Building of Informatics, Technology and Science (BITS)
Volume 6, No 4, March 2025 Page: 2756−2765
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i4.7079

Nearest Neighbor Interpolation and AES Encryption for Enhanced Least Significant Bit (LSB) Steganography

Nenny Anggraini*, Luh Kesuma Wardhani, Muhammad Hudzaifah Assyahid, Nashrul Hakiem, Muhammad Yusuf, Okky Bagus Setyawan

Faculty of Science and Technology, Informatics Engineering, UIN Syarif Hidayatullah, Jakarta, Indonesia
Email: [1,*]nenny.anggraini@uinjkt.ac.id, [2]luhkesuma@uinjkt.ac.id, [3]muhammad.hudzaifah19@mhs.uinjkt.ac.id, [4]hakiem@uinjkt.ac.id, [5]m.yusuf@uinjkt.ac.id, [6]okky.bagus@uinjkt.ac.id
Correspondence Author Email: nenny.anggraini@uinjkt.ac.id
Submitted: **01/03/2025**; Accepted: **26/03/2025**; Published**: 27/03/2025**

Abstract−The increasing use of digital communication raises concerns about data security, especially when transmitting sensitive information. Steganography conceals messages within digital media to prevent detection. However, conventional methods face storage limitations, leading to message truncation or distortion, making hidden messages more detectable. This study proposes a combination of Nearest Neighbor Interpolation (NNI) and Least Significant Bit (LSB) steganography to dynamically expand the cover image, allowing larger encrypted messages to be embedded while maintaining image integrity. NNI was chosen over other interpolation techniques such as Bilinear and Bicubic due to its lower computational complexity and preservation of sharp edges, which minimizes blurring artifacts that could make steganographic alterations more noticeable. AES-128 encryption ensures message confidentiality before embedding. The system was developed as a web-based application to improve usability. The research followed the Waterfall Software Development Life Cycle (SDLC), and Black Box Testing validated system functionality. Testing results showed that the method successfully embedded and extracted messages without data loss, maintaining PSNR values above 40 dB, ensuring minimal perceptual distortion. However, the maximum interpolation limit was 5310 × 5310 pixels, beyond which system constraints caused failures. The stego-images retained original aspect ratios, reducing suspicion. Despite its success, the system remains vulnerable to modifications such as color changes, cropping, rotation, and compression, which can disrupt the message.

Keywords: AES; Steganography; LSB; Nearest Neighbor Interpolation; Encryption

1. INTRODUCTION

In everyday life, humans constantly seek and exchange information. Information can be obtained from various sources such as libraries and the internet [1], which serves as a vast repository of knowledge and communication medium [2]. With the advancement of technology, the internet has become a crucial tool that enables individuals and organizations to communicate and exchange information without geographical or time limitations [3]. According to Indonesia's Central Bureau of Statistics (BPS) in 2021, internet users have reached 62.10% of the total population of 272.68 million, highlighting the widespread adoption of internet-based communication. However, the extensive transmission of data through the internet raises concerns about data security. Information sent over public networks is vulnerable to interception by unauthorized parties [4]. Data security is essential in digital communication, ensuring that data remains Confidentiality, Integrity, and Availability, commonly known as the CIA Triad [5]. To enhance data security, encryption techniques are often used to transform plaintext messages into unreadable ciphertexts that can only be decrypted by authorized recipients [6].

Data in digital form exists in various types, including text, audio, and images, all of which are susceptible to modifications. A digital image consists of a matrix of pixels, each represented by numerical values that define colors and intensity levels [7]. Image processing techniques allow for transformations such as color adjustments, resizing, and compression, which can be applied efficiently in real-time [1]. One common transformation is interpolation, a mathematical technique used to estimate new pixel values between existing ones, thereby increasing image resolution while preserving overall structure. Nearest Neighbor Interpolation (NNI) is a simple and computationally efficient method for enlarging digital images [8]. Unlike Bilinear or Bicubic Interpolation, which use weighted averages of neighboring pixels, NNI directly copies the nearest pixel value, making it faster but potentially introducing visual artifacts. Despite its simplicity, NNI has practical applications, particularly when speed is prioritized over smoothness.

Digital images can be used to protect sensitive information by embedding messages within them, a technique known as steganography [9]. In the digital era, steganography can be applied to various media types, including text, images, audio, frequencies, and video [10]. Steganography aims to conceal secret messages within a cover medium while ensuring that the hidden information can be retrieved without alteration or loss [10]. The effectiveness of steganography depends on key factors such as fidelity (how much the cover image is preserved) and robustness (how resistant the hidden message is to distortions or attacks).

One of the most commonly used methods in image steganography is the Least Significant Bit (LSB) method. This technique embeds secret messages in the least significant bits of an image's pixel values, ensuring minimal visual distortion [11]. Since LSB only modifies the last bits of pixel values, the changes are imperceptible to the human eye, making it an effective and widely adopted approach [10]. Additionally, its low computational complexity allows for fast processing, making it a preferred choice in many applications.

