Natural Resources for Human Health
eISSN: 2583-1194

Original Research

**Received** 20 May 2026
**Revised** 29 June 2026
**Accepted** 24 July 2026

Natural Resources for Human Health 2026, Vol. 6 (9s): 498–506

**KEYWORDS**:
Cryptography, Playfair, VTE algorithm, Ciphertext, Encryption algorithm, brute-force attacks

# Extended Vowel Text Encryption Algorithm for Robust Reversible Encryption

Sasmita Dash[1], Jayanta Mondal[1]*, Debabala Swain[2], Bimalendu Das[3], Prachee Dewangan[4]

[1]School of Computer Engineering, KIIT University, Bhubaneswar, India
[2]Dept of Computer Science, Rama Devi Women's University, Bhubaneswar, India
[3]Dept of Cloud & Cyber Security, WAPCOS, Gurgaon, Haryana
[4]Department of Computer Science and Engineering, Centurion University of Technology and Management, Bhubaneswar, India.
*Corresponding Author: Jayanta Mondal, jayanta.mondalfcs@kiit.ac.in

**ABSTRACT:** In an era where data transmission across networks holds increasing significance, data encryption becomes paramount. Two primary techniques employed for data encryption are Substitution and Transposition. Transposition involves altering the arrangement of characters within a given text, while substitution entails the replacement of each character in the plaintext with an alternative character. Here we discuss the existing Playfair algorithm, a substitution algorithm, its extended versions, and their advantages and disadvantages. This paper deals with the modified version of the VTE algorithm [6]. This advancement will increase the strength of the ciphertext, making it better than the existing VTE algorithm against the brute-force attacks present in today's world.

## 1. INTRODUCTION

Cryptography can be understood as the study or practice of writing codes or solving codes to hide information from unauthorized access. It plays a key role in securing sensitive information in this digital age. The fundamental concept of encryption systems involves transforming plaintext into ciphertext at the sender's location through the utilization of a secret key, employing an encryption algorithm. Subsequently, the ciphertext is deciphered at the receiver's location using the same secret key, but with a decryption algorithm. Encryption algorithms can be broadly categorized into two types based on their use of keys for both processes: symmetric and asymmetric. A singular private key is shared between the sender and the receiver, serving for both the encryption and decryption processes in symmetric encryption. Asymmetric encryption employs two distinct keys, a public key for encryption and a private key for decryption [1-3].

In modern technology environments like, IoT, cloud computing, and embedded systems, computational overhead is a major concern. So they need a proper balance between security measures and computational cost. Although classical techniques like AES and DES provide strong security but due to their time complexity, they are less suitable for different resource-constrained platforms. Hence, to address these kinds of challenges, different classical approaches are modified using different transformations to produce different hybrid encryption techniques. We have introduced such a method called Vowel Text Encryption (VTE), which is based on vowel-based transformations. It improves the randomness of the ciphertext. Still, it has limited diffusion capability and deterministic structure. To fix this, the proposed work represents an enhanced encryption scheme called Extended Vowel Text Encryption (EVTE). It is mainly designed to improve randomness with an expanded key space. It strengthens the algorithm's resistance against different cryptographic attacks. At the same time, it maintains low computational complexity.

The primary objective of this study is to develop a secure yet efficient cryptographic method that enhances ciphertext unpredictability, improves resistance to statistical and brute-force attacks, and remains suitable for low-resource environments. The proposed EVTE algorithm achieves this by integrating matrix-based substitution, vowel-driven

transformation, and dynamic key scheduling.

The Playfair cipher, a representative of polyalphabetic substitution [2] ciphers and a member of the symmetric cipher category, employs a (5×5) matrix for encrypting plaintext. However, the original Playfair cipher exhibited certain vulnerabilities, leading to the development of enhanced versions, including the extended Playfair matrix (6×6) and the universal Playfair matrix (7×4). In pursuit of increased robustness, the Playfair VTE (Vowel Text Encryption) method was introduced, leveraging a (5×6) matrix to fortify the encryption process. This evolution aims to address security concerns and enhance the cryptographic strength of the Playfair cipher.

