Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 29 (2026), No. 6, pp. 2349–2363
DOI : 10.47974/JDMSC-2610

A robust data hiding scheme using modular key mapping and AES encryption

Naman Beri [§]
Department of Electronics Science
Acharya Narendra Dev College
University of Delhi
Delhi 110019
India

Vibha Gaur [†]
Department of Computer Science
Acharya Narendra Dev College
University of Delhi
Delhi 110019
India

Ravneet Kaur *
Department of Electronics Science
Acharya Narendra Dev College
University of Delhi
Delhi 110019
India

Abstract

In the contemporary digital environment, data security is of the utmost importance. Utilizing Advanced Encryption Standard (AES) encryption and modular arithmetic-based steganography, a novel method is developed that combines cryptographic protection with steganographic concealment. This dual-layer methodology addresses both visibility and cryptanalysis challenges in secure communication. This robust approach combines the strengths of both AES encryption and text steganography, making it highly challenging for unauthorized parties to detect or decipher the concealed information. A stego-key is generated by utilizing a mapping function based on modular arithmetic and modulo

[§] *ORCID-ID:* **0009-0000-3775-7195**
[†] *ORCID-ID:* **0000-0001-6668-9339**
* *ORCID-ID:* **0000-0001-8186-0982**

[§] *E-mail:* **naman.230423@andc.du.ac.in**
[†] *E-mail:* **vibhagaur@andc.du.ac.in**
* *E-mail:* **ravneetkaur@andc.du.ac.in (Corresponding Author)**

© Taru Publications

operations. The cover text remains unaltered, with no discernible alterations or additions. The stego-key is employed to embed the concealed message within the cover text, and an automated algorithm is employed to perform encryption. Using the inverse encryption algorithm and the generated stego-key, the confidential message can be retrieved by the recipient. This method employs a customized implementation of the pseudo-random number generator and basic file manipulation techniques. The Tkinter module in Python was also implemented to develop a user-friendly interface for the proposed text-based encryption and decryption. The proposed method provides an efficient and practical secure data communication solution, maintaining the integrity of the cover text, accomplishing faster processing speeds, and minimizing memory usage.

Subject Classification: *94A60.*

Keywords: *Steganography, Cryptography, Steganalysis, Text-steganography, Mapping function, Sender, Receiver.*

1 Introduction

The expansion of internet connectivity has propelled the need for robust data protection to unprecedented levels. In a variety of sectors, such as government, healthcare, legal, and financial institutions, text steganography has emerged as a critical defense mechanism, enabling the concealment of sensitive communications within innocuous text.

Steganography refers to the practice of concealing secret information such that only the intended sender and recipient can interpret it. The challenge is to include significant data without eliciting suspicion. Cryptography and steganography, when integrated synergistically, constitute a robust barrier against illegal access, thereby enhancing the privacy and security of data transmitted across communication networks. Cryptography converts data into an unreadable format to safeguard its integrity, and steganography employs a more discreet method by hiding information in seemingly innocuous content. This deliberate amalgamation employs numerous methods to protect sensitive information in our interconnected environment [1].

Modern methods, including homomorphic cryptography and least significant bit (LSB) embedding, improve security by effectively concealing data and complicating detection [2]. Private journaling, anonymous mailing systems, and secure data transmission via web pages or social media are among the practical applications that employ steganography in conjunction with online platforms, such as social networks, to enable clandestine and confidential communication [3]. In the healthcare sector, steganography integrated with image encryption is used to protect sensitive patient information within medical images with ensured privacy regulations [4].

Sani and Nujjaid [5] integrate cryptographic algorithms with steganography to guarantee secured transmission of confidential information. To enhance the security and efficiency of data transmission in the Internet of Things (IoT) environment, encryption and a deep learning-based steganography method are employed. [6]. A novel method is presented [7] that embeds secret data within cover text using invisible Unicode characters, combining multilayer encoding

with format-preserving encryption and Huffman coding. These studies underscore the potential of integrating encryption with text steganography to develop communication systems that are both imperceptible and highly secure.

