Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th October 2025. Vol. 5. No. 1; e-ISSN: 2808-4519

Integration of Key Derivation Function (KDF) Development for Advanced Encryption Standard (AES) 256 Key Generator in Digital File Security

Khairi Faldi Adinata[1*], Achmad Fauzi[2], Husnul Khair[3]

[1,2,3]*Informatics Engineering, STMIK Kaputama*
Jl. Veteran No. 4A-9A, Binjai, North Sumatra, Indonesia
khairifaldiadinatalts@gmail.coml[1*], fauzyrivai88@gmail.coml[2], husnul.khair@gmail.com[3]

Abstract

Digital file security has become increasingly crucial along with the rapid development of information technology. The Advanced Encryption Standard (AES) 256 bit algorithm is a strong cryptographic solution; however, its effectiveness greatly depends on the quality of the encryption key used. The use of weak keys can significantly reduce the level of security. This research aims to enhance the security of the AES key generation process by integrating the development of a Key Derivation Function (KDF). The proposed KDF utilizes a 512 bit external key that is divided into two blocks, processed using an XOR operation, and subsequently transformed with the AES SubBytes substitution to generate a more complex 256 bit derived key. The system is implemented as a desktop application with a graphical user interface (GUI) using the Python programming language with the tkinter and cryptography libraries. The test results show that the application successfully encrypts and decrypts various digital file formats (.pdf, .docx, .xlsx, .png, .mp3, and .mp4). Encrypted files cannot be accessed and can only be restored to their original form through the decryption process with the correct key. The integration of this KDF has proven effective in strengthening the key for the AES 256 algorithm, thereby providing an additional security layer to protect digital files from unauthorized access.

Keywords: *AES, Cryptography, File Security, KDF, Python*

1. Introduction

Rapid developments in information technology have changed the way data is stored and exchanged, from personal documents to company archives, most of which are now in digital file form. As a result, data security has become a crucial aspect of protecting information from threats such as theft, modification, or misuse by unauthorized parties [1]. Cryptography offers a fundamental solution for maintaining data confidentiality and integrity. One of the most widely adopted symmetric cryptography standards is the Advanced Encryption Standard (AES), particularly the AES-256 variant, which is considered highly resistant to brute-force attacks due to its key length [2].

However, the strength of the AES algorithm is highly dependent on the quality of the encryption key used. Weak, easily guessed, or insecurely generated keys can be a significant security vulnerability. To address this issue, a Key Derivation Function (KDF) is used as a function to convert the initial input (such as a password or master key) into a more secure and random cryptographic key [3]. KDF strengthens the key to resist dictionary attacks and brute-force attacks.

This research focuses on the design and implementation of a KDF method specifically developed to be integrated with the AES-256 algorithm. The proposed KDF method is designed to generate complex 256-bit keys from 512-bit external key inputs. The main objective of this research is to integrate the KDF into a digital file encryption and decryption application, thereby providing an additional layer of security to the key generation process and improving the overall security of digital files.

2. Theoretical Foundation

2.1. Cryptography

Cryptography is the study of mathematical techniques related to maintaining information security, such as maintaining confidentiality, data integrity, and authentication. Cryptography is the process of using various techniques or sciences to protect the security of data, messages, and information. Cryptographic algorithms are mathematical functions used to encrypt and decrypt data. There are two interrelated functions, one used for encryption and the other for decryption [2].

2.2. Advanced Encryption Standard (AES)

AES is a symmetric block cipher algorithm designed to provide a high level of security with a variety of key length options. Key lengths in AES are available in 128-bit, 192-bit, and 256-bit, each of which affects the number of rounds and key structure. In AES-128, the key length (Nk) consists of 4 words, each of which is 32 bits in size, resulting in a total key length of 128 bits. This algorithm uses a block size of 128 bits and runs 10 rounds (Nr) during the encryption process [4].

The AES algorithm encryption process consists of four types of byte transformations, namely SubBytes, ShiftRows, MixColumns, and AddRoundKey. At the beginning of the encryption process, the input that has been copied into the state will undergo the AddRoundKey byte transformation. After that, the state will undergo the SubBytes, ShiftRows, MixColumns, and AddRoundKey transformations repeatedly as many times as Nr. This process in the AES algorithm is called the round function. The last round is slightly different from the previous rounds in that in the last round, the state does not undergo the MixColumns transformation.

