International Journal of Informatics and Computation (IJICOM)
Vol. 8, No. 1, 2026
ISSN: 2685-8711, E-ISSN: 2714-5263 DOI: 10.35842/ijicom

Implementation Analysis of AES-256 and SHA-256 in EOF Steganography

Rini Indrayani

Abstract
The rapid advancement of information technology has increased the demand for data security mechanisms that not only provide strong cryptographic protection but also conceal the existence of sensitive information. This study proposes a multi-layered security scheme that integrates cryptography and steganography to protect secret messages. The process begins by encrypting plaintext using the AES-256 algorithm, where the encryption key is generated from a 256-bit SHA-256 hash function. The resulting ciphertext is then embedded into digital media using the End-of-File (EOF) steganography method. Experimental results demonstrate that SHA-256 consistently produces a 32-byte output suitable for secure key generation, while AES-256 effectively transforms plaintext into ciphertext through a block-based padding mechanism, ensuring data confidentiality before embedding. Furthermore, the EOF method successfully hides the ciphertext within various file types without altering their visual appearance or primary functionality, although the file size increases proportionally to the embedded message. Thus, the integration of SHA-256, AES-256, and EOF steganography provides an effective and practical approach to multi-layered data security.

Keywords:
Steganography, EOF, AES-256, SHA-256

1. Introduction

The rapid growth of digital communication continuously increases the need for stronger information security mechanisms. Researchers widely adopt cryptography to protect data confidentiality and integrity during transmission. The National Institute of Standards and Technology standardizes the Advanced Encryption Standard (AES), and AES-256 becomes one of the most trusted symmetric encryption schemes due to its large key size and resistance to brute-force attacks. Studies demonstrate that AES provides strong diffusion and confusion properties suitable for securing multimedia data and textual information before embedding into cover media. However, encryption alone does not conceal the existence of communication; it only transforms plaintext into ciphertext. Attackers can still detect encrypted traffic and raise suspicion. This limitation motivates researchers to combine encryption with steganography to hide not only the content but also the presence of secret communication [1], [18], [28].

Steganography addresses this limitation by embedding secret data within digital media such as images, audio, or video. Researchers often implement Least Significant Bit (LSB) or transform-domain techniques to minimize perceptual distortion. Nevertheless, conventional LSB methods expose vulnerabilities to statistical steganalysis, compression, and noise interference. Several studies enhance LSB by integrating AES encryption and pixel selection strategies to strengthen payload security and unpredictability. These works show that combining encryption and steganography increases resistance against

Corresponding Author: Rini Indrayani(riniindrayani@umpalopo.ac.id)
1 Rini Indrayani, Universitas Muhammadiyah Palopo, riniindrayani@umpalopo.ac.id

extraction attacks, yet they still struggle with robustness when subjected to file manipulation or format conversion. Therefore, the security design must consider both invisibility and cryptographic strength simultaneously [4], [11], [18].

Hash functions, particularly SHA-256, play a crucial role in ensuring data integrity and authentication. SHA-256 generates a fixed 256-bit digest that uniquely represents input data and detects even minor modifications. Researchers integrate SHA-256 with AES to provide layered protection, where AES secures confidentiality, and SHA-256 verifies authenticity. In steganographic systems, hash-based preprocessing also determines embedding positions or generates dynamic keys. Despite these advantages, many implementations treat hashing merely as an auxiliary feature rather than a core structural component. As a result, they do not fully evaluate how hash-based control mechanisms influence embedding security and extraction reliability [2], [3], [12].

Hybrid cryptography–steganography frameworks continue to evolve in response to emerging cyber threats. Researchers propose AES-based secure image steganography with dynamic key derivation and hash-controlled embedding to mitigate brute-force and differential attacks. These frameworks aim to achieve three main objectives: confidentiality, integrity, and imperceptibility. However, practical implementations often prioritize visual quality metrics such as PSNR and MSE without deeply analyzing structural vulnerability, especially in file-level embedding approaches. Consequently, many systems demonstrate high image quality but lack a comprehensive evaluation of cryptographic resilience under adversarial scenarios [6], [16], [17].

Another relevant development focuses on robust steganography in encrypted domains. Researchers explore embedding data into already encrypted images to increase unpredictability and resist steganalysis. Such approaches highlight the importance of maintaining reversibility and preventing information leakage during extraction. Nevertheless, embedding inside transform domains (DCT, DWT) or encrypted domains frequently introduces computational complexity and synchronization challenges. These issues become critical when the system targets lightweight applications or large-scale data transmission. Therefore, simpler yet secure embedding strategies remain attractive for practical deployment [19], [25].