In the present work, a modular arithmetic-based steganography method is introduced that combines the Advanced Encryption Standard (AES) [8] with modulo mathematical operations [9] to protect sensitive data in a variety of applications. The suggested method conceals massive amounts of data without significantly increasing file size by embedding AES-encrypted messages into pseudo-randomized cover text positions. As a result, confidentiality is ensured regardless of detection, even in the event that the ciphertext is extracted and the steganographic layer is compromised, since the message cannot be decrypted without the decryption key [10].

The multi-layered security model complicates unauthorized access, as assailants would need to comprehend both the steganographic method applied and the AES key in order to decrypt the message [10].

2 Related Work

Text steganography techniques can be categorized based on various criteria mentioned below.

2.1 *Embedding Techniques*

In this steganography technique, secret communications are encoded within the semantic or syntactic structure of text to obscure information within natural language. This necessitates sophisticated linguistic analysis to guarantee non-detection of concealed messages while preserving the text's coherence and intelligibility [7].

2.2 *Cover Sources*

In cover source dependent steganography, secret communications are embedded into cover sources according to their distinctive attributes. The alteration of a cover source's attributes thus requires domain-specific knowledge. The Natural Language Text Method, the Generated Text Method, and the Structured Text Method are examples of cover source dependent steganography [7].

2.3 *Linguistic Properties*

These steganography methods conceal information within text by utilizing qualities such as syntax, semantics, grammar, and lexicon. Common methods include Homographic Steganography, Synonym Substitution, and Grammar-Based Steganography [1].

2.4 *Detection Difficulty*

The detection difficulty-based steganography methods increase the complexity of detecting hidden information within cover objects. It exacerbates the challenges that adversaries face in distinguishing between legitimate cover objects and those that contain concealed data.

Sophisticated techniques are employed frequently utilizing complex algorithms and encoding schemes to improve security and reduce detection [2].

Innovative techniques are necessary to conceal substantial data without affecting the text's appearance. In order to prevent suspicion or alterations in meaning, it must maintain text coherence and resist detection by steganalysis tools. It is imperative to prioritize security in order to resist decryption attempts. Efficient key administration is essential, as is robustness across formats and platforms. Continuous research is necessary to improve the resilience, capacity, and efficacy of steganography in order to overcome these obstacles. The proposed method employs mathematically driven patterns to ensure the dispersed, non-linear placement of embedded secret data within a textual cover medium, thereby enhancing secrecy and security. The data's confidentiality is guaranteed through AES encryption.

3 Stego-Key Generation Secure Text Embedding using Modular Key-Based Steganography

The proposed method uses a pseudo-random generator with modular arithmetic and AES encryption to optimize data concealment while simultaneously ensuring minimal detectability and enhanced security. The approach directly utilizes the cover text without introducing additional information.

A mapping technique to determine the embedding positions within a digital cover medium—typically a text, is used. This approach uses mathematical operations to generate pseudo-random, non-linear embedding positions to generate pseudo-random numbers [11]. These indices are subsequently encrypted through the symmetric AES algorithm. The key and initialization vector (IV) are generated and transmitted with the cover text (the original text) in AES ciphertext. These elements are also used to reconstruct the original text. The encryption process is initiated by generating a random value, which is then used with the encryption key. The IV guarantees that the ciphertexts will be distinct, even if the same plaintext is encrypted multiple times with the same key. This is crucial for security purposes, as it prevents assailants from detecting patterns or relationships in the ciphertext.

The pseudo-random numbers are generated by the steganography algorithm using a linear equation shown in Equation (1). It is used to determine the positions in the cover medium where the secret message will be embedded. The recurrence relation is given as

$$X_{n+1} = (aX_n + b) \bmod (m) \tag{1}$$

Here X is the sequence of pseudo-random numbers such that X_n is the current seed value and X_{n+1} is the subsequent random value. ***a*** and ***b*** are constant multiplier and increment parameters, respectively. ***m*** is the number of occurrences of a given character to be mapped in the cover text, and ***n*** is the iteration index. The parameter ***a*** controls the rate of change between successive random values, influencing the sequence's periodicity and uniformity. A well-distributed sequence and an extended period before repetition are guaranteed by an optimal

selection of $\boldsymbol{a}$. The parameter b introduces a constant shift, which further randomizes the sequence.