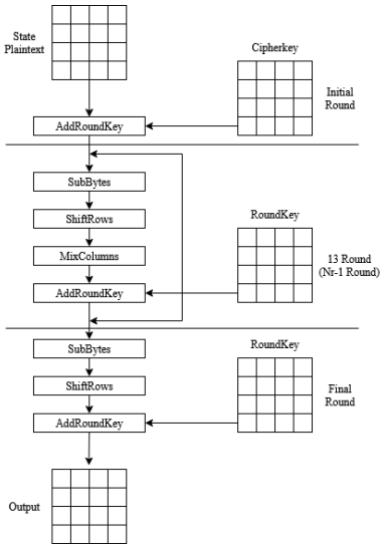


Fig. 1: AES Encryption Process Diagram

In the AES algorithm decryption process, the cipher transformation can be reversed and implemented in the opposite direction to produce an inverse cipher that is easy to understand for the AES algorithm. The byte transformations used in the inverse cipher are InvShiftRows, InvSubBytes, InvMixColumns, and AddRoundKey [2].

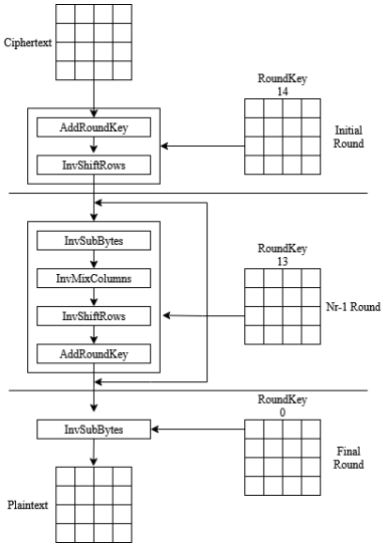


Fig. 2: AES Decryption Process Diagram

2.3. Key Derivation Function (KDF)

A Key Derivation Function (KDF) is a fundamental cryptographic algorithm that serves to derive or generate one or more secure cryptographic keys from an initial secret value, such as a password, master key, or private string. The main function of a KDF is to convert input that may not be completely random into keys that have pseudorandom properties, so that they cannot be distinguished from random binary strings of the same length. This process is crucial for protecting data during transmission or storage, as the generated keys will be used by other cryptographic algorithms for encryption or authentication [3].

3. Research Methods

The research methodology consists of several main stages, including literature study, KDF method design, integration with AES-256, implementation in the form of an application, and system testing.

3.1. Key Derivation Function (KDF) Design

The proposed KDF method is designed to generate a 256-bit derived key from a 512-bit external key through a series of simple but effective cryptographic operations. The steps of the process are as follows:

1. External Key Input: The system accepts input in the form of an external key with a length of 512 bits (64 bytes).
2. Key Segmentation: The 512-bit key is divided into two equal blocks, Block 1 and Block 2, each with a length of 256 bits (32 bytes).
3. XOR operation: A bitwise XOR operation is performed between Block 1 and Block 2. This operation aims to randomize and combine information from both parts of the key, resulting in a new 256-bit block whose value depends on the entire input key.
4. SubBytes Transformation: The result of the XOR operation is then processed through a SubBytes transformation identical to that used in the AES algorithm. Each byte of the XOR result is substituted using the standard AES S-Box table. This step adds non-linearity to the key derivation process, making it more resistant to cryptographic analysis.
5. Derived Key Output: The final result of the SubBytes transformation is a 256-bit derived key that is ready to be used for the AES encryption process.

3.2. KDF Integration with AES-256 Encryption and Decryption

The 256-bit derived key generated by the KDF is used as the master key (cipher key) for the AES-256 encryption algorithm. The process flow for encrypting digital files after the key is generated is as follows:

1. Input File and Key: The user selects the digital file to be encrypted and enters a 512-bit external key.
2. Key Generation: The system runs the KDF process as described in the previous subsection to generate a 256-bit derived key.
3. AES Key Expansion: A 256-bit derived key is used in the AES key expansion process to generate 14 round keys that will be used in each encryption round.
4. Encryption Process: Digital files are read and processed in 128-bit blocks. Each block of data undergoes 14 rounds of AES transformation (SubBytes, ShiftRows, MixColumns, AddRoundKey).
5. Cipherfile Output: All encrypted blocks are combined to form a cipherfile (encrypted file).