Steganography is often combined with cryptographic encryption to enhance security further. Encryption ensures that even if the hidden message is extracted, it remains unreadable without the correct decryption key [12].

Building of Informatics, Technology and Science (BITS)
Volume 6, No 4, March 2025 Page: 2756−2765
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i4.7079

Cryptography has been used since ancient times, with notable contributions from Islamic scholar Muḥammad bin Mūsā al-Khawārizmī. Modern cryptographic techniques are categorized into symmetric and asymmetric encryption [12]. Symmetric encryption uses a single key for both encryption and decryption, whereas asymmetric encryption employs a public key for encryption and a private key for decryption. Each approach has its advantages: symmetric encryption is faster, but it requires secure key exchange, whereas asymmetric encryption provides better security but requires more computational resources[13], [14]. One of the most widely used modern encryption standards is the Advanced Encryption Standard (AES), which was established in 2000 as a replacement for the outdated Data Encryption Standard (DES). AES offers three key-length variations: 128-bit, 192-bit, and 256-bit encryption. With current computing power, brute-force attacks on AES-128 encryption would take approximately 5.4×10^{18} years, making it a highly secure encryption standard [15]. In this study, AES encryption is combined with LSB steganography to enhance security, ensuring that hidden messages are both well-concealed and protected from unauthorized access. Unlike traditional LSB-based steganography, this approach integrates Nearest Neighbor Interpolation to dynamically adjust the cover image size when the message exceeds the available pixel capacity.

Several studies have explored the implementation of Least Significant Bit (LSB) steganography in different contexts. Nur'aini (2019) investigated LSB steganography using BMP image formats, highlighting that color degradation depends on the number of pixel bits altered by the embedded message [9]. However, the study only utilized static cover images, and the embedded messages were shorter than the available storage capacity.

Hotlan Sitorus et al. (2020) compared LSB+3 with Most Significant Bit (MSB) steganography and found that LSB+3 provided better results in terms of storage capacity, degradation, and processing speed [16]. Similarly, Yudistira Muria Muh et al. (2019) compared 1-bit and 2-bit LSB embedding. The 2-bit method allowed for larger message storage but resulted in higher PSNR values and longer processing times. These studies were also limited to static images, meaning they did not address cases where message sizes exceeded the available pixel capacity.

To improve security, Herteno et al. (n.d.) incorporated RSA-CRT encryption into LSB steganography on Android platforms [12]. While this approach enhanced security without significantly impacting computation speed, it still relied on static image containers. Similarly, Sofian et al. (2019) proposed a different approach by applying LSB steganography and AES-128 encryption in multiple PDF documents instead of images. This method increases storage capacity while maintaining security but is constrained by the characteristics of PDF files.[17]

A more dynamic approach was proposed by Harahap & Khairina (2020), who introduced Nearest Neighbor Interpolation (NNI) to expand the cover image when the embedded message exceeded available pixel space [18]. This method successfully prevented message truncation, ensuring that the hidden data remained intact. However, the approach caused excessive horizontal stretching, making the modified image visibly distorted and suspicious.

Despite various advancements in LSB-based steganography, significant challenges remain, particularly in storage capacity, image distortion, and security. Traditional methods rely on static cover images, restricting the amount of data that can be embedded and leading to message truncation when available pixel space is insufficient. While Nearest Neighbor Interpolation (NNI) has been introduced to dynamically expand images and accommodate longer messages, existing implementations often cause noticeable distortions, particularly horizontal stretching, making the modifications visually detectable. Additionally, while encryption techniques such as AES and RSA-CRT improve security, most implementations do not integrate encryption seamlessly with dynamic image expansion, leaving hidden messages vulnerable to extraction attacks.

To address these issues, this study proposes a hybrid approach that combines LSB steganography, AES encryption, and bidirectional NNI-based image interpolation. Unlike previous methods that result in unbalanced image scaling, this approach ensures proportional expansion in both horizontal and vertical dimensions, preserving the visual integrity of the cover image while maximizing message capacity and security. Furthermore, implementing this method in a web-based platform enhances usability, accessibility, and multi-platform support, making it a more practical and scalable solution for secure digital communication.

2. RESEARCH METHODOLOGY

We employs the Software Development Life Cycle (SDLC) Waterfall method in the implementation of this JavaScript-based program. The Waterfall approach was chosen because it provides a clear and structured definition for each stage and ensures proper control over the development process [19]. Unlike iterative methods, each phase in Waterfall does not necessarily need to be fully completed before transitioning to the next phase, allowing for a more organized and systematic development flow [20]. The research flow is shown in Figure 1.