End-of-File (EOF) steganography represents one such simple strategy. Instead of modifying pixel values, EOF techniques append secret data at the end of a digital file without altering its visual structure. This approach preserves image quality completely and avoids statistical distortion within pixel distributions. However, EOF steganography faces detection risks because forensic analysis can identify abnormal file sizes or appended data segments. Without strong encryption and hashing mechanisms, attackers can easily extract or manipulate the hidden payload. Thus, integrating AES-256 encryption and SHA-256 hashing into EOF methods becomes essential to reinforce confidentiality and integrity while maintaining structural invisibility [1], [23], [30].

Recent surveys on cryptographic steganography emphasize the need for multi-layered security designs that combine encryption, hashing, and adaptive embedding mechanisms. Researchers argue that secure communication systems must resist not only passive eavesdropping but also active tampering and statistical analysis. Despite significant progress, limited studies explicitly evaluate the performance of AES-256 and SHA-256 when implemented within EOF-based steganography frameworks. Most prior works focus on LSB or transform-domain embedding, leaving a research gap in analyzing file-structure-based concealment combined with modern cryptographic primitives [7], [29], [26].

Based on these observations, this study evaluates the integration of AES-256 encryption and SHA-256 hashing within an EOF steganography framework. The research addresses key issues: (1) how effectively AES-256 protects appended payload data, (2) how SHA-256 enhances integrity verification and key management, (3) how EOF embedding affects detectability and file consistency, and (4) how the combined system

performs in terms of security, robustness, and computational efficiency. By focusing on both cryptographic strength and structural concealment, this study aims to fill the identified research gap and provide a comprehensive evaluation model for secure data hiding systems [2], [6], [25].

2. Related Works

Researchers extensively investigated the integration of encryption and steganography to strengthen secure communication systems. Saraireh and Matarneh combined AES encryption with Discrete Wavelet Transform (DWT) steganography to protect transmitted data and reported improved confidentiality and acceptable image quality performance [1]. Their study demonstrated that pre-encrypting secret messages using AES significantly reduced the risk of meaningful extraction even when attackers detected the presence of hidden data. However, they primarily evaluated visual quality metrics such as PSNR and did not thoroughly analyze resilience against advanced steganalysis or structural file inspection. Their work focused on transform-domain embedding rather than file-structure-based techniques, leaving limitations in evaluating alternative embedding strategies.

Several studies enhanced Least Significant Bit (LSB) steganography by incorporating AES and hash functions to improve embedding randomness. Singhal et al. integrated AES encryption and SHA-256 hashing within an LSB framework and reported increased security due to layered protection [2]. Their method used hashing to validate data integrity and encryption to conceal content. Although they achieved improved resistance against direct payload extraction, the LSB modification still altered pixel distributions, making the method potentially vulnerable to statistical steganalysis. The study emphasized encryption strength but paid limited attention to structural detectability and file-level anomaly detection.

Cheddad et al. proposed a hash-based image encryption and steganographic approach using SHA-2 to generate embedding keys and enhance unpredictability [3]. They demonstrated that cryptographic hashing could effectively randomize embedding positions and reduce pattern-based detection. Their findings highlighted the importance of integrity verification before and after extraction. However, their work concentrated on image-domain embedding and did not explore lightweight file-appending methods such as EOF steganography. Additionally, the study primarily evaluated algorithmic functionality rather than comprehensive adversarial robustness.

Gangurde and Tiwari introduced an AES-secured LSB technique that utilized a pixel locator sequence to increase embedding randomness [4]. They reported improved security compared to conventional LSB due to controlled pixel selection and encrypted payloads. Their method strengthened confidentiality and reduced predictable embedding patterns. Nevertheless, the approach still relied on pixel modification, which inherently introduced statistical artifacts detectable through modern steganalysis tools. The authors did not investigate alternative embedding domains that preserve original pixel distributions completely.

Firdaus and Rahmatulloh implemented AES-256 encryption with pseudorandom embedding in digital images and demonstrated enhanced resistance against unauthorized extraction [13]. Their results confirmed that AES-256 provided strong brute-force resistance due to its 256-bit key length. They showed satisfactory visual quality and computational feasibility. However, the embedding process modified image pixels and, therefore, remained susceptible to compression and transformation attacks. The study did not incorporate hashing mechanisms to verify data integrity or protect against tampering.

Castillo Soria et al. developed a lossless data hiding scheme combining AES encryption with DWT-based embedding [19]. They emphasized reversibility and ensured that the original image could be reconstructed without distortion after extraction. Their work contributed significantly to reversible data hiding research and demonstrated the importance of maintaining data fidelity. Despite these strengths, the transform-domain

approach increased computational complexity and required careful synchronization between sender and receiver. The method did not consider file-structure-based hiding techniques that avoid altering image coefficients.

Yang et al. investigated robust steganography in encrypted images and analyzed embedding strategies within encrypted domains [25]. They highlighted the advantage of concealing data after encryption to prevent information leakage and resist steganalysis. Their findings showed improved unpredictability and theoretical robustness. However, embedding within encrypted domains introduced additional processing overhead and key management complexity. The study targeted image-domain encryption and did not evaluate simpler approaches, such as appending encrypted data at the end of files.