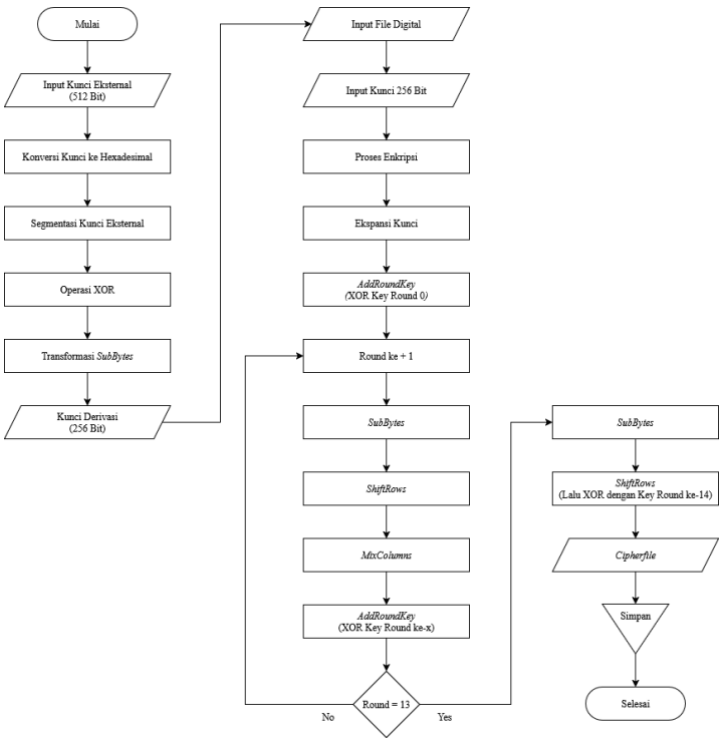


Fig. 3: KDF Integration Flowchart in the AES Encryption Process

The decryption process is the reverse of the encryption process. The user enters the cipherfile and the same 512-bit external key. A 256-bit derived key will be regenerated through KDF, then used to decrypt the file back to its original form through the inverse AES transformation.

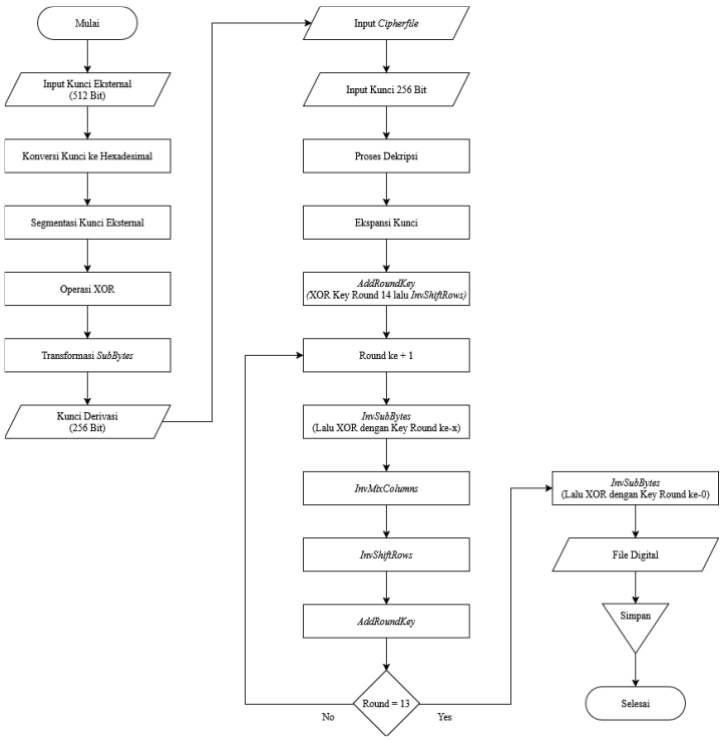


Fig. 4: KDF Integration Flowchart in the AES Decryption Process

3.3. System Implementation

This system is implemented as a desktop application with a graphical user interface (GUI) using the Python programming language. The tkinter library is used to build the interface, while the KDF and AES-256 cryptographic processes are implemented using the cryptography library, which provides modern and secure cryptographic functions.