2.1. Requirement Analysis

In this initial stage, the identification, understanding, and documentation of business and functional requirements for the system to be developed are conducted. Interaction with stakeholders is also carried out to understand the goals, features, and specifications that the system must meet.

a. Literature Review
   To define the system's requirements, a literature review was conducted on existing research related to steganography. This process involved analyzing previous studies, published research papers, and scientific articles

Building of Informatics, Technology and Science (BITS)
Volume 6, No 4, March 2025 Page: 2756−2765
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i4.7079

that discuss LSB-based steganography, encryption techniques, and image interpolation methods. The objective was to identify common challenges, limitations, and potential improvements that could be applied to this study.

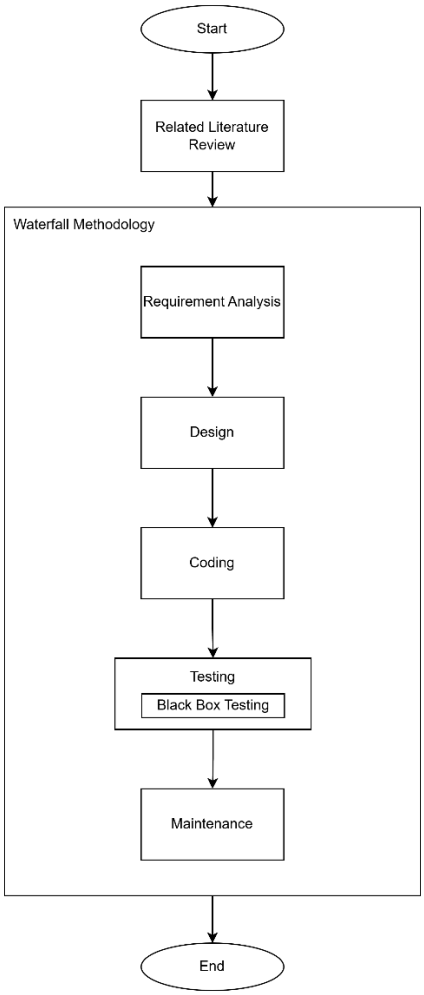

Figure 1. Research Stage

b. Observational Study
In addition to literature research, an observational study was conducted by examining and experimenting with various steganography implementations available on the internet, including GitHub repositories and Paper with Code implementations. The assessment focused on key aspects such as the type of algorithm used, its ability to handle different message sizes, the accuracy of message extraction, the impact on image quality, and the ease of implementation in a web-based system. These evaluations guided the selection of the most suitable algorithm for this research.

c. System User Analysis
Based on findings from the literature review and observational study, the target users of the system were identified. The intended users are individuals or organizations who require secure and undetectable secret text transmission using dynamic image containers. This approach ensures that messages remain concealed without raising suspicion, providing an additional layer of security and privacy for confidential communication.

2.2. Design

In this stage, various system designs are created for development. This includes system architecture design, software design, database design, and user interface design. The goal is to provide a detailed guideline on how the system will be built. The proposed system performs encoding, encryption, interpolation, and steganography to embed secret messages into images while ensuring security and adaptability to different file sizes. The system is divided into two main processes, Steganography (Embedding) and Extraction (Decoding) as shown in Figure 2.

Building of Informatics, Technology and Science (BITS)
Volume 6, No 4, March 2025 Page: 2756−2765
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i4.7079

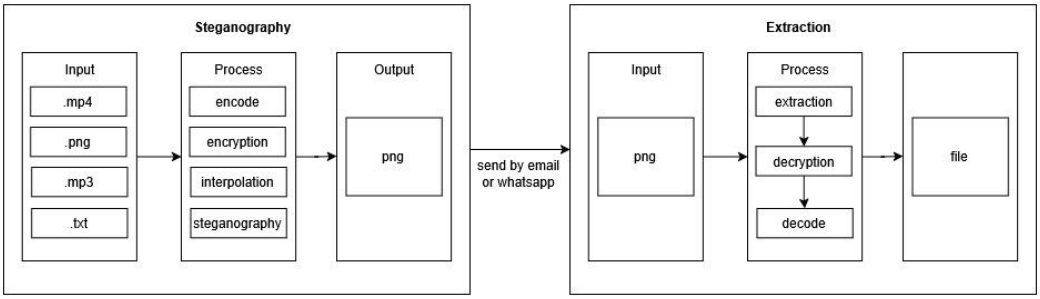

Figure 2. Steganography and Extraction Framework

2.2.1. Steganography Process

The steganography process consists of input, processing, and output stages, as follows:

a. Input Stage
The system accepts various input file types that contain the secret message, including .mp4 (Video), .png (Image), .mp3 (Audio), .txt (Text).

b. Processing Stage
The steganography process involves multiple steps to securely encode and embed the secret message within an image. Initially, the uploaded file is transformed into a Base64 string, which enables it to be handled as text data suitable for embedding. Following this, the Base64 string undergoes encryption using the AES-128 algorithm to safeguard the message from unauthorized access. In cases where the encrypted string surpasses the pixel capacity of the chosen image, the system employs Nearest Neighbor Interpolation to dynamically enlarge the image, ensuring that the entire message can be embedded without any truncation or data loss. Once it is verified that the image has sufficient capacity, the encrypted text is then embedded into the Least Significant Bits (LSB) of the image pixels, completing the steganography process.

c. Output Stage
The final output is a PNG image containing the hidden encrypted message. This stegaimage (steganography image) is now ready to be shared via email, WhatsApp, or other social media platforms without raising suspicion.

