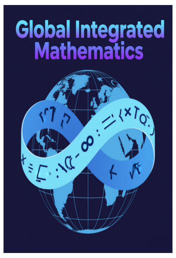
Global Integrated Mathematics

https://gim.gospub.com/gim

Global Open Share Publishing



Article

A Graph-Based Cryptographic Framework Using Complete Graphs and Lower Triangular Identity Key Matrices

Maimoona Safdar*

Department of Mathematics, COMSATS University Islamabad, Vehari Campus, Vehari, Pakistan

*Corresponding author: Maimoona Safdar, maimoonasafdar55@gmail.com

Abstract

In this study, we present a new cryptographic method inspired by the principles of graph theory, offering an innovative way to ensure data security through the use of complete graphs, adjacency matrices, and edge weights. Our approach involves encoding data by mapping it onto weighted complete graphs and performing specific matrix operations on their corresponding adjacency structures. What sets this method apart is the introduction of a lower triangular identity matrix, which serves as a key component in the encryption process. This key design not only strengthens the security of the algorithm but also contributes to its structural simplicity and ease of implementation. The framework we propose takes advantage of the inherent properties of graph theory such as connectivity, symmetry, and weight distribution to provide a mathematically sound and computationally efficient encryption mechanism. Through detailed analysis and a series of practical tests, we demonstrate that the proposed technique is resilient to several well-known cryptographic attacks. Furthermore, its design enables it to be adaptable for various types of digital data, making it a promising tool for modern applications where data protection is critical. By combining theoretical insight with real-world applicability, this paper highlights how classical mathematical concepts can be reimagined to meet the evolving demands of cybersecurity.

Keywords

Graph, Complete graph, Adjacency matrix, Matrix encryption, Identity matrix

Article History

Received: 05 August 2025

Revised:17 August 2025

Accepted: 02 September 2025

Available Online: 05 September 2025

1. Introduction

Cryptography is the science of communicating and providing information in such a way that unauthorized access or alteration of that information is prevented. Cryptography is based on algorithms which transform data into unreadable format and guarantee (confidentiality, integrity, and authenticity). Today's connected world relies increasingly on the use of digital communication and exchange, especially of data and information, and cryptography has become essential to ensuring this information is being handled securely, particularly when it concerns financial transactions, a personal identity, and classified data owned by government [1]. Technologies like online banking, e-commerce, secure messaging, are all built on the assumption that they can take place in a safe and secure environment, and that includes both stopping cyber threats such as hacking, identity theft and breach of data [2,3]. Cryptography and network security have learnt very much from graph theory. Recent research has investigated different ways in which even graphing algorithms may be used to safeguard communication networks, and particularly encryption methods [4]. One area which has been largely addressed is the application of complete graphs in encryption, providing new means for data protection [5]. Graph based encryption techniques also incorporate self-invertible matrices, creating these matrixes special to cryptographic systems [6,7]. These matrices are useful when the graph structures are complete graph structures for building encryption methods that are durable against different types of attacks [8,9]. Results of new solutions to protecting knowledge graph and other complex data forms [10] have arisen from adaptation of graph theory with the ordinary cryptographic approaches. Graph based crypto is now largely being implemented in python, and therefore, it has become much easier for researchers to develop and analyze new encryption techniques. Graph theory is used in modern cryptography, as basic encryption is not enough, they use this to assist with network security protocols, and other advanced encryption standards [11]. Moreover, graph labeling is being employed in recent cryptographical research to explore methods for encryption [12], making graph theory less applicable to cryptography and more generally applicable. For more applications readers are encouraged to read [13-15]. Cryptography has been the cornerstone for many years in securing trust in and privacy of the digital age and will continue to play a crucial role in solving ever evolving cybersecurity challenges to withstand threats. The main innovation of this study is the introduction of a graph-based cryptographic framework that combines complete graphs with lower triangular identity key matrices. This integration ensures a hierarchical and non-redundant key generation mechanism, which enhances resistance against structural and brute-force attacks while maintaining computational simplicity. Unlike conventional number-theoretic cryptosystems that rely on prime factorization or discrete logarithms, our approach leverages graph connectivity properties and matrix operations, thereby offering polynomial-time complexity for both encryption and decryption. This balance of mathematical rigor, structural security, and computational efficiency distinguishes our proposed framework from existing graph-based and classical cryptographic methods.

In this research, we propose a novel cryptographic method using transformations from cycle graphs to path graphs with adjacent matrix techniques in the theory of graphs. Instead, this approach encodes data in graph structures that transparently encrypt each word as well as entire sentences, allowing flexibility to deal with a variety of data formats. The methodology takes advantage of the structure of these graph representations and achieves strong encryption and decryption through matrix-based operations. This approach is shown [16,17] to be effective both in theoretical analysis and practical experiments to enhance data security. The results show that cycle graph transformations and adjacency matrices can lend themselves to cryptographic applications beyond their use in networks, adding a new and important perspective on secure data communication in modern networks.

A graph-based encryption technique provides a novel approach to secure communication by using the properties of graphs such as connectivity, structure and symmetry. All these methods are resistant to classical cryptographic attacks and are secure, while keeping design flexibility. This paper claims a contribution to this growing field by presenting algorithms based on graph theory and on the matrix operations which provide the robust cryptographic solutions. Besides being theoretically meaningful, these techniques have respective practical consequences for modern cryptography. This study integrated abstract mathematical concepts to advance cryptographic capability, while illustrating how real-world security challenges may be addressed. Some studies can be seen in [18-20]. Figure 1 presents the flowchart of the article.

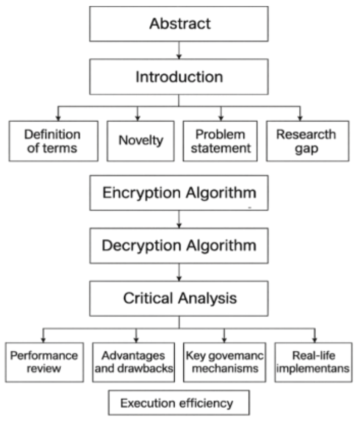


Figure 1. Flow chart.

1.1 Problem Statement

In the current landscape of cybersecurity, traditional cryptographic systems are facing increasing challenges from advancements in computational power and emerging attack strategies. Many existing encryption algorithms rely heavily on number-theoretic assumptions or complex