Recent surveys by Alenizi et al. and Nguyen et al. reviewed cryptographic steganography techniques and identified common trends and research gaps [7], [29]. They concluded that hybrid systems combining encryption, hashing, and adaptive embedding mechanisms offered stronger protection than standalone approaches. However, they also noted that most existing studies concentrated on LSB, DCT, or DWT domains and rarely examined file-structure-based methods such as EOF steganography integrated with modern cryptographic primitives. These surveys highlighted the need for systematic evaluation of AES-256 and SHA-256 within alternative embedding frameworks. Their observations directly motivated further investigation into combining strong encryption, secure hashing, and EOF steganography to address confidentiality, integrity, and detectability challenges simultaneously.

3. Proposed Method

This research consists of various stages of data security to protect secret messages (plaintext). The security process begins with processing the plaintext using the 256-bit AES encryption method, whose encryption key uses the results of processing a 256-bit hash function. The result of the encryption process is called ciphertext. The ciphertext is then inserted into the data security media using the End of File (EOF) steganography method.

In this paper, we utilize a hybrid security framework that integrates AES-256 encryption**,** SHA-256 hashing**,** and End-of-File (EOF) steganography to provide confidentiality, integrity, and structural invisibility. We design the method in three sequential layers: (1) cryptographic encryption, (2) hash-based integrity control and key derivation, and (3) EOF-based embedding. We first encrypt the secret message before embedding it to ensure that even if an attacker extracts the hidden payload, the content remains unintelligible. We then compute a cryptographic hash to verify authenticity and prevent tampering. Finally, we append the protected payload to the end of the cover file without modifying its internal pixel or structural components. This design preserves original media quality while strengthening security at multiple levels.

We denote the secret message as $M \in \{0,1\}^*$. We generate a 256-bit symmetric key K and encrypt the message using AES-256 in Cipher Block Chaining (CBC) mode. The encryption process produces ciphertext C defined as:

$$C = \text{AES}_{256}(M, K, IV) \tag{1}$$

where IV is a randomly generated initialization vector. AES-256 operates on 128-bit blocks and applies 14 transformation rounds, ensuring strong confusion and diffusion properties. We utilize AES-256 because its key space 2^{256} provides strong resistance against brute-force attacks. The encryption layer guarantees confidentiality before any embedding process occurs.

To ensure integrity and authenticate the encrypted payload, we apply SHA-256 hashing to the ciphertext. We compute the hash value H as:

$$H = \text{SHA256}(C) \tag{2}$$

where $H \in \{0,1\}^{256}$. The hash function produces a fixed-length digest that uniquely represents the ciphertext. If any modification occurs during transmission or extraction, the recomputed hash $H' \neq H$ indicates tampering. We also utilize the hash value for lightweight key validation by binding the encryption key with a derived verification value:

$$K_v = \text{SHA256}(K \parallel H) \tag{3}$$

where $\parallel$ denotes concatenation. This step strengthens resistance against unauthorized decryption attempts.

After encryption and hashing, we construct the protected payload P as a concatenation of structured components:

$$P = \text{Header} \parallel IV \parallel C \parallel H$$

The header stores metadata such as payload length $| C |$ and validation markers to support reliable extraction. We then apply EOF steganography by appending the payload directly to the cover file F. The stego file F_s is generated as:

$$F_s = F \parallel P$$

This approach does not alter the original pixel values, frequency coefficients, or visual structure of the file. Consequently, we preserve image quality completely and avoid statistical distortion commonly found in LSB or transform-domain methods.

During extraction, we read the stego file F_s, parse the appended segment based on the predefined header, and retrieve IV, C, and H. We recompute the hash:

$$H' = \text{SHA256}(C)$$

If $H' = H$, we proceed with decryption:

$$M = \text{AES}_{256}^{-1}(C, K, IV) \tag{4}$$

Otherwise, we reject the payload and report integrity failure. This verification step ensures that attackers cannot modify the ciphertext without detection.

At the final stage, we apply a layered security model where AES-256 guarantees confidentiality, SHA-256 ensures integrity and key validation, and EOF steganography preserves structural invisibility. This paper utilizes this integrated framework to achieve three security objectives simultaneously: (1) resistance against brute-force and cryptanalytic attacks, (2) protection against tampering through hash verification, and (3) elimination of statistical detectability by avoiding pixel modification. The structured mathematical formulation clarifies each processing stage and supports systematic evaluation of computational efficiency, robustness, and security performance. Fig. 1 depicts the entire process in this research