2.2.2. Extraction Process

The extraction process consists of input, processing, and output stages to retrieve the original secret file from the steganography image.

a. Input Stage
The system accepts a PNG file as input, which contains the hidden message embedded during the steganography process.

b. Processing Stage
During the processing stage, the hidden message is retrieved from the image through a series of steps. First, the system extracts the Least Significant Bits (LSB) from the image pixels to reconstruct the encrypted message embedded within the image. Once the encrypted data is successfully extracted, it undergoes decryption using the AES-128 algorithm, converting it back into its encoded Base64 form. Finally, the Base64 string is decoded to restore the original file—whether it be in .mp4, .mp3, .png, or .txt format—exactly as it was before the embedding process.

c. Output Stage
The final output is the original file, successfully extracted and restored without modification or corruption.

2.3. Coding

Once the system design is completed, the coding phase begins. The development team uses the relevant programming languages to write software code based on the prepared design. Coding is carried out according to established standards and guidelines.

2.4. Testing

After the software coding is completed, the testing phase is conducted to ensure that the system functions according to the defined requirements. This stage includes unit testing, integration testing, system testing, and acceptance testing. The primary goal is to detect and fix errors or defects in the system before deployment.

2.5. Maintenance

This phase involves monitoring, maintaining, and improving the developed system. Maintenance may include bug fixes, performance enhancements, adjustments to changing requirements, and the addition of new features if necessary.

Building of Informatics, Technology and Science (BITS)
Volume 6, No 4, March 2025 Page: 2756−2765
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i4.7079

3. RESULT AND DISCUSSION

3.1 Result

3.1.1 Interface Implementation

The developed system was implemented as a web-based application, allowing users to securely embed and extract encrypted messages using Nearest Neighbor Interpolation (NNI) and Least Significant Bit (LSB) steganography. The interface is designed to be user-friendly and intuitive, ensuring ease of access for users with minimal technical knowledge.

The application provides several key functionalities, including message input, encryption, interpolation, steganographic embedding, extraction, and decryption. Users can upload a text file, image, audio, or video as the message container. If the message size exceeds the available pixel capacity, the system dynamically expands the image using NNI interpolation to accommodate the encrypted message. Upon successful embedding, users can download the stego-image and later extract the hidden message using the decryption module.

Below in the Figures 3, 4, 5, 6, and 7 are screenshots of the system interface, showcasing each step of the process, including file upload, encryption, image interpolation, message embedding, and message extraction. The interface layout is designed for efficient navigation, ensuring that each function is easily accessible.

a. Encryption Process
Figure 3 below illustrates the encryption process interface, where users can upload a file and input a secret key before initiating the encryption. The interface clearly outlines the supported message formats and provides an option to begin the encryption process with a single click.

Figure 3. Encryption process interface

b. Steganography Process
Figure 4 below illustrates the steganography process interface, where users can select an image file in PNG or JPG format to embed the encrypted message. The interface provides a file selection option and a preview area to display the chosen image before proceeding with the embedding process. This step ensures that users can verify the carrier image before the message is hidden within it.

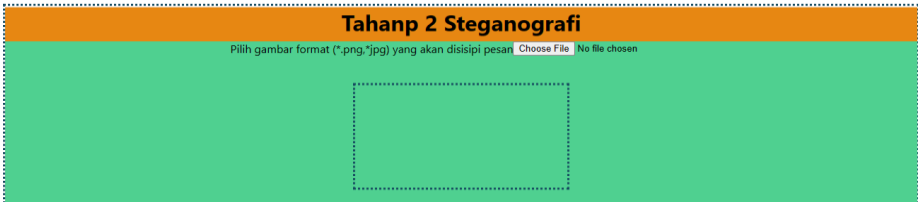

Figure 4. Steganography process interface

c. Interpolation Process
Figure 5 below illustrates the interpolation process interface, where the system applies Nearest Neighbor Interpolation (NNI) to expand the image. This step is necessary when the message size exceeds the available pixel capacity, allowing the image to accommodate the encrypted data. The interface includes a preview area to display the interpolated image, ensuring users can verify the changes before proceeding to the next stage.