4. Results and Discussion

The developed application provides three main functionalities: key generation (Generate Key), file encryption (Encrypt), and file decryption (Decrypt), which can be accessed through the main interface as shown in the following figure.

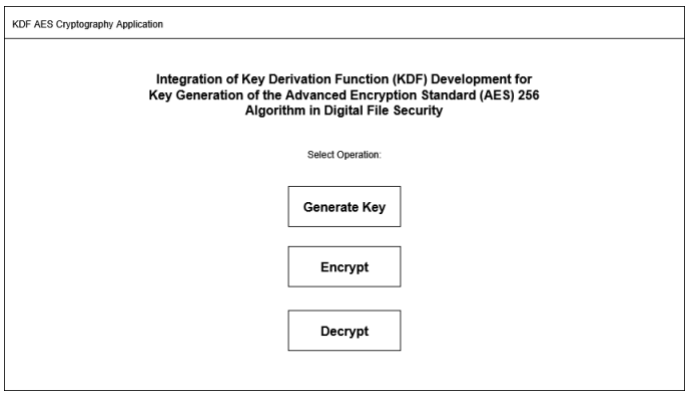


Fig. 2. Main Form

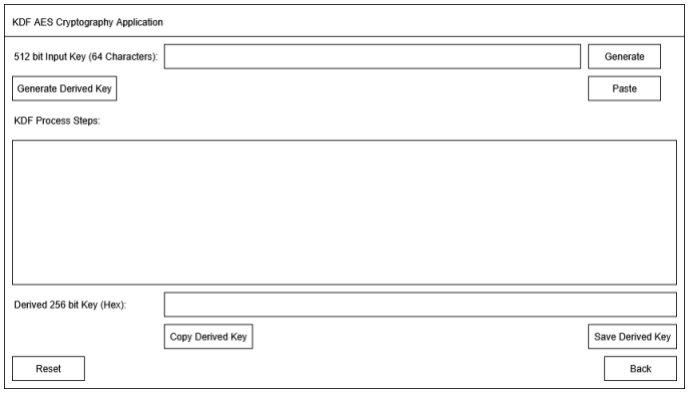


Fig. 3: Generate Key Form

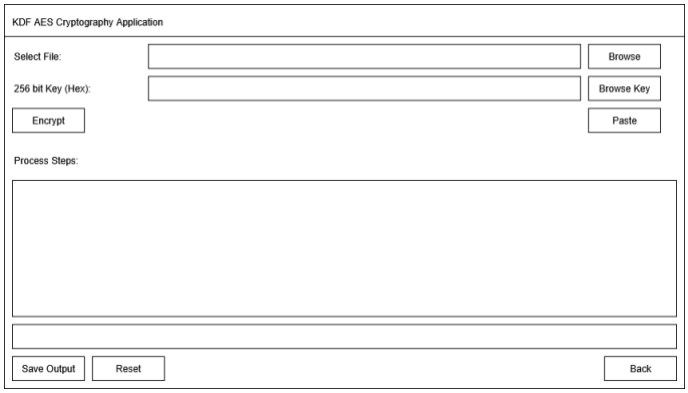


Fig. 4: Encryption Form

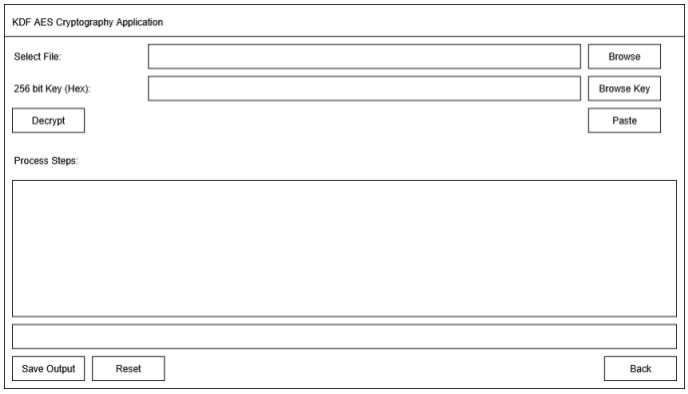
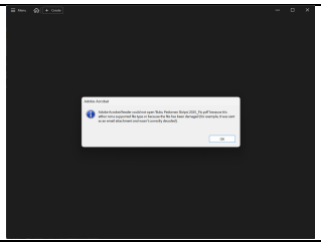
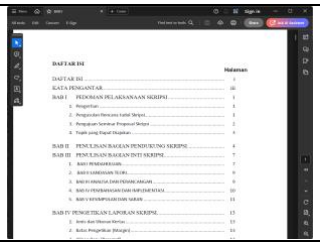