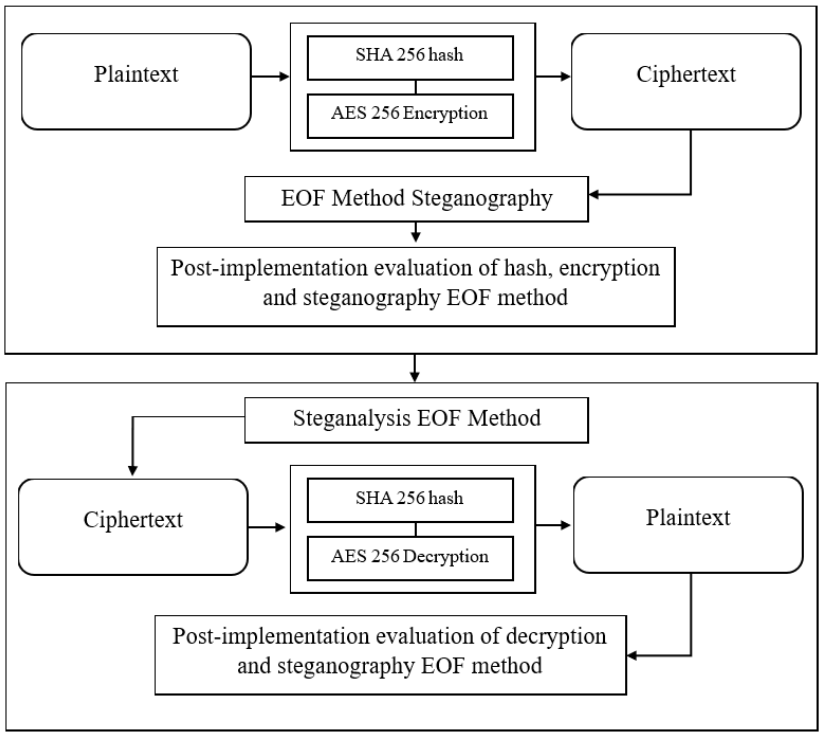

Fig 1. Research Flow

4. Experimental Setup

4.1. 256-Bit Hash Function

The 256-bit hash function is implemented using the Secure Hash Algorithm 256 (SHA-256) algorithm as a data integrity verification mechanism. SHA-256 generates a 256-bit (32-byte) message digest from a variable-length message through a series of standardized logical operations, bit shifts, and block compression[7]. The deterministic, one-way, and collision-resistant nature of SHA-256 ensures that any changes to the message will result in a significantly different hash value[8].

Technically, the hashing process begins with a preprocessing stage that involves padding the message until its length is congruent with 448 modulo 512, then adding a 64-bit representation of the original message length at the end. The processed message is then divided into 512-bit blocks[9]. Each block is processed through 64 rounds of a compression function that uses eight working variables (a, b, c, d, e, f, g, h) with fixed constants (round constants) and bitwise operations such as ROTR, SHR, XOR, AND, and NOT. The results of each block will update an intermediate hash value until the entire block is processed and produces a final 256-bit hash value

4.2. 256-Bit AES Encryption

In this study, message confidentiality was ensured before embedding into EOF steganography media using the Advanced Encryption Standard algorithm with a key length of 256 bits (AES-256). AES is a symmetric block cipher algorithm that operates on a fixed block size of 128 bits and has become the international standard for digital data security. AES-256 was chosen based on its extremely large key space complexity (2^{256} possible

keys) and its resistance to various modern cryptanalysis techniques, thus ensuring strong confidentiality for the embedded message[10].

Technically, the AES-256 encryption process begins by dividing the plaintext into 16-byte (128-bit) blocks. Each block is represented by a 4x4 byte matrix called a state. The encryption process proceeds through 14 rounds of sequential transformations. Each round consists of four main operations: SubBytes (non-linear substitution using S-Boxes), ShiftRows (row shifts in the state matrix), MixColumns (column mixing using the Galois Field GF(2^8) operation), and AddRoundKey (XOR operation between the state and the round key). In the final round, the MixColumns operation is not performed. This series of transformations is designed to achieve high data diffusion and confusion[11].

The 256-bit master key used in the encryption process is not used directly in each round but is first processed through a key expansion algorithm to generate 15 round keys. This key expansion process involves byte rotation (RotWord), S-Box substitution (SubWord), and the addition of a round constant (Rcon)[12]. This mechanism ensures that each round uses a different key, increasing cryptographic complexity and making key analysis difficult for unauthorized parties.

4.2. EOF Method Steganography

End-of-File (EOF) steganography is one of many steganographic methods used to hide secret messages. This method inserts a secret message at the end of a file after the bits that make up the file structure end[1]. The challenge of this method is ensuring that the bits of the secret message do not overwrite the bits that make up the test file structure. Moreover, each file type has a different structure. Fig. 2 depicts an example of inserting a secret message using the EOF method.