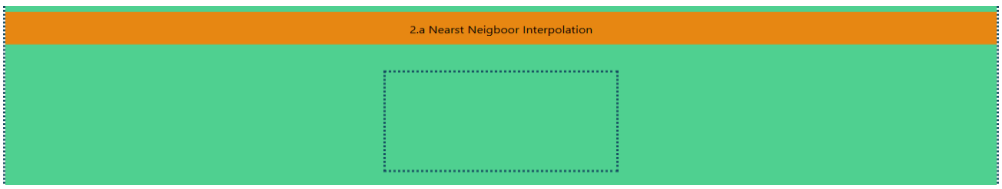

Figure 5. Interpolation process interface

Building of Informatics, Technology and Science (BITS)
Volume 6, No 4, March 2025 Page: 2756−2765
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i4.7079

d. Message Embedding Process
Figure 6 illustrates the message embedding process interface, where the encrypted message is inserted into the interpolated image using the Least Significant Bit (LSB) steganography technique. The interface provides a preview of the modified image along with the calculated Peak Signal-to-Noise Ratio (PSNR) value, which helps assess the visual quality after embedding. This ensures that the hidden message remains imperceptible while maintaining image integrity.

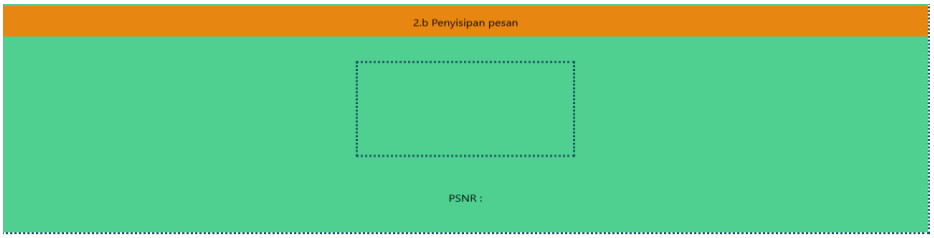

Figure 6. Message embedding process interface

e. Decryption Process
Figure 7 illustrates the decryption process interface, where users can upload the steganographic image containing the hidden message. The interface provides an input field for the secret key, ensuring that only authorized users can extract the embedded message. Once the correct key is entered, the decryption process reveals the hidden text, allowing users to retrieve the original message securely.

Figure 7. Decryption process interface

3.1.2 Black Box Testing Result

In this study, Black Box Testing was used to evaluate whether the Encoding, Encryption, Interpolation, and Steganography functions operate correctly according to their expected behavior. This method focuses on functional testing without examining the internal code structure, ensuring that each function performs as intended from a user perspective. The testing process involved executing various test cases for each function, observing their outputs, and verifying if they met the expected results. The results of the Black Box Testing are summarized in Table 1 below.

Table 1. Black Box Testing Result

No	Number	Expected Result	Testing Result	Conclusion
1	Input the key and image container, then trigger the encryption process	The Base64 text is converted into ciphertext while remaining intact	Ciphertext successfully obtained in its entirety	Valid
2	Once the image container is available, the program performs interpolation on the image if the encrypted text length exceeds the number of pixels in the container	The container is successfully interpolated on both the X and Y axes	The container is successfully interpolated on both the X and Y axes	Valid
3	After ensuring the container is sufficient, embed the Base64 text into the interpolated image	The entire message is successfully embedded into the image without raising suspicion	The entire message is successfully embedded into the image without raising suspicion	Valid
4	When the image containing the embedded message is uploaded to the message extraction element, the message should be retrieved and decrypted	The message is retrieved and decrypted completely and accurately	The message is retrieved and decrypted completely and accurately	Valid

Building of Informatics, Technology and Science (BITS)
Volume 6, No 4, March 2025 Page: 2756−2765
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i4.7079

3.1.3 Interpolation Result

Image interpolation is applied in this study to expand the cover image when the encrypted message size exceeds the available pixel capacity. Nearest Neighbor Interpolation (NNI) is used in both the X and Y axes, ensuring that the message can fit within the image without truncation. To test the effectiveness of this approach, experiments were conducted with different image dimensions, message sizes, and scaling factors. The results of these tests are shown in Table 2 below.

Table 2. Interpolation result details

Original Dimension (px)	Message Length (bit)	Scaling Factor (x)	Before Interpolation	After Interpolation	Missing Bits for Message Before Interpolation (bit)	Alpha Channel After Interpolation (bit)	Total RGBA After Interpolation	Usable RGB/Wadah After Interpolation (bit)	Unused RGB/Wadah (bit)	Final Dimension (px)	Result
10x10	212,232	531x	400	300	112,784,400	84,588,300	211,932	28,196,100	112,784,000	84,588,000	84,376,068
10x10	212,744	532x	400	300	113,209,600	84,907,200	212,444	28,302,400	113,209,200	84,906,900	84,694,456
10x10	214,952	538x	400	300	115,777,600	86,833,200	214,652	28,944,400	115,777,200	86,832,900	86,618,248
1x1	584	146x	4	3	85,264	63,948	581	21,316	85,260	63,945	63,364
1x1	211,208	5266x	4	3	110,923,024	83,952,300	211,205	26,970,724	110,923,020	83,952,297	83,741,092
1x1	21,256	5314x	4	1	112,954,384	84,715,788	21,255	28,238,596	112,954,380	84,715,787	84,694,532
1x1	83,528	2088 2x	4	1	1,744,231,696	1,308,173,772	83,527	436,057,924	1,744,231,692	1,308,173,771	1,308,090,244

3.1.4 Encryption Result

The encryption process in this study is used to secure the message before embedding it into an image. AES-128 encryption is applied after the message is Base64-encoded, ensuring that the content remains protected. To evaluate the performance of encryption, several tests were conducted with different file sizes and formats. The results are shown in Table 3 below.

