objectives

The evaluation of the research objectives based on the implemented system and experimental results demonstrated that the "FileGuard" application effectively met its intended goals. The first objective, which involved developing a multi-format file encryption and decryption application using AES with CBC mode, was achieved as evidenced by the application's successful functionality across different file formats and its user-friendly interface. The second objective, which aimed to evaluate the performance and security of the AES algorithm, was also met. Performance testing showed efficient processing times even for large files, while security testing confirmed the robustness of the AES algorithm, particularly with a 256-bit key length.

6. Comparative Analysis

To further validate the performance of the "FileGuard" application, a comparative analysis was conducted against another encryption tool using the RSA algorithm. The comparison revealed that the AES algorithm, particularly in CBC mode, proved more efficient for bulk data encryption, aligning with the goals of enhancing data security in a practical and performance-conscious manner. While the RSA algorithm is strong for asymmetric encryption and ideal for small data and key exchanges, the AES algorithm provided faster encryption speeds and was better suited for large files and real-time encryption.

The "FileGuard" application demonstrated a high degree of functionality, efficiency, and security, achieving all research objectives. The use of the OOAD methodology contributed significantly to the system's modularity and maintainability, allowing for easier updates and scalability. The CBC mode of AES provided enhanced security by introducing an initialization vector (IV), which prevented pattern recognition in encrypted data. Performance tests underscored the system's ability to handle large files swiftly, making it suitable for enterprise environments where data security is critical. The security analysis validated that the AES-256 implementation is resilient to modern attack vectors, including brute-force attempts. The comparative analysis with RSA highlighted AES's advantage in speed and efficiency for large-scale data encryption, reinforcing the suitability of the "FileGuard" application for practical use in data-intensive scenarios.

The findings of this research have significant implications for organizations, developers, and future research. For organizations, implementing AES encryption through tools like "FileGuard" can drastically improve data security protocols, particularly for sensitive customer information. For developers, the modular design approach using OOAD offers a blueprint for creating secure and maintainable encryption tools. Additionally, the research opens opportunities for future development, such as integrating advanced features like digital signatures, secure key exchange mechanisms, and potential enhancements for cloud-based encryption services.

The "FileGuard" application effectively addressed the need for a secure and efficient data encryption and decryption tool using AES with CBC mode. The system's performance and security evaluations confirmed its potential as a robust solution for data protection. Future enhancements could include support for additional encryption algorithms, integration with cloud storage services, and the development of a more advanced key management system to further strengthen data security practices. The research not only contributes to practical applications in data security but also sets a foundation for further innovations in cryptographic solutions and data protection methodologies.

V. CONCLUSION AND SUGGESTION

Conclusions and recommendations can be presented in separate subsections. The conclusions section should address the research objectives, provide a concise summary of the research findings (avoiding numerical or statistical information), and highlight the main outcomes of the study. The recommendations section should suggest further research that is deemed necessary for enhancing the utility and applicability of the study's findings.

VI. BIBLIOGRAPHY

Dupré, G. (2024). Energy efficiency in AES encryption on ARM Cortex CPUs: Comparative analysis across modes of operation, data sizes, and key lengths. In.

Hafsa, A., Sghaier, A., Malek, J., & Machhout, M. (2021). Image encryption method based on improved ECC and modified AES algorithm. *Multimedia Tools and Applications, 80*, 19769-19801.

Hranický, R. (2022). *Digital Forensics: The Acceleration of Password Cracking.* Brno University of Technology Brno, Czechia,

Jimale, M. A., Z'aba, M. R., Kiah, M. L. B. M., Idris, M. Y. I., Jamil, N., Mohamad, M. S., & Rohmad, M. S. (2022). Authenticated encryption schemes: A systematic review. *IEEE Access, 10*, 14739-14766.

Lannge, L., & Sendek, S. (2024). Comparative Analysis of Programming Languages in Cryptography. In.

Lin, C.-H., Hu, G.-H., Chan, C.-Y., & Yan, J.-J. (2021). Chaos-based synchronized dynamic keys and their application to image encryption with an improved AES algorithm. *Applied Sciences, 11*(3), 1329.

Lu, Z., & Mohamed, H. (2021). A complex encryption system design implemented by AES. *Journal of Information Security, 12*(2), 177-187.

Minkovska, D., & Ivanova, M. (2021). *Security in Multimedia Information Systems: Analysis and Prediction.* Paper presented at the Methodologies and Intelligent Systems for Technology Enhanced Learning, 10th International Conference. Workshops: Volume 2.

Nabi, S. T., Kumar, M., Singh, P., Aggarwal, N., & Kumar, K. (2022). A comprehensive survey of image and video forgery techniques: variants, challenges, and future directions. *Multimedia Systems, 28*(3), 939-992.

Shakor, M. Y., Khaleel, M. I., Safran, M., Alfarhood, S., & Zhu, M. (2024). Dynamic AES encryption and blockchain key management: a novel solution for cloud data security. *IEEE Access, 12*, 26334-26343.

Sheikhpour, S., Mahani, A., & Bagheri, N. (2021). Reliable advanced encryption standard hardware implementation: 32-bit and 64-bit data-paths. *Microprocessors and Microsystems, 81*, 103740.

Sousi, A.-L., Yehya, D., & Joudi, M. (2020). Aes encryption: Study & evaluation. *CCEE552: Cryptography & Network Security. Presented to: Dr. Jad Nasreddine, Rafik Hariri University.*