The mapping function uses a two-step approach to generate random numbers. The first step utilises the mathematical mapping function given in Equation (2) to generate the random numbers.

$$x_{n+1} = (a * x_n + b) \tag{2}$$

Although deterministic, this method does not produce truly random numbers but instead generates values that mimic randomness. In order to guarantee that the random number generated correlates with a valid index within the cover text, the character's index is subsequently adjusted in the second step using the function given in Equation (3).

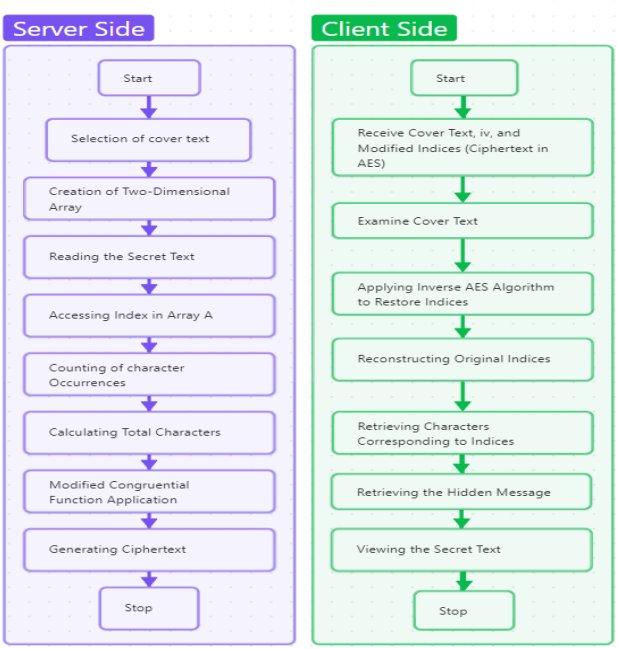
$$i = (X_{n+1}) mod\ (m). \tag{3}$$

Here, i represents the character's position in the cover text, the(X_{n+1}) is the pseudo-random value generated in the first step, and m denotes the total occurrences of the character being mapped. The value of m is modified to correspond with the least common character in the cover text. This modification is crucial as it guarantees that the indices produced by the two-step method stay within the permissible range, so averting out-of-bounds errors and preserving the integrity of the mapping process. The indices acquired during the second step are modified utilizing the AES method. The updated indices, obtained from the cover text and altered random values, are encrypted using AES to augment security. The AES method produces an IV and ciphertext, accompanied by a secret key, which are subsequently delivered securely to the recipient for decryption.

3.1 *Mapping and Encryption at the sender-Server Level*

At the sender side, i.e., at the server level, a 2D array A, i.e., A[row][column], with the number of rows equal to the number of consonants (a-z), i.e., 26, and each row storing the indices of the particular alphabets that appear in the cover text as column entries, is used. The procedure at the sender's side is described in the flowchart as given in Figure 1(a).

A cover text of sufficient length is used to conceal the secret text. To retain the position of each character, the cover text file is scanned and a two-dimensional array (A) is generated. The columns represent the occurrences (indices) of a specific character in the cover text, while each row corresponds to that character. Beginning from the first alphabet, say 'c,' one character at a time is read from the secret file. The index of the character 'c' in the two-dimensional array A is found. This index indicates the position of the character within the cover text. The number of occurrences of the character 'c' in the cover text is counted. This count is used as the modulus parameter (m) in the two-step stego-key generation algorithm.



(a) (b)

Figure 1

(a) Server-side procedure for Concealing Secret Text (b) Client-side procedure for deciphering the concealed message

The total number of characters (b) in the secret text is determined and is used in the algorithm as given in Equation (1). All these parameters are carefully selected to ensure a full-period pseudo-random sequence. This sequence determines the location of the secret data in the cover media, thereby reducing the predictability of the concealed data's location.