## 2. LITERATURE REVIEW

The Playfair cipher is one of the popular classical symmetric key encryption techniques. It encrypts pairs of characters similar to other monoalphabetic substitution cipher techniques. It provides better security than contemporary encryption techniques. It has introduced the polygraphic substitution concept. A number of advanced Playfair algorithms has been proposed over the last decade to overcome the restricted key and predictable substitution pattern constraints [7-10,14,16].

Early researchers have primarily focused on increasing the complexity of the traditional 5×5 matrix. Then the 6×6, 8×8, and even 16×16 matrix structures were introduced to accommodate alphabets, numerals, punctuation marks, and extended ASCII characters. So the substantial key space was enlarged with less ambiguity. It also helped to communicate messages containing alphanumeric and special characters. Further, many researchers introduced the concept of dynamic key generation. This is possible using the techniques like, pseudorandom number generator, chaotic systems, hash functions, DNA-based encoding, evolutionary algorithms etc. Due to the randomness of the keys, the brute force attacks become more challenging. Several studies also propose multiple secret keys techniques, layer key to strengthen security. These techniques improve both confusion and diffusion principles. These technique provides greater resistance against unauthorized access and cryptanalytic attacks.

The Playfair cipher technique combined with different classical techniques like, AES, RSA, DES, Blowfish, Hill Cipher, Vigenère Cipher, Transposition Cipher, Elliptic Curve Cryptography to evolve new hybrid cryptographic approaches. These hybrid techniques achieve better computational efficiency and strong security. Many research studies have been proposed using advanced Playfair cipher algorithms featuring steganography, and reversible data hiding techniques [11-13, 15, 17-22]. Here, the encrypted information gets embedded with cover images, text documents or video streams, etc. Due to this a multi-layer security architecture is formed. It enhances the privacy of the hidden information and resistance against different forensic attacks. These techniques are most suitable for sensitive images like, military image, forensic evidence, and secure cloud storages.

In spite of all these advanced techniques, the Playfair techniques also face many challenges. It suffers from computational overhead, limited scalability for high volume data. These are less resistant against sophisticated cryptanalytic attacks and resource-constrained environments. Hence, the new techniques should strengthen the encryption and embedding technique to ensure more confidentiality, integrity, reversibility, and zero tolerance towards information loss.

### 2.1 Traditional Playfair (5×5) Matrix

This conventional algorithm [2, 4] is simple yet intuitive. It proposes using a 5×5 matrix to store English characters. The characters (I) and (J) are placed in the same position in the matrix.

A keyword is used to generate a particular ciphertext, and the matrix is filled with this keyword from left to right. The remaining letters are filled in other positions. The plain text is divided into blocks of two letters. If the text has an odd number of letters, an extra character is appended. Similarly, for duplicate letters, an extra character is used to segregate them. Encryption is done in the following manner:

i. The characters in the pairs belonging to the same row of the matrix are replaced with the character to the right (in a cyclic manner).

ii. The characters in the pairs belonging to the same matrix column are replaced with the character to the bottom (in a cyclic manner).

iii. The characters in the pairs belonging to different rows or columns of the matrix are replaced with the character in its row

or column.

The decryption process is the same as done in a reverse manner. This algorithm has several shortcomings, such as the presence of I and J in the same cell, which creates issues in the decryption process. The addition of extra letters brings about utter confusion as to whether the letter was present in the plain text or was used as an extra letter.

### 2.2 Extended Playfair (6×6) Matrix

In the year 2011, an extension to the conventional Playfair algorithm was introduced by Ravindra Babu et al. [4] to overcome the shortcomings of the existing one. To address the placement of the letters 'I' and 'J' in different cells, a 6×6 matrix is used. The keyword and remaining letters are filled in the same way as in the traditional one, with the remaining cells filled with numbers from 0 to 9. This extension can encrypt numeric values as well. All the remaining processes remain unchanged. The goal was to eliminate the confusion between the letters 'I' and 'J' and to encrypt both alphabets and numbers. However, this algorithm failed to overcome the other setbacks faced by the traditional one.