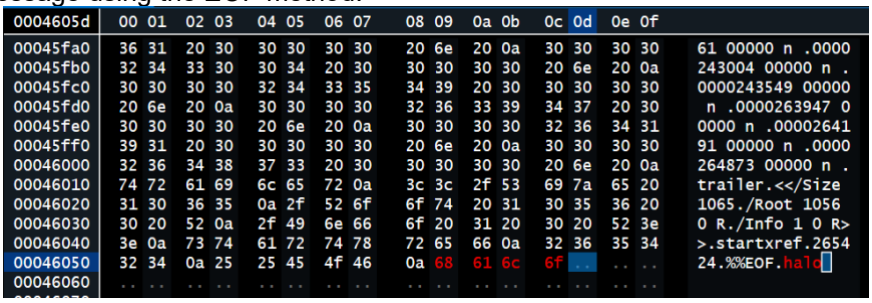

Fig 2. Example of the EOF Method steganography before Cryptography Implementation

Fig. 2 shows that secret messages inserted directly without any additional cryptographic processing are at high risk of being easily read by third parties. Therefore, additional cryptographic processing is essential to ensure the obscurity and security of secret messages. An example of inserting a secret message using cryptography into a test file using the EOF steganography method can be seen in Fig. 3.

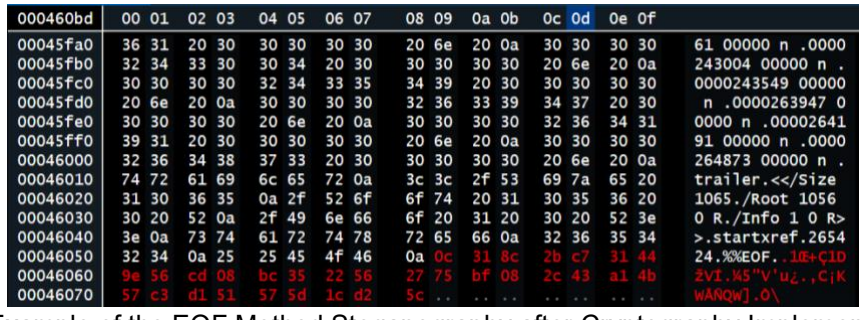

Fig 3. Example of the EOF Method Steganography after Cryptography Implementation

4.3 Steganalysis EOF Method

Steganography is the process of analyzing a test file to reconstruct the secret message embedded in it using the method used in steganography. The steganalysis process begins by identifying the file structure and searching for the end-of-file tag. Typically, each file type has its own tag, or the secret message is initially embedded with a tag to facilitate its re-creation by the infiltrator. Once the tag is found, the bits of the secret message can be reassembled into a complete secret message. Fig. 4 depicts an example of identifying file structure tags in the steganalysis process.

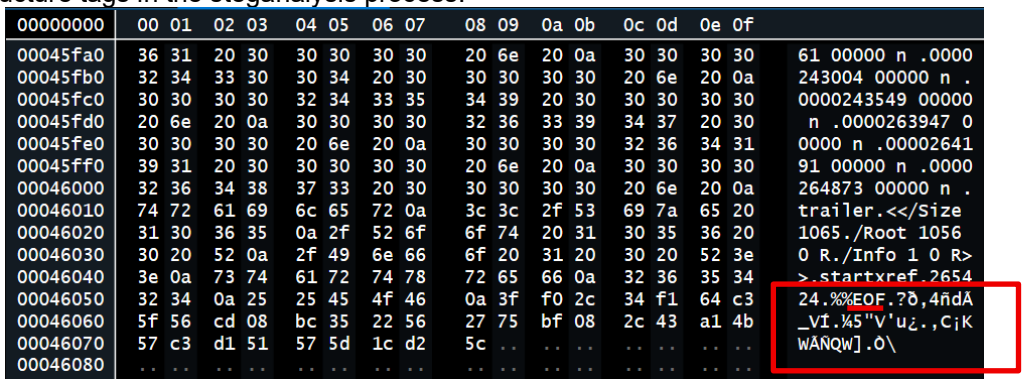

Fig 5. Example of Identifying the End-of-File Marker Structure

4.4. AES 256 Method Description

The AES-256 decryption process in this study aims to restore the encrypted ciphertext to its original plaintext form after data is extracted from the EOF steganography medium. Decryption is performed using the same 256-bit key used in the encryption process, in accordance with the characteristics of symmetric cryptography algorithms. The AES-256 decryption process is the inverse operation of the encryption stage, which also takes place in 14 rounds. Each decryption round consists of four main transformations that are the inverse of the encryption process: InvShiftRows, InvSubBytes, InvMixColumns, and AddRoundKey. The process sequence begins with AddRoundKey using the final round key resulting from key expansion, followed by a series of inverse transformations until reaching the first round. In the final round of decryption, the InvMixColumns operation is omitted, in keeping with the structure of the final round of encryption[12].