The mapping function is employed to determine the index within the cover text at which the secret text character 'c' would be mapped. The obtained index is used as input for AES, resulting in the creation of a modified index. This modified index serves as the ciphertext in AES. By following the steps as given in Figure 1(a), the secret text is embedded within the cover text, and the ciphertext is generated to facilitate the extraction of the hidden information.

3.2 *Retrieval Mechanism at the Receiver-Client Level*

The client-side procedure, as depicted in Figure 1(b), consists of key steps including receiving and initializing data, deciphering the concealed message, decrypting with the Inverse AES algorithm, and retrieving the hidden message. Upon receiving the cover text, IV, and modified indices (ciphertext in AES), the receiver initiates the process of deciphering the concealed message. The cover text is examined, and the Inverse AES algorithm is applied to restore the modi-

fied indices to their original values. The corresponding characters are retrieved, thereby disclosing the secret and hidden text.

3.3 *Working of the Proposed Model*

The suggested technique for embedding of the secret word "hello" within the cover text as provided in Figure 2 is illustrated in Figure 3. In the stated case, the first alphabet of the secret text was 'h,' and the values of a, b, and x0 were taken as 1, 5, and 9, respectively. Then, using the two-step stego-key generation algorithm, a random number was generated using equation (4) as

$$x_1 = (1 * 9 + 5) = 14 \tag{4}$$

and the corresponding index was generated as shown in equation (5).

$$i = (1 * 9 + 5)\ mod\ 10 = 4 \tag{5}$$

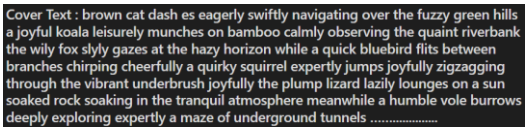


Figure 2

Snippet of the cover text.

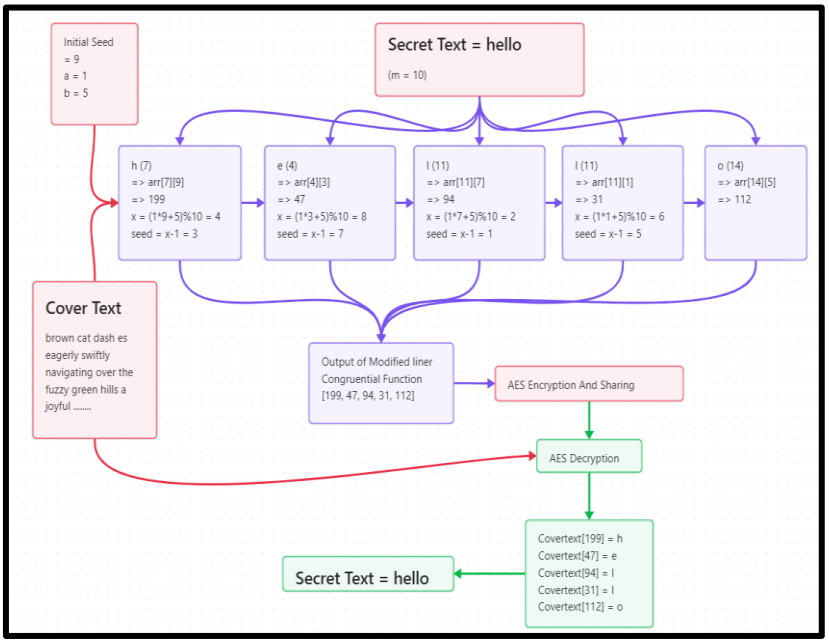


Figure 3

Implementation of the proposed method for a secret text "hello"

This generated number minus 1, i.e., $(i-1)$ was taken as the column number for selecting the index location of the next character of the secret text from array ***A*** for mapping. The next character in the secret text was **e**, and the value of the index at A [4][3] was 47. It was this index in the cover text where the alphabet '***e'*** of the secret text was mapped. Similarly, all characters in the secret text were mapped. Figure 4 illustrates the outcome of the mapping function along with the generation of IV and ciphertext, accompanied by an AES key.