### 2.3 Universal Playfair (7×4) Matrix

Aftab Alam et al. [5] proposed yet another version of the traditional Playfair algorithm, wherein a 7×4 matrix was used instead of 5×5, with the addition of two special characters '*' and '#'. The placement of the keyword and letters is the same, with the last two cells filled with '*' and '#'. The process of encrypting and decrypting letters remains the same. The only distinction is in the separation process, where '*' is added to separate the duplicate characters, and '#' is added to make them even. This removes any confusion in the process of decryption. However, it still fails to meet the security standards.

### 2.4 Vowel Text Encryption (VTE)

In 2023, S Dash et al. proposed a vowel text encryption algorithm [6], employing a 6×5 matrix for the storage of uppercase English characters. The concept involves the use of a (6×5) matrix. A secret key is generated by assigning an index to each row and column, agreed upon by both the sender and the receiver. This matrix is formed by filling characters in a row from left to right, starting with 'A' and progressing to the next row upon encountering the next vowel. Any remaining empty cells are filled with a special symbol, '$\phi$.'

During encryption, the plain text is divided into individual characters and replaced by the integers corresponding to the row and column indices. In the decryption stage, the ciphertext is divided into blocks of two letters, where the first number denotes the row number, and the second represents the column number. The plain text is generated by replacing those numbers with the respective characters in the matrix.

## 3. PROPOSED EXTENDED VOWEL TEXT ENCRYPTION (EVTE)

We have extended the existing VTE algorithm to improve the security of the cipher. This version uses a 5×8 matrix for storing the uppercase alphabet. The matrix is filled column-wise starting from the first column and filling the first place with 'A'. The process is repeated until the next vowel is found. When a vowel is encountered, it will start from a new column. The last two rows are filled using numbers ranging from '0' to '9'. The vacant cells are filled with '$\alpha$', '$\beta$', '$\gamma$', and, '$\Phi$'

### 3.1 Key Generation

The generation of the key has the following steps:

1. Assign any random 104-bit stream binary number.

2. Divide the 104 bits from the MSB into 40-bit streams for creating the column.

3. The rest of the 64-bit stream will be used for creating the row.

4. For column indices, a total of 5 of 2 digit decimal numbers and for row indices, a total of 8 of 2 digit decimal numbers are needed.

5. Represent this key in the bit stream (Column and row numbers will be represented using a 2-digit Decimal equivalent).

6. Perform a 1-bit cyclic right shift in this total bit stream (8-bit binary) for the column indices part and make the equivalent 2-digit decimal number.

7. Perform mod 100 if the decimal equivalent exceeds 99, and if any number is repeated in column indices; keep adding 1 to it until a new number gets generated.

8. Repeat Step 5,6,7 to generate the row indices and compare the bits within the rows.

9. This stream of 13 integers is the key for the next round. The first five 2-digit integers in the stream are column indices and the next eight 2-digit integers are row indices.

**Let's take an example:**

1. 0110001101010111010000010010100000000000100011101001001010100010000000000000001100000011110101110100101100

2. 0110001101010111010000010010100000000001- For column index

3. 0001110100100101010001000000000000000110000001111010111010010110 0 - For row index

4. 99,87,65,40,01 - For column index (in 2-digit decimal equivalent)

29,37, 68,00,12,15,93,44 - For row index (in 2-digit decimal equivalent)

5. Key representation is given in Table-1, 2.

**Table 1:** Encryption Matrix Representation

| | **99** | **87** | **65** | **40** | **01** |
|---|---|---|---|---|---|
| **29** | A | E | I | O | U |
| **37** | B | F | J | P | V |
| **68** | C | G | K | Q | W |
| **00** | D | H | L | R | X |
| **12** | α | β | M | S | Y |
| **15** | γ | Φ | N | T | Z |
| **93** | 0 | 1 | 2 | 3 | 4 |
| **44** | 5 | 6 | 7 | 8 | 9 |