Similar to encryption, decryption utilizes the key expansion of the 256-bit master key to generate 15 round keys, which are used in reverse order, starting from the 14th round key and continuing through the initial round key. This reversed use of round keys ensures that each transformation stage accurately restores the state to its previous state until a plaintext identical to the original message is obtained.

5. Result and Analysis

The implementation of hash functions, 256-bit AES encryption, and EOF steganography methods provides both security and impact on the test files used. Therefore, it is necessary to conduct both a structural and visual evaluation of these impacts to illustrate the extent to which they cause changes in line with one of the security objectives, namely, obscuring the presence of secret messages.

5.1. Post-Implementation Evaluation of Hash Functions

The 256-bit SHA hash function used is implemented on the characters used as the 256-bit AES encryption input key. The results of the hash function implementation on the 256-bit AES input key used in this study are described in Table 1.

Table 1. The Results of The Hash Function Implementation

Input (before Hash)	Number of Inputs (Bytes)	Output (after Hash)	Output Amount (Bytes)
987654321	9	8a9bcf1e51e812d0af8465a8dbcc9f741064 bf0af3b3d08e6b0246437c19f7fb	32
Mnbvcxzasd12345	15	e359edc5a85318e9770ab3b57a27dfdd198 a0dfbb27d005295b8c06152a5bb7c	32
%^&*()QWE!@#$G HJK	17	33644c14b968be98377bb7b87c0deec224 17d7c7e5b13dbc9696ecd005786352	32
QWERTY!@#$%&* (123456	20	d768e46a53f81154811a21763be22e23f1a c1451e4df34657c88b67f3c511c3d	32
MnMn23456!@#$ %^NBMHQW	21	ffef8284e2358050f81b09c161e400e999d4 b5f889e1652f81d7c3a790416e7e	32

Table 1 shows that regardless of the number of inputs used, the output size is consistently 32 bytes. These results indicate that the use of a 256-bit hash function closely aligns with the input requirements of 256-bit AES encryption, which is also 36 bytes. This enhances the hash function's functionality, not only as a data security measure, but also as a way to simplify the selection of a 256-bit AES input key without having to manually select 16 characters.

5.2. Post-Implementation Evaluation of AES Encryption

Using 256-bit AES encryption on secret messages (plaintext) before embedding them in steganographic media files impacts the secret message itself. The impact of using 256-bit AES encryption on the plaintext is outlined in Table 2.

Table 2. The Impact of Using AES 256-Bit Encryption

Text before Encryption (Plaintext)	Plaintext Size (bytes)	Text after Encryption (Ciphertext)	Ciphertext Size (bytes)
Hello	4	53616c7465645f5f56cd08bc3522562775bf082c43a1 4b57c3d151575d1cd25c	32
Good afternoon all	22	2c6ac674d5a45f5fcf7b1ceca4b8a8944a2959adac5d 36595c77f201e88c8b1eb8df38cf93999f21b7fb949ed 27c06bc	48
where is the passion today?	31	f02d4a756a6450ff63da29d956901e2e415168291743 8a296d9bdae2edfde7914b4f2805f15f92bc8be75803 3218e5bc	48
How are you	53	3fc50258c0349febdd4efee6aa012142feb26b9a9c1b cbb5c3cef6cdeeee0eeb5eecaa865987828833c0b4e	80

today? I hope you are always healthy and blessed.		0a70b08292e842cf991c5b68673e4d9d68000b2bf528d06cce75a2f6181e9e9833e473dc6	
May a bright day always be accompanied by good hopes and may these good hopes be granted	105	08c51f0c480e37b26f7c767286e76300eb4c221d2523988689b2528ef2153ff9788a9e13d1c0a485dffacabb827af6c20ded428566300349bfb513eb0334173f877ddc8940f8f2d986aa93bff5d8d2dcf9f537a057861df825b64725e51a5e9896bfa3dd2a6868183dfb01f8afff32d80be016d47687409f6b788221e425948b	128

Table 2 shows the change in size from plaintext (the secret message before being processed with 256-bit AES encryption) to ciphertext (the secret message after being processed with 256-bit AES encryption). The greater the number of plaintext characters, the greater the number of ciphertext characters. However, in the second and third plaintexts, both have the same number of ciphertext characters even though they have different numbers of plaintext characters. This occurs due to the use of padding in the encryption process. The 256-bit AES encryption method is one of the encryption methods that uses a block-cipher technique that involves a padding process or adding blocks to complete the blocks used. So if the number of output blocks does not match the AES rules, then an additional block will occur which also affects the size and number of output encryption results.

5.3. Post-Implementation Evaluation of EOF Steganography Method

After all keys are processed with a 256-bit SHA hash function and processed together with the plaintext using 256-bit AES encryption, the output in the form of ciphertext is inserted into the test file used. This study used five types of test files with different file types: pdf, xls, jpg, mp3, and mkv. The implementation of the EOF steganography method appears to affect the test file structure. The increase in the size of the test file after the implementation of the EOF steganography method is shown in Table 3.