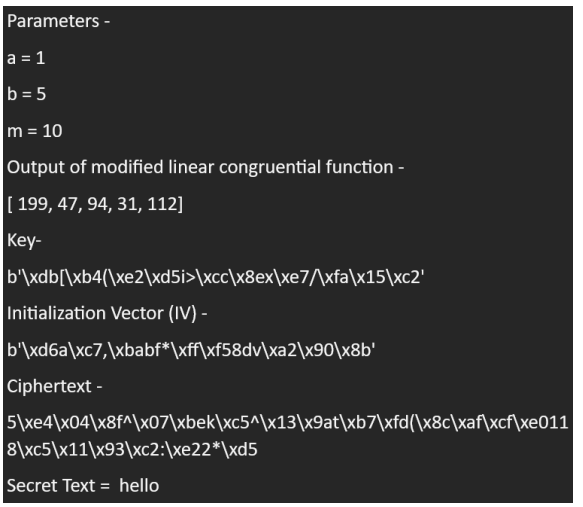


Figure 4
Output for a given secret text "hello" using AES

4 Result Analysis

The proposed work was implemented in Python using Google Colab. It employed 2D array for result storage and basic file handling for accessing both cover text and secret text files. Integer data types were used for indexing, with negative values resulting from integer overflow in step 1 were converted to positive values for subsequent steps. To test the robustness of the proposed method, covert text and parameters a, m, and b (e.g., reformatting) were modified to see if hidden data remains intact. For testing the algorithm, the cover text and secret text were generated randomly, potentially with repeated characters. To handle this variability, 'm' was adjusted to match the least frequent character in the cover text. This adjustment ensured that the output of the mapping function remained within the bounds of the alphabet's count in the cover text. Additionally, various parameters essential for the mapping function were assessed. The character indices from the cover text file were stored in a 2D array denoted as A. In this study, the authors capped the maximum occurrences of any character in the cover text to 10 to simplify the calculations. This influenced the size of the 2D array A to A[26][10] for index storage.

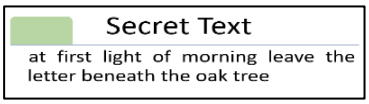
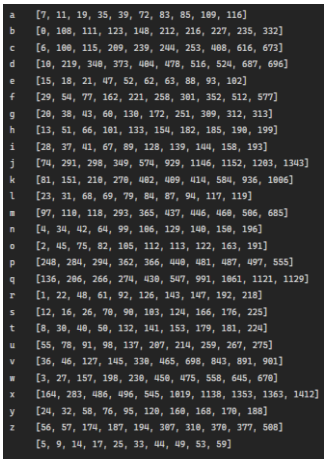


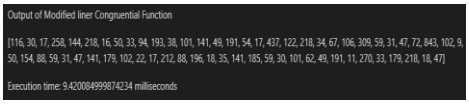
Figure 5

Snippet of the secret text

On the server side, the mapping process was initiated by identifying the first alphabet of the secret text as shown in Figure 5 and then locating its position in the 2D array A. The indices encoding the occurrences of secret text alphabets in the cover text were recorded in rows of a 2D array A as shown in Figure 4. Only the first 10 occurrences of all the given alphabets in a secret text were stored in A, as shown in Figure 6(a). Any of these indices stored for the selected alphabet could serve as an initial value to start the process. In this scenario, the starting position was chosen as "position 0," which is 116, corresponding to one of the indices of the first alphabet of the secret text found at location A[0][9].



(a)



(b)



(c)

Figure 6

Result of Mapping at the Sender Side: (a) 2-D array A, (b) Output of two-step stego-key mapping, (c) Output after AES

The multiplication parameter 'a' and the increment parameter 'b' for the two-step stego-key generation algorithm were chosen as 1 and 63, respectively. The value of 'b' corresponded to the total number of characters in the secret text file, as shown in Figure 7. The column number where the first character of the secret text occurred in the cover text acted as the seed value 'x_o' for the algorithm. Figure 6(b) illustrates the outcome of the mapping function for a specific cover text, along with the generation of an IV and ciphertext, accompanied by an AES key, as shown in Figure 6(c). The decrypted output is given in Figure 7.