**Table 2:** Key Generation Table

| **Initial 8-bit key** | **Decimal (Key 1)** | **1-bit cyclic right shift** | **Integer representation** | **Modulo 100** | **Key 2** |
|---|---|---|---|---|---|
| 01100011 | 99 | 00110001 | 49 | 49 mod 100 | 49 |
| 01010111 | 87 | 10101011 | 171 | 171 mod 100 | 71 |
| 01000001 | 65 | 10100000 | 160 | 160 mod 100 | 60 |
| 00101000 | 40 | 10010100 | 148 | 148 mod 100 | 48 |
| 00000001 | 01 | 00000000 | 00 | 00 mod 100 | 00 |
| 00011101 | 29 | 10001110 | 142 | 142 mod 100 | 42 |
| 00100101 | 37 | 10010010 | 146 | 146 mod 100 | 46 |
| 01000100 | 68 | 10100010 | 162 | 162 mod 100 | 62 |

| | | | | | |
|---|---|---|---|---|---|
| 00000000 | 00 | 00000000 | 00 | 00 mod 100 | 00 |
| 00001100 | 12 | 00000110 | 06 | 06 mod 100 | 06 |
| 00001111 | 15 | 00000111 | 07 | 07 mod 100 | 07 |
| 01011101 | 93 | 10101110 | 174 | 174 mod 100 | 74 |
| 00101100 | 44 | 10010110 | 150 | 150 mod 100 | 50 |

**3.2 Encryption Algorithm**

1. Add 'α' as the starting character in the plaintext.

2. Add 'β' as the ending character in the plaintext.

3. Add 'ϕ' as the indication of the space between the words.

4. Divide the plaintext into sub-parts, each containing 8 characters.

5. If any sub-part has less than 8 characters, fill it with 'γ' to make the length 8.

6. Different keys generated in each round are used to encrypt the text of each sub-part.

7. Replace each character in the plaintext with its corresponding column index followed by its row index to generate the ciphertext.

**Let's take an example:**

1. Plain Text: COPY THE TEXT

2. Adding starting, ending, separator, and padding character αCOPYΦTHEΦTEXTβγ

3. Plaintext to be encrypted using Key 1: αCOPYΦTHEΦTEXTβ

4. First part - αCOPYΦTH

Second part - EΦTEXTβ

5. Addition of 'γ' to the second part - EΦTEXTβγ

6. Plaintext first part to be encrypted using Key 1: 998765400129376800121593 44

Plaintext first part to be encrypted using Key 2: 49716048004246620106077450

7. Ciphertext generated for the first part: 991299684029403701128715401587 00

Ciphertext generated for the second part: 71427107480771420000480771064907

**3.3 Decryption Algorithm Steps**

1. Divide the generated ciphertext into sub-parts. Each sub-part consists of 32 digits.

2. Further subdivide every 32 digits into 8 sub-parts, each containing 4 digits.

3. In each subpart of 4 digits, the first 2 digits represent the column number and the next 2 digits represent the row number.

4. Replace each of these 4-length strings with the element present in the corresponding column index and row index.

5. For the first 32 digits, the initial key will be used for decryption, for the next 32 digits, the second key will be used, and so on.