Table 3. Impact of Steganography on File Size

File Name	File Type	Ciphertext Size	Size before Steganography (bytes)	Size after Steganography
Coba1	.pdf	32	187309	187347
Coba2	.xls	48	159886	159934
Coba3	.jpg	48	1812749	1812797
Coba4	.mp3	80	778496	778576
Coba5	.mkv	128	2133765	2133893

5.4. Post-Implementation Evaluation of EOF Method Steganalysis

The steganalysis process is to identify the secret message inserted at the end of the test file and reassemble it to be arranged as before being inserted into the test file without changing the structure of the test file itself. However, because the previous EOF steganography method involved the process of adding structure to the end of the file, so the structure of the test file was changed and could not be returned to the way it was before the steganography process. Therefore, it is recommended to duplicate the original test file before carrying out the EOF steganography process so that the original test file remains available if needed at any time.

In addition to evaluating the test file structure, an evaluation was also conducted on the test file size. The evaluation results show that the steganalysis process does not change the test file size. The test file size remains the same as before the steganalysis process. This is relevant to the previous evaluation which showed that steganalysis does not change the test file structure, the test file size also remains unchanged. The results of the test file size evaluation are shown in Table 4.

Table 4. The File Size Evaluation

File Name	File Type	Plaintext Size	Size before Steganalysis (bytes)	Size after Steganalysis (bytes)
Coba1	.pdf	32	187309	187347
Coba2	.xls	48	159886	159934
Coba3	.jpg	48	1812749	1812797
Coba4	.mp3	80	778496	778576
Coba5	.mkv	128	2317367	2317495

5.5. Post-Implementation Evaluation Description

After all the ciphertexts were successfully collected from the steganalysis process, they were then decrypted to return them to their pre-encryption plaintext form. The evaluation results showed that all the ciphertexts could be restored to their original size, remaining the same as before encryption. The evaluation results are shown in Table 5.

Table 5. Evaluation of the Matching of Secret Messages After the Decryption Process

Text before Decryption (Ciphertext)	Ciphertext Size (bytes)	Text after Decryption (Plaintext)	Plaintext Size (bytes)
0c318c2bc731449e56cd08bc3522562775bf082c43a14b57c3d151575d1cd25c	32	Hello	4
2c6ac674d5a45f5fcf7b1ceca4b8a8944a2959adac5d36595c77f201e88c8b1eb8df38cf93999f21b7fb949ed27c06bc	48	Good afternoon all	22
f02d4a756a6450ff63da29d956901e2e4151682917438a296d9bdae2edfde7914b4f2805f15f92bc8be758033218e5bc	48	Where is the passion today?	31
3fc50258c0349febdd4efee6aa012142feb26b9a9c1bcbb5c3cef6cdeeee0eeb5eecaa865987828833c0b4e0a70b08292e842cf991c5b68673e4d9d68000b2bf528d06cce75a2f6181e9e9833e473dc6	80	How are you today? I hope you are always healthy and blessed.	53

08c51f0c480e37b26f7c767286e763 00eb4c221d2523988689b2528ef21 53ff9788a9e13d1c0a485dffacabb82 7af6c20ded428566300349bfb513eb 0334173f877ddc8940f8f2d986aa93 bff5d8d2dcf9f537a057861df825b64 725e51a5e9896bfa3dd2a6868183d fb01f8afff32d80be016d47687409f6 b788221e425948b	128	May a bright day always be accompanied by good hopes, and may these good hopes be granted	105

6. Conclusion

This study demonstrates that integrating SHA-256, AES-256, and End-of-File (EOF) steganography creates an effective multi-layered data security scheme. We utilize the SHA-256 hash function to generate consistent 32-byte outputs, which serve as a reliable basis for AES-256 key generation. The AES-256 encryption process transforms plaintext into ciphertext using a block-based padding mechanism, significantly enhancing data confidentiality before embedding. We then utilize EOF steganography to embed the encrypted message into various file types without altering their visual appearance or core functionality, although the file size increases proportionally to the embedded data. The extraction and decryption processes successfully restore the original message, confirming that the overall system operates accurately and consistently.

This study also shows that the primary limitation of the EOF method can be effectively mitigated through cryptographic integration. By combining hashing and encryption before embedding, we ensure that even if hidden data is detected, the content remains protected. The layered approach strengthens confidentiality and improves the overall robustness of the security model. The results confirm that cryptography and steganography can complement each other in building practical and secure data protection mechanisms.

However, this study has certain limitations. The evaluation was conducted under controlled conditions with specific file types and message sizes. Future research should involve broader experimental scenarios, including larger message variations, more diverse media formats, and resilience testing against advanced digital forensic and steganalysis techniques. Such an extended evaluation would provide deeper insight into system scalability, durability, and real-world applicability.