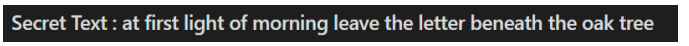
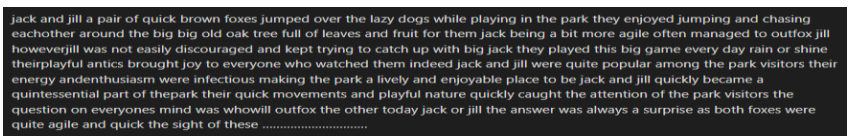


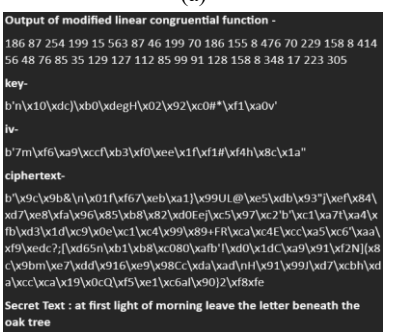
Figure 7

Snippet of Decrypted text

The robustness of the proposed method was tested against modifications like changing covert text and parameters a, m, and b (e.g., reformatting) to see if hidden data remains intact. When the cover text was changed, as demonstrated in Figure 8(a), the mapping shifted to a new position on the client side, consequently altering the output post-AES implementation as shown in Figure 8(b). With the alteration of initial values of the parameters such as 'a', 'm', and 'b', used in the mapping function, the resulting mapping also changed. This alteration is evident in Figure 9(a), where 'a' is modified to 2 and 3, respectively. Similarly, Figures 9(b) and 9(c) demonstrate the impact of changing 'm' from 10 to 6 and ‘b’ from 63 to 8, respectively, in the mapping process according to the described approach. These changes highlighted how adjustments in these parameters can influence the mapping outcomes, underscoring the sensitivity of the mapping algorithm to its input parameters.



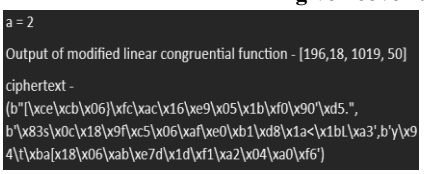
(a)



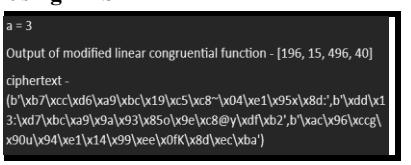
(b)

Figure 8

(a) New cover text for the proposed approach, (b) Output for a given cover text using AES



a=2



a=3

(a)

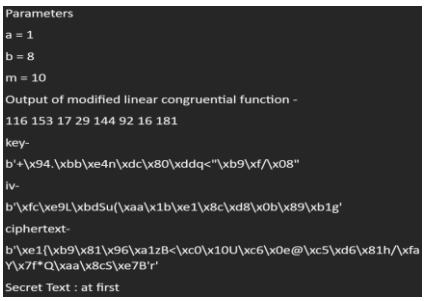


m=10



m=6

(b)



b=4



b=63

(c)

Figure 9

(a) Output after changing the value of the parameter a to 2 and a to 3, (b) Output when the value of m reduced from 10 to 6, (c) Execution time with the number of characters in secret text (b) equal to 8 and 63

Since the characters in the cover text are not altered and only the indices of the characters are used to create a stego key, the evaluation of visual similarity using PSNR and MSE (Mean Squared Error) becomes needless. The content of the cover text remained unchanged, and thus there were no noticeable differences between the original cover text and the stego text.

The processing time of the algorithm to map and encrypt the message and embed it into the cover text also plays a crucial role in assessing the overall performance of the system. The runtime of the implementation is intricately tied to the number of characters contained within the secret text. To assess this relationship, the algorithm's runtime was measured for varying numbers of characters in the secret text (denoted as 'b'), specifically considering cases where 'b' was set to 8 and 63. These evaluations were conducted using AES encryption at both the server-side and the client-side, and were obtained as 7.62 milliseconds and 12.95 milliseconds., respectively.

The time complexity of the proposed approach involving mapping secret text onto the cover text using AES encryption was evaluated. The approach consists of four key components: mapping secret text to cover text, generating positions using a modular arithmetic-based random function generator, creating the stego key, and encrypting the stego key with AES.