**Explanation of Decryption**

1. First part - 991299684029403701128715401587 00

   Second part- 7142710748077142000048077106 4907

2. First part (divided into 8 subparts)- 9912 9968 4029 4037 0112 8715 4015 8700

   Second part (divided into 8 subparts) - 7142 7107 4807 7142 0000 4807 7106 4907

3. Ciphertext to be decrypted using Key 1: 9912 9968 4029 4037 0112 8715 4015 8700

   Key 1 – 998765400129376800121593 44

   Plain text generated – αCOPYΦTH

4. Ciphertext to be decrypted using Key 2: 7142 7107 4807 7142 0000 4807 7106 4907

   Key 2 – 497160480042466201060774 50

   Plain text generated – EΦ TEXTβγ

5. Finally, the message that the receiver will get – "COPY THE TEXT".

## 4. COMPARATIVE ANALYSIS

It can be easily analysed from the above table that the proposed EVTE algorithm utilizes a key of length 104 bits. It yields a key space of 2104. So, it can create 2104 number of possible keys, which is considerably larger than the Data Encryption Standard. The DES algorithm offers only 256 number of possible keys. So, it can offer enhanced resistance against brute force attacks. Although the key space of EVTE is smaller than the Advanced Encryption Standard-128 algorithm, which produces 2128 number of possible keys, still it provides a substantial high-level computational security in real-time applications.

**Table 3:** Comparison of Key Sizes and Key Spaces of EVTE, DES, and AES

| NAME OF THE ALGORITHM | KEY SIZE | NUMBER OF POSSIBLE KEYS |
|---|---|---|
| PROPOSED ALGORITHM(EVTE) | 104 bits | 2^104 |
| DATA ENCRYPTION STANDARD (DES) | 56 bits | 2^56 |
| ADVANCED ENCRYPTION STANDARD (AES) | 128 bits | 2^128 |

**Table 4:** Comparative Analysis with Respect to Execution Time (in Sec)

| Plain text | AES | DES | 3DES | EVTE (8*5) |
|---|---|---|---|---|
| CRYPTOGRAPHY IS TO DATA SECURITY WHAT A LOCK IS TO A KEY | 0.005634 | 0.005056 | 0.005283 | 0.003473 |
| PROGRAMMING IS THE ART OF CREATING INSTRUCTIONS FOR COMPUTERS TO EXECUTE SPECIFIC TASKS EFFICIENTLY AND ACCURATELY | 0.006267 | 0.004778 | 0.00598 | 0.001992 |
| WEB DEVELOPMENT INVOLVES CREATING AND MAINTAINING WEBSITES | 0.067783 | 0.005118 | 0.00626 | 0.001991 |

Table 4 compares the execution times of AES, DES, 3DES, and the proposed EVTE algorithm for three plaintext samples of varying lengths. Execution time is a critical performance metric, as lower values indicate faster encryption and decryption processes [7-10, 14,16].

For the first plaintext, "CRYPTOGRAPHY IS TO DATA SECURITY WHAT A LOCK IS TO A KEY", EVTE records an execution time of 0.003473 s, which is lower than AES (0.005634 s), DES (0.005056 s), and 3DES (0.005283 s). This demonstrates that EVTE performs more efficiently for short text messages.

For the second plaintext, "PROGRAMMING IS THE ART OF CREATING INSTRUCTIONS FOR COMPUTERS TO EXECUTE SPECIFIC TASKS EFFICIENTLY AND ACCURATELY", EVTE requires only 0.001992 s, whereas AES, DES, and 3DES require 0.006267 s, 0.004778 s, and 0.005980 s, respectively. The proposed algorithm therefore achieves a substantial reduction in processing time compared to the conventional encryption algorithms.

Similarly, for the third plaintext, "WEB DEVELOPMENT INVOLVES CREATING AND MAINTAINING WEBSITES", EVTE records the lowest execution time of 0.001991 s, while AES, DES, and 3DES require 0.067783 s, 0.005118 s, and 0.006260 s, respectively. The significant increase in AES execution time for this test case further highlights the computational advantage of EVTE.

The experimental evaluation demonstrates that the proposed Extended Vowel Text Encryption (EVTE) algorithm consistently achieves lower execution times than AES, DES, and 3DES under the experimental conditions considered in this study. The observed performance indicates that EVTE incurs comparatively lower computational overhead during the encryption process, enabling faster execution across varying plaintext lengths. Moreover, the execution time remains relatively stable as the input size increases, suggesting that the algorithm exhibits predictable computational behaviour and efficient resource utilization.