Table 3. Encryption result details

Format	File Size	Message Size (bits)	Cipher Text Size (bits)	Steganography Success	Extraction Success	Steganography Time	Extraction Time
.mp4	5.6 MB	7,843,380	10,457,880	Sometimes successful, sometimes not	Sometimes successful, sometimes fails (memory exhaustion)	More than 1 minute	More than 1 minute
.mp4	4.49 MB	6,274,004	8,365,376	Successful but slow	Successful but very slow	More than 1 minute	59 seconds
.mp4	1.85 MB	2,594,336	3,459,160	Successful but sometimes slow	Successful	31 seconds	12 seconds
.mp4	1.12 MB	1,568,840	2,091,820	Stable success	Successful	16 seconds	11 seconds
.mp4	740 KB	1,010,364	1,347,180	Stable success	Successful	9 seconds	7 seconds
.mp3	1.77 MB	2,476,348	3,301,824	Stable success	Successful	26 seconds	18 seconds
.mp3	775 KB	1,058,364	1,411,180	Stable success	Successful	7 seconds	7 seconds
.mp3	387 KB	529,212	705,644	Stable success	Successful	4 seconds	3 seconds
.txt	13.4 KB	18,424	24,600	Stable success	Successful	2 seconds	2 seconds

Building of Informatics, Technology and Science (BITS)
Volume 6, No 4, March 2025 Page: 2756−2765
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i4.7079

3.1.5 Steganography Result

Steganography is the process of embedding a hidden message into a cover image while ensuring that modifications remain undetectable. In this study, the Least Significant Bit (LSB) method was used to embed AES-128 encrypted messages into images. When the message size exceeded the pixel capacity of the original image, Nearest Neighbor Interpolation (NNI) was applied to increase the image dimensions, allowing for the full embedding of the message without truncation. To evaluate the performance of the steganographic embedding and interpolation process, several tests were conducted with different image sizes, message sizes, and scaling factors. The test results, including PSNR values, pixel utilization, and percentage of empty pixels, are presented in Table 4 below.

Table 4. Steganography result details

Sample	Initial Size (px)	Final Size (px)	Initial Size (KB)	Message Size (KB)	Final Size (KB)	Scaling Factor (x)	PSNR (dB)	Original Data in Container (%)	Size Increase (%)	Total Pixels	Empty Pixels
1	150x150	300x300	9.3	13.4	139	2	47.7	9.64	1394.62	90,000	67,500
2	85x134	1105x1742	32.8	33.8	2030	13	46.47	1.67	6089.02	1,924,910	1,924,910
3	364x368	1456x1472	48.7	370	2500	4	47.14	14.8	5033.47	2,143,232	2,009,280
4	186x171	976x984	8.44	13.4	9.58	8	47.12	139.87	13.51	960,384	928,578

3.1.6 Attack Testing Result

To evaluate the robustness of the steganographic method, several attack scenarios were conducted to assess whether the hidden message remains intact and decryptable after modifications to the cover image. These attacks include visual alterations such as color changes, rotation, cropping, and compression. The results of the attack testing are presented in Table 5 below.

Table 5. Attack testing result details

Sample	Attack Type	Cipher Text Found	Decryption Status
Sample 1	Visual Color Change	Not Found	Failed
Sample 2	Visual Rotate	Not Found	Failed
Sample 3	Visual Crop	Found	Failed
Sample 4	Compression	Not Found	Failed

3.2 Discussion

The Black Box Testing results confirm that all system functions operated correctly. The encryption, interpolation, and steganography processes were successfully executed, allowing messages to be securely embedded and extracted without errors. These results indicate that the system operates reliably under normal conditions, ensuring the integrity of the hidden messages during transmission.

Regarding interpolation, the test results show that the maximum scaling limit is 5310 x 5310 pixels, allowing up to 10.57 MB of messages to be embedded within the image. Beyond this limit, out-of-range errors occur, making further interpolation impractical. This highlights the trade-off between storage capacity and system performance, emphasizing the need for efficient message embedding without excessive scaling, which could negatively impact system efficiency.

In terms of encryption performance, it was observed that encrypting and interpolating files larger than 1 MB significantly increases processing time, occasionally causing the program to slow down or become unresponsive. However, when the message file size is below 1 MB, the encryption and interpolation processes run stably, without significant delays. This suggests that message size has a direct impact on system performance, indicating a need for optimization when handling larger encrypted messages to maintain usability.