The overall time complexity of the proposed approach at the server side for mapping secret text onto cover text using a modular arithmetic-based random function generator and AES encryption was computed as linear, or $O(n)$. This indicates that performance scales linearly with the size of the secret text, making it efficient for practical applications in steganography. The overall time complexity at the client side for decrypting AES-encrypted data and retrieving original indices after performing the inverse of a modular arithmetic-based random function generator steganography was also linear with respect to the number of characters in the secret text or original indices, or $O(n)$. This indicates that while AES decryption is efficient and operates in constant time per block, any retrieval or reconstruction of indices will scale linearly with the size of the input data.

5 Comparative Analysis

A comparative analysis of the proposed approach against other recent literature in text steganography is presented in this section. The use of LSB steganography, though simple and easy to implement, suffers from significant vulnerabilities. This method, while capable of higher embedding capacities, are prone to detection and susceptible to even minor modifications of the carrier medium, thereby jeopardizes the integrity of the hidden message. Hybrid Cryptographic-Steganographic Techniques offer more security by combining cryptographic methods like AES with steganographic techniques, ensures that the message remains secure, even if detected [12]. Statistical methods, using the inherent statistical properties of language to conceal information, offer better imperceptibility. These techniques often involve modifying word frequencies or sentence structures to subtly embed data. However, their implementation complexity is high, demanding significant linguistic knowledge, and its capacity can vary, depending on the language and style used. Capacity optimization in text steganography while maintaining imperceptibility using machine learning models has been a focus for recent advancements [13].

Format-based steganography, on the other hand, focuses on manipulating non-content aspects of documents, such as font size, style, or whitespace. This approach maintains high readability and imperceptibility by not altering the text's meaning directly [14]. Yet, it is inherently limited by the document format and structure, which restrict its capacity and applicability.

Neural network-based approaches have also emerged as powerful tools in text steganography, offering adaptive, context-aware embedding strategies. These methods utilize deep learning models to achieve high imperceptibility and capacity by understanding the semantic nuances of the text [15]. However, they are computationally intensive, requiring significant training data and computational resources, and their complexity can be a barrier to implementation. Evalu-

ation metrics for Text Steganography such as PSNR, MSE, robustness, and processing time, have provided more insights and ways of evaluating the effectiveness of text steganography. A comprehensive review of modern techniques, including an arithmetic-based random function generator and their applications in secure communications, is presented [16].

6 Conclusions

This paper introduces an innovative approach to text steganography combined with cryptography. The steganographic technique employs a modular arithmetic-based method using modulo operations to generate pseudo-random numbers. The implementation of this method effectively transfers secret messages without raising suspicion, thereby avoiding potential investigations or disclosures of confidential information. Notably, the cover text remains unchanged throughout the process, making it unfeasible to reveal the hidden information by analyzing just the cover text.

The proposed algorithm transmits the modified indices, serving as the IV, key, and ciphertext, from the sender to the receiver over a secured channel. Even if unauthorized parties obtain the indices, it's hard to recover the secret message without access to the cover text, which is transmitted alongside other regular messages, making it tough to identify. Therefore, the integration of steganography and cryptography as proposed enhances security and ensures the safe transmission of information over open networks.

Acknowledgement

The authors are grateful to the Principal, Acharya Narendra Dev College for the support provided under the STAR DBT College Scheme.

References

[1] K. Herman, B. Y. Dwiandiyanta, and S. Suyoto, "Information hiding in CSS: A secure scheme text-steganography using public key cryptosystem," *International Journal on Cryptography and Information Security* (IJCIS), vol. 1, no. 1, pp. 13–22 (Dec. 2011), doi: 10.2139/ssrn.3635340.

[2] Z. I. Nezami, H. Ali, M. Asif, H. Aljuaid, I. Hamid, and Z. Ali, "An efficient and secure technique for image steganography using a hash function," *PeerJ Computer Science,* vol. 8, p. e1157 (Nov. 2022), doi: 10.7717/peerj-cs.1157.