This performance characteristic can be attributed to the algorithm's lightweight substitution-based architecture and dynamic key evolution mechanism, which together provide enhanced cryptographic functionality without imposing significant computational complexity. Unlike conventional block ciphers that involve multiple rounds of complex mathematical transformations, permutation operations, substitution boxes (S-boxes), and key expansion procedures, EVTE employs a comparatively streamlined encryption process that minimizes processing latency while maintaining dynamic ciphertext generation.

The scalability analysis further indicates that the increase in execution time with respect to plaintext size is gradual and nearly linear, reflecting the algorithm's suitability for processing both short messages and moderately large textual datasets. Such predictable performance characteristics are particularly advantageous in environments where deterministic response times and low processing delays are critical.

Consequently, the proposed EVTE algorithm can be regarded as a lightweight encryption framework with favourable computational efficiency, making it well suited for deployment in resource-constrained environments, including Internet of Things (IoT) devices, embedded systems, edge computing platforms, wireless sensor networks, mobile applications, and real-time communication systems. Its reduced computational overhead also contributes to lower processor utilization and energy consumption, both of which are important design considerations for battery-powered and low-resource devices.

It is important to note, however, that while EVTE demonstrates superior execution speed under the evaluated experimental conditions, execution time alone does not constitute a comprehensive measure of cryptographic strength. Security evaluation must also consider resistance to cryptanalytic attacks, key space, entropy, diffusion and confusion properties, statistical randomness of ciphertext, and resilience against known-plaintext, chosen-plaintext, and brute-force attacks. Therefore, EVTE should be assessed through a balanced framework that jointly considers both computational efficiency and cryptographic security.

Within this context, the experimental results indicate that EVTE offers an attractive trade-off between processing performance and security, making it a promising candidate for lightweight secure communication applications where rapid encryption and low computational cost are primary operational requirements.

## 5. CONCLUSION

The proposed EVTE method can perform better than the conventional Playfair cipher by using enhanced cryptographic methods with flexibility and efficient implementation. The proposed framework has substantial expanded key space of 2104 possible key combinations. Hence, an exhaustive brute-force attack is infeasible to crack the keys. In addition to its resistance against brute-force attack, the proposed algorithm effectively mitigates the attacks through the dynamic key generation mechanism. The traditional Playfair method relies on a static substitution matrix throughout the encryption.

Future extensions of EVTE may include RDH and encrypted domain techniques to support secure multimedia and cloud-based applications [11-22]. Further, the proposed framework exhibits excellent extensibility by increasing the dimensions of the encryption matrix and expanding the key generation mechanism. EVTE can be extended to support additional special characters, punctuation symbols, multilingual character sets, Unicode representations, etc. Consequently, EVTE represents a scalable and adaptable cryptographic solution.

## REFERENCES

[1] Chandra, S., Paira, S., Alam, S. S., & Sanyal, G. (2014). A comparative survey of symmetric and asymmetric key cryptography. In 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE) (pp. 83–93). IEEE. https://doi.org/10.1109/ICECCE.2014.7086640

[2] Hannan, S. A., & Asif, A. M. A. M. (2017). Analysis of polyalphabetic transposition cipher techniques used for encryption and decryption. International Journal of Computer Science and Software Engineering, 6(2), 41–46.

[3] Stallings, W. (2017). Cryptography and network security: Principles and practice (7th ed.). Pearson.

[4] Ravindra, K., Kumar, S., Babu, A., Aditya, I., & Komuraiah, P. (2011). An extension to traditional Playfair cryptographic method. International Journal of Computer Applications, 17. https://doi.org/10.5120/2213-2814

[5] Alam, A., Ullah, S., Wahid, I., & Khalid, S. (2011). Universal Playfair cipher using M×N matrix. International Journal of Advanced Computer Science, 1, 113–117.