For steganography, the results indicate that LSB-based embedding combined with interpolation preserves image quality, as demonstrated by PSNR values consistently above 40 dB. A PSNR value above this threshold suggests that the modifications introduced during the embedding process remain visually imperceptible to the human eye. However, message embedding efficiency varies depending on the scaling factor and original image size. Some test cases showed significant unused pixel space, with empty pixel percentages ranging from 75% to 100%, reinforcing the importance of optimizing the interpolation factor to achieve an effective balance between storage efficiency and steganographic security.

The attack testing results indicate that LSB-based steganography is highly sensitive to image modifications. In the color change and rotation attacks, the hidden message was completely destroyed, making extraction impossible.

Building of Informatics, Technology and Science (BITS)
Volume 6, No 4, March 2025 Page: 2756−2765
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i4.7079

Similarly, compression eliminated the embedded message entirely, preventing retrieval. In contrast, cropping allowed partial extraction, but the retrieved ciphertext was incomplete, leading to decryption failure. These findings suggest that even minor pixel alterations disrupt LSB steganography, making it unsuitable for scenarios where image modifications may occur.

Overall, the results suggest that while LSB steganography with AES encryption and interpolation is effective for securely embedding messages, it remains vulnerable to certain types of attacks and excessive scaling inefficiencies. Future improvements may focus on error correction techniques, alternative steganographic methods, or adaptive interpolation strategies to enhance storage efficiency and robustness against visual modifications.

4. CONCLUSION

The study successfully applied Nearest Neighbor Interpolation steganography and Most Significant Bits (LSBs) to embed encrypted messages into interpolated images while ensuring data integrity, achieving full embedding and extraction without errors. The web-based system enhances usability, with Nearest Neighbor Interpolation effectively expanding images to accommodate larger messages while maintaining a Peak Signal-to-Noise Ratio (PSNR) above 40 dB for minimal perceptual distortion. However, testing revealed a maximum interpolation limit of 5310 x 5310 pixels, exceeding memory constraints and causing failures. While maintaining the original aspect ratio reduces detection risks, the method remains vulnerable to modifications such as color changes, rotation, cropping, and compression, which may disrupt message retrieval. uture work could explore advanced interpolation methods like Bicubic Interpolation or Deep Learning-based High-Resolution Models and implement error correction techniques, such as Reed-Solomon codes for error resilience or convolutional neural networks (CNN) for detecting and recovering corrupted message bits, to enhance robustness against image modifications.

REFERENCES

[1] N. Anggraini, S. H. Ramadhani, L. K. Wardhani, N. Hakiem, I. M. Shofi, and M. T. Rosyadi, "Development of Face Mask Detection using SSDLite MobilenetV3 Small on Raspberry Pi 4," in *2022 5th International Conference of Computer and Informatics Engineering (IC2IE)*, IEEE, Sep. 2022, pp. 209–214. doi: 10.1109/IC2IE56416.2022.9970078.

[2] E. P. Setiawan and I. Ismurjanti, "Penggunaan Internet sebagai sumber informasi dalam penyusunan karya ilmiah Siswa SMA Negeri 8 Yogyakarta," *Jurnal Kajian Informasi dan Perpustakaan*, vol. 6, no. 2, Dec. 2018, doi: 10.24198/jkip.v6i2.18590.

[3] R. Irawan, D. Wijaya, I. Prana, and I. K. Dewi, "Pelatihan Internet Sebagai Media Informasi dan Komunikasi Untuk Santri Pada Pondok Pesantren Daarul Hasanah Bogor," *Abditeknika Jurnal Pengabdian Masyarakat*, vol. 1, no. 2, pp. 113–119, Oct. 2021, doi: 10.31294/abditeknika.v1i2.633.

[4] I. Darmayanti, D. N. Astrida, and D. Ariyus, "Penerapan Keamanan Pesan Teks Menggunakan Modifikasi Algoritma Caisar Chiper Kedalam Bentuk Sandi Morse," *Jurnal Ilmiah IT CIDA*, vol. 4, no. 1, Jul. 2019, doi: 10.55635/jic.v4i1.78.

[5] R. N. Fuad, "PERANCANGAN APLIKASI PENYISIPAN PESAN MELALUI METODE MIXING (PARITY CODING DAN ENKRIPSI ADVANCED ENCRYPTION STANDARD/AES) PADA FILE MP3," *Jurnal TIMES*, vol. 10, no. 2, pp. 12–21, Jan. 2022, doi: 10.51351/jtm.10.2.2021653.

[6] E. R. Sardju, R. Magdalena, and R. D. Atmaja, "Implementasi Algoritma Rsa Untuk Enkripsi Dan Dekripsi Sms (short Message Service) Pada Ponsel Berbasis Android," *eProceedings of Engineering*, vol. 2, no. 2, 2015, doi: 10.22441/fifo.v9i2.2566.

[7] S. Ratna, "PENGOLAHAN CITRA DIGITAL DAN HISTOGRAM DENGAN PHYTON DAN TEXT EDITOR PHYCHARM," *Technologia: Jurnal Ilmiah*, vol. 11, no. 3, p. 181, Jul. 2020, doi: 10.31602/tji.v11i3.3294.