[3] P. Sousa, A. Pinto, and P. Pinto, "Exploiting online services to enable anonymous and confidential messaging," *Journal of Cybersecurity and Privacy*, vol. 2, no. 3, pp. 700–713 (2022), doi: 10.3390/jcp2030035.

[4] R. Bala Krishnan, N. Rajesh Kumar, N. R. Raajan, G. Manikandan, A. Srinivasan, and D. Narasimhan, "An approach for attaining content

confidentiality on medical images through image encryption with steganography," *Wireless Personal Communications*, vol. 127, pp. 979–995 (2022), doi: 10.1007/s11277-021-08477-1.

[5] N. F. M. Sani and M. A. Nujjaid, "Efficient text data hiding technique using cryptography and steganography," *Journal of Computer Science*, vol. 21, no. 1, pp. 43–51 (2025).

[6] B. Kumar Pandey, D. Pandey, V. K. Nassa, T. Ahmad, C. Singh, A. S. George, and M. A. Wakchaure, "Encryption and steganography-based text extraction in IoT using the EWCTS optimizer," *The Imaging Science Journal*, vol. 69, no. 1–4, pp. 38–56 (2021), doi: 10.1080/13682199.2022.2146885.

[7] M. A. Majeed, R. Sulaiman, and Z. Shukur, "New text steganography technique based on multilayer encoding with format-preserving encryption and Huffman coding," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 12, pp. 163–172 (2022), doi: 10.14569/IJACSA.2022.0131222.

[8] D. L. Evans, P. J. Bond, and K. H. Brown, "Advanced Encryption Standard (AES)," FIPS PUB 197, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, Nov. 2001, updated May 9 (2023), doi: 10.6028/NIST.FIPS.197-upd1.

[9] J. Free, "Uniform random variate generation with the linear congruential method," *PANDION: The Osprey Journal of Research and Ideas*, vol. 1, no. 1, Art. 3 (2020). [Online]. Available: https://digitalcommons.unf.edu/pandion_unf/vol1/iss1/3

[10] A. Kumar, D. Idnani, K. Soni, N. Taneja, R. Shrivastava, and Ajeenkya D. Y. Patil University, "Steganography using AES algorithm," *International Journal of Research and Analytical Reviews,* vol. 6, no. 2, pp. 139–142 (2019).

[11] R. Kaur, A. Pant, F. Avi, P. Jain, and V. Gaur, "Text steganography using linear congruence," *International Journal of Information Technology, Communications and Convergence,* vol. 4, no. 3, pp. 237–264 (2024), doi: 10.1504/IJITCC.2024.145821.

[12] A. I. H. Al-Jarah and J. L. Ortega-Arjona, "Enhancing the capacity and robustness of an LSB algorithm using a novel insertion method, hashing function, and secret key," *IEEE Access*, vol. 12, pp. 159534–159544 (2024), doi: 10.1109/ACCESS.2024.3483832.

[13] A. Almalawi, S. Hassan, A. Fahad, F. Alqahtani, M. Alotaibi, and R. Alshamrani, "A hybrid cryptographic mechanism for secure data transmission in edge AI networks," *International Journal of Computational Intelligence Systems*, vol. 17, p. 24 (2024), doi: 10.1007/s44196-024-00417-8.

[14] M. Taleby Ahvanooey, Q. Li, J. Hou, A. R. Rajput, and Y. Chen, "Modern text hiding, text steganalysis, and applications: A comparative analysis," *Entropy*, vol. 21, no. 4, p. 355 (2019), doi: 10.3390/e21040355.

[15] O. Farook Mohammad, M. H. Alkawaz, and A. M. Fadhil, "Deep learning-based steganography: A neural network approach for secure data embedding," *Journal of Information Systems Engineering and Management*, vol. 10, Art. 9008, pp. 681–687 (2025), doi: 10.52783/jisem.v10i46s.9008.

[16] K. Bhattacharjee and S. Das, "A search for good pseudo random number generators: Survey and empirical studies," *Computer Science Review*, vol. 45, p. 100471 (2022), doi: 10.1016/j.cosrev.2022.100471.

Received May, 2025