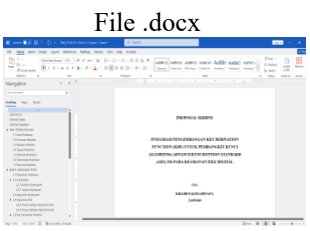
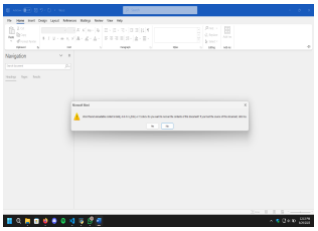
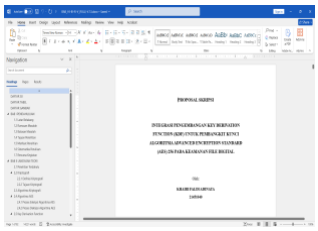
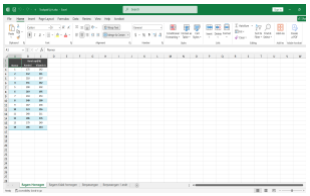
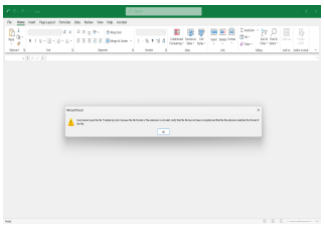
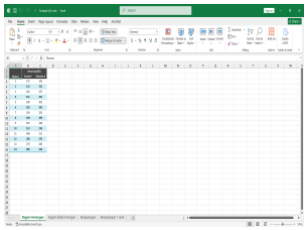
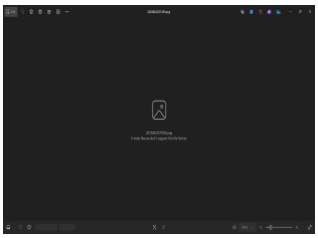
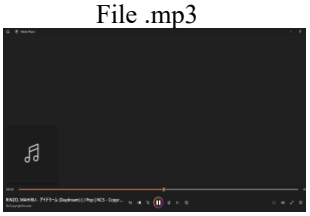
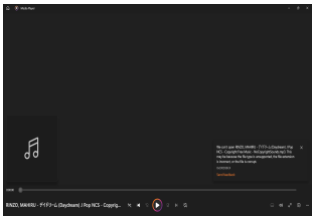
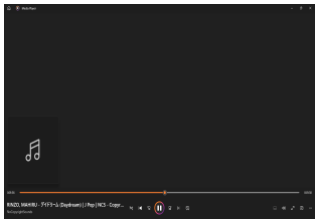
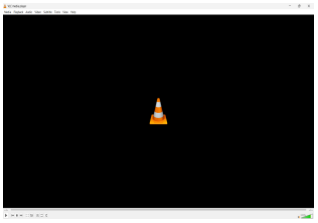
Fig. 5: Decryption Form

Functionality testing was conducted on various common digital file formats, including .pdf, .docx, .xlsx, .png, .mp3, and .mp4. The test results showed that the application successfully encrypted all of these file formats. Encrypted files cannot be opened or accessed by their standard applications, indicating that the file structure has been successfully altered and its contents secured. Furthermore, the encrypted files were successfully restored to their original form through a decryption process using the correct key.

Table 1 shows the visual results of the encryption and decryption processes on several file formats. The “Original File” column displays the file before processing, the “Encrypted File” column shows the condition of the file after encryption (cannot be opened), and the “Decrypted File” column displays the file that has been successfully recovered.