[8] B. Hardiansyah, A. P. Armin, and A. B. Yunanda, "REKONSTRUKSI CITRA PADA SUPER RESOLUSI MENGGUNAKAN INTERPOLASI BICUBIC," *INTEGER: Journal of Information Technology*, vol. 4, no. 2, Oct. 2019, doi: 10.31284/j.integer.2019.v4i2.684.

[9] S. Nur'aini, "Steganografi Pada Digital Image Menggunakan Metode Least Significant Bit Insertion," *Walisongo Journal of Information Technology*, vol. 1, no. 1, p. 73, Nov. 2019, doi: 10.21580/wjit.2019.1.1.4025.

[10] I. G. N. B. Pramana Putra, I. K. G. Suhartana, I. K. A. Mogi, C. R. A. Pramartha, I. P. G. H. Suputra, and I. G. A. Wibawa, "Penerapan Steganography Untuk Perlindungan Hak Cipta Menggunakan Metode Least Significant Bit (LSB)," *JELIKU (Jurnal Elektronik Ilmu Komputer Udayana)*, vol. 10, no. 4, p. 329, Jun. 2022, doi: 10.24843/JLK.2022.v10.i04.p03.

[11] S. Lutfi and R. Rosihan, "PERBANDINGAN METODE STEGANOGRAFI LSB (LEAST SIGNIFICANT BIT) DAN MSB (MOST SIGNIFICANT BIT) UNTUK MENYEMBUNYIKAN INFORMASI RAHASIA KEDALAM CITRA DIGITAL," *JIKO (Jurnal Informatika dan Komputer)*, vol. 1, no. 1, pp. 34–42, Apr. 2018, doi: 10.33387/jiko.v1i1.1169.

[12] R. Herteno, W. Ramadansyah, O. Soesanto, R. A. Nugroho, and A. Rusadi, "Steganografi Untuk Pesan Terenkripsi Menggunakan Algoritma Kriptografi Rsa-Crt Di Android," *KLIK-KUMPULAN JURNAL ILMU KOMPUTER*, vol. 6, no. 1, pp. 16–26, 2019, doi: dx.doi.org/10.20527/klik.v6i1.246.

[13] M. N. Alenezi, H. Alabdulrazzaq, and N. Q. Mohammad, "Symmetric encryption algorithms: Review and evaluation study," *International Journal of Communication Networks and Information Security*, vol. 12, no. 2, pp. 256–272, 2020, doi: doi.org/10.17762/ijcnis.v12i2.4698.

[14] C.-L. Chen, Z.-Y. Lim, X. Xue, and C.-H. Chen, "Special Issue: Symmetric and Asymmetric Encryption in Blockchain," *Symmetry (Basel)*, vol. 15, no. 2, p. 458, Feb. 2023, doi: 10.3390/sym15020458.

[15] T. D. A. P. Wardhani and Y. Asriningtias, "Implementasi Algoritma AES-256 Dalam Perancangan Aplikasi Pengamanan Dokumen Digital Perusahaan Berbasis Android," *INTECOMS: Journal of Information Technology and Computer Science*, vol. 6, no. 2, pp. 1289–1293, Jan. 2024, doi: 10.31539/intecoms.v6i2.8027.

Building of Informatics, Technology and Science (BITS)
Volume 6, No 4, March 2025 Page: 2756−2765
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v6i4.7079

[16] Uray Ristian, Jatrya, and Sampe Hotlan Sitorus, "PERBANDINGAN STEGANOGRAFI METODE LEAST SIGNIFICANT BIT+3 (LSB+3) DENGAN MOST SIGNIFICANT BIT (MSB)," *Coding Jurnal Komputer dan Aplikasi*, vol. 8, no. 1, Jan. 2020, doi: 10.26418/coding.v8i1.39195.

[17] N. Sofian, A. Wicaksana, and S. Hansun, "LSB Steganography and AES Encryption for Multiple PDF Documents," in *2019 5th International Conference on New Media Studies (CONMEDIA)*, IEEE, Oct. 2019, pp. 100–105. doi: 10.1109/CONMEDIA46929.2019.8981842.

[18] M. K. Harahap and N. Khairina, "Dynamic Steganography Least Significant Bit with Stretch on Pixels Neighborhood," *Journal of Information Systems Engineering and Business Intelligence*, vol. 6, no. 2, p. 151, Oct. 2020, doi: 10.20473/jisebi.6.2.151-158.

[19] S. Pargaonkar, "A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering," *International Journal of Scientific and Research Publications*, vol. 13, no. 8, pp. 120–124, Aug. 2023, doi: 10.29322/IJSRP.13.08.2023.p14015.

[20] R. Mokhtar and M. Khayyat, "A Comparative Case Study of Waterfall and Agile Management," *SAR Journal - Science and Research*, pp. 52–62, Mar. 2022, doi: 10.18421/SAR51-07.