[6] Dash, S., Mondal, J., Bhattacharyya, A., & Swain, D. (2023). VTE—An advanced Playfair encryption method. In AIP Conference Proceedings (Vol. 2981, No. 1, Article 020038). AIP Publishing. https://doi.org/10.1063/5.0182891

[7] Nechvatal, J. R., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., & Roback, E. (1999). Status report on the first round of the development of the Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST).

[8] Daemen, J., & Rijmen, V. (2002). The design of Rijndael: AES—The Advanced Encryption Standard. Springer. https://doi.org/10.1007/978-3-662-04722-4

[9] Dworkin, M. (2001). Recommendation for block cipher modes of operation: Methods and techniques (NIST Special Publication 800-38A). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-38A

[10] Dworkin, M., & Mouha, N. (2024). Report on the block cipher modes of operation in the NIST SP 800-38 series (NIST Interagency Report 8459). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.IR.8459

[11] Puteaux, P., Ong, S. Y., Wong, K. S., & Puech, W. (2021). A survey of reversible data hiding in encrypted images: The first 12 years. Journal of Visual Communication and Image Representation, 77, 103093. https://doi.org/10.1016/j.jvcir.2021.103093

[12] Gandhi, S., & Kumar, R. (2025). Survey of reversible data hiding: Statistics, current trends, and future outlook. Computer Standards & Interfaces, 92, 104004. https://doi.org/10.1016/j.csi.2024.104004

[13] Ren, F., Zhang, Z., Jiang, K., et al. (2025). Reversible data hiding and authentication scheme for encrypted image based on prediction error compression. Scientific Reports, 15, Article 95433. https://doi.org/10.1038/s41598-025-95433-9

[14] Ganesh, R., Khan, B. U. I., Khan, A. R., et al. (2025). A panoramic survey of the Advanced Encryption Standard: From architecture to security analysis, key management, real-world applications, and post-quantum challenges. International Journal of Information Security. Advance online publication. https://doi.org/10.1007/s10207-025-00986-4

[15] Mohammadi, A. (2021). A general framework for reversible data hiding in encrypted images by reserving room before encryption. Multimedia Tools and Applications, 80(20), 30921–30945. https://doi.org/10.1007/s11042-021-10817-6

[16] Cohen, A., D'Oliveira, R. G. L., Duffy, K. R., Woo, J., & Médard, M. (2022). AES as error correction: Cryptosystems for reliable communication. IEEE Transactions on Information Theory, 68(9), 6135–6152. https://doi.org/10.1109/TIT.2022.3177635

[17] Wan, H., Zhang, M., Ke, Y., et al. (2024). General secure encryption algorithm for separable reversible data hiding in encrypted domain. Journal of King Saud University – Computer and Information Sciences, 36(8), 102217. https://doi.org/10.1016/j.jksuci.2024.102217

[18] Zhang, Y., Li, X., Wang, J., et al. (2024). A fine-grained reversible data hiding in encrypted domain based on the ciphertext redundancy of encryption process. Heliyon, 10(18), e7573. https://doi.org/10.1016/j.heliyon.2024.e7573

[19] Yavuz, E. (2025). Reversible data hiding in encrypted images using chaos theory and Chinese remainder theorem. Pattern Analysis and Applications, 28(2), 1–15. https://doi.org/10.1007/s10044-025-01495-w

[20] Ke, Y., Liu, J., & Han, Y. (2025). Two-stage reversible data hiding in encrypted domain with public key embedding mechanism. Signal Processing, 229, 109887. https://doi.org/10.1016/j.sigpro.2025.109887

[21] Wang, X., & Ding, H. (2025). Encrypted domain reversible data hiding based on proxy re-encryption in distributed environment. International Journal of Information Security and Privacy. Advance online publication. https://doi.org/10.1007/s44443-025-00250-9

[22] Saeidi, Z., & Mirzakuchaki, S. (2025). Reversible data hiding in encrypted images using LWE-based secret sharing. Scientific Reports, 15, Article 28912. https://doi.org/10.1038/s41598-025-28912-8