Table 1. Encryption and Decryption Test Results on Various File Formats

No.	File Asli	File Hasil Enkripsi	File Hasil Dekripsi
1.	File .pdf 		

2.	File .docx		 
3.	File .xlsx		 
4.	File .png		 
5.	File .mp3		 
6.	File .mp4		 

These results prove that the integration of the proposed KDF method with the AES-256 algorithm works successfully. The generated keys are able to secure and recover data correctly, demonstrating that this method is effective and can be implemented for digital file security needs.

5. Conclusion

This research successfully integrated a Key Derivation Function (KDF) development with the AES-256 algorithm for digital file security. The proposed KDF method effectively generates complex 256-bit derived keys from 512-bit key inputs through a combination of XOR operations and SubBytes transformations. The implementation of the system in the form of a Python-based desktop application has been proven capable of correctly encrypting and decrypting various digital file formats, such as .pdf, .docx, .png, and others. Encrypted files cannot be accessed without a decryption process using the correct key, confirming that this integration provides an additional functional layer of security to protect data from unauthorized access.

References

[1] E. Soesanto, A. Romadhon, B. D. Mardika, and M. F. Setiawan, “Analisis dan Peningkatan Keamanan Cyber: Studi Kasus Ancaman dan Solusi dalam Lingkungan Digital Untuk Mengamankan Objek Vital dan File,” Sammajiva: Jurnal Penelitian Bisnis dan Manajemen, vol. 1, no. 2, pp. 172–191, Jun. 2023, doi: 10.47861/sammajiva.v1i2.226.

[2] R. Ghinaa Sinambela, A. Fauzi, and H. Khair, “Enhancing AES Key Generation Using Diffie-Hellman Method for Image Security,” 2024. [Online]. Available: https://ioinformatic.org/

[3] C. W. Chuah, N. Z. Harun, and I. R. A. Hamid, “Key Derivation Function: Key-Hash Based Computational Extractor And Stream Based Pseudorandom Expander,” PeerJ Comput Sci, vol. 10, p. e2249, Aug. 2024, doi: 10.7717/PEERJ-CS.2249/SUPP-1.

[4] AR. Nurjaman and A. T. Turnip, "Kombinasi Algoritma Kriptografi AES-256 Dan SHA3-512 Untuk Meningkatkan Keamanan Dokumen PDF," 2024.
[5] Muh. Fitrah and D. Kusnadi, "Integrasi Nilai-Nilai Islam Dalam Membelajarkan Matematika Sebagai Bentuk Penguatan Karakter Peserta Didik," 2022.
[6] B. D. Kurniawan, M. A. Rosid, I. A. Kautsar, and N. E. Pratama, "Rancang Bangun Library Web Token untuk Enkripsi HTTP Data Menggunakan Eksklusif-OR (XOR)," Physical Sciences, Life Science and Engineering, vol. 1, no. 1, p. 14, Jan. 2024, doi: 10.47134/pslse.v1i1.164.
[7] Z. Fauziah and I. I. Lawanda, "Pengelolaan Arsip Digital Pribadi: Studi Kasus Mahasiswa Manajemen Rekod Dan Arsip Ui," Multikultura, Oct. 2024, doi: 10.7454/multikultura.v3i4.1069.
[8] Hayyina Farahdiba, Christian Wiradendi Wolor, and Marsofiyati Marsofiyati, "Analisis Pengelolaan Arsip Digital Pada PT Anugrah Alam Karunia Abadi," Journal Of Administrative And Social Science, vol. 5, no. 1, pp. 41–53, Dec. 2023, doi: 10.55606/jass.v5i1.807.
[9] N. Sitohang, "Penerapan Metode Five Modulus Dalam Mengkompresi File Dokumen (PDF) Pengembangan Bahan Ajar," Bulan Februari, 2023.
[10] Maolana, "Analisis Perbandingan Hasil Kompresi Citra Format PNG dengan SVG untuk Penyimpanan File Gambar," vol. 29, 2024.
[11] T. B. Situmorang, "Perancangan Aplikasi Kompresi File MP3 Dengan Menggunakan Algoritma Lempel Ziv Welch (LZW)," 2023.