References

1. S. M. Saraireh and A. M. Matarneh, “Higher level security approach for data communication system based on AES cryptography and DWT steganography,” *JJCIT*, vol. 18, no. 3, pp. 100-110, 2016.
2. V. Singhal, Y. K. Shukla, and N. Prakash, “Image steganography embedded with Advance Encryption Standard (AES) securing with SHA-256,” *Int. J. Innovative Tech. & Exploring Eng.*, vol. 9, no. 8, pp. 641-645, 2020.
3. A. Cheddad, J. Condell, K. Curran, and P. McKevitt, “A hash-based image encryption algorithm using SHA-2 for secure steganography,” *Proc. Int. Conf. Secure Multimedia Systems*, 2012.
4. S. Gangurde and K. Tiwari, “LSB steganography using pixel locator sequence with AES,” *arXiv preprint* arXiv:2012.02494, 2020.
5. Jayeeta Majumder and C. Pradhan, “Pixel value differencing based image steganography using AES and SHA-2 cryptography method,” *Int. J. Recent Tech. Eng.*, vol. 8, no. 5, pp. 5329-5335, 2020.
6. H. S. Banoori et al., “An improved hybrid image steganography method using AES algorithm,” *Sci. Rep.*, 2025.

7. A. Alenizi et al., “A review of image steganography based on multiple methods,” *Comput. Mater. Contin.*, vol. 78, pp. 1235-1250, 2024.
8. *Adaptive Invisible Steganography for Securing Medical Images*, J. Judy, *Springer J.*, 2025.
9. A. Sykot et al., “Multi-layered security system: integrating quantum key distribution with classical cryptography to enhance steganographic security,” *Aerospace Science and Technology*, Elsevier, 2025.
10. “Secure image transmission using multilevel chaotic encryption and video steganography,” *Algorithms*, MDPI, 2025.
11. “Image steganography using LSB and hybrid encryption algorithms,” *Appl. Sci.*, vol. 13, no. 21, 2023.
12. “Securing of AES based on secure hash algorithm for image steganography,” *Int. J. Health Sci.*, vol. 6, no. S2, 2022.
13. M. Akbar Firdaus and A. Rahmatulloh, “Implementasi steganografi gambar digital LSB menggunakan enkripsi AES-256 and embedding pseudorandom,” *J. Inform. Tek. Elektro Terapan*, 2023.
14. “Crypto-Stego: a hybrid method for encrypting text messages within images using AES and LSB algorithms,” *Int. J. Intelligent Syst. Appl. Eng.*, 2024.
15. “Cryptosystem for secure data transmission using AES and steganography,” *Int. J. Sci. Res. Eng. Dev.*, 2019.
16. M. Kamil and A. Mohammed, “Secure image steganography using AES-CBC encryption and dynamic key derivation,” *Int. J. Adv. Sci. Res. Eng.*, 2025.
17. “Image encryption and steganography method based on AES algorithm and secret sharing algorithm,” *IIETA International J.*, 2024.
18. X. Tauhid et al., “A secure image steganography using AES and DCT,” *J. Info. Security*, vol. 10, pp. 117–129, 2019.
19. *A Lossless Data Hiding Technique Based on AES-DWT*, F. R. Castillo Soria et al., arXiv:1212.2669, 2012.
20. R. D. Shannon, “Cryptography and communication complexity,” *IEEE Trans. Inf. Theory*, 1949 (fundamental cryptography reference).
21. J. Fridrich, “Steganography in digital media: principles, algorithms, and applications,” *Cambridge Univ. Press*, 2009.
22. U. S. Cho et al., “Enhanced LSB image steganography with key hashing and AES encryption,” *IEEE Access*, 2023.
23. L. Chen and S. Zhao, “Image steganography using hash-based embedding and AES encryption,” *Elsevier J. Inf. Secur. Appl.*, vol. 62, 2021.
24. D. Singh et al., “A novel 3-D image encryption algorithm based on SHA-256 and chaos theory,” *Aerospace Science and Technology*, Elsevier, 2025.
25. Y. Yang et al., “Robust steganography in encrypted images,” *IEEE Trans. Circuits Syst. Video Technol.*, 2022.
26. R. Mittal and M. K. Singh, “Secure text steganography with AES cryptography,” *Springer J. Multimedia Tools Appl.*, 2023.
27. P. Singh, “Hash-based steganography techniques in secure communication,” *Elsevier Comput. Secur.*, 2024.
28. K. Patel and A. Meena, “Hybrid secure image steganography using AES and dynamic key generation,” *IEEE Int. Conf. Emerg. Trends Commun. Tech.*, 2016.
29. H. X. Nguyen et al., “Survey of cryptographic steganography approaches,” *Frontiers in Computer Science*, 2024.
30. A. B. Jafari et al., “Analysis of image steganography robustness with AES encryption and hash preprocessing,” *Springer J. Signal Process. Image Commun.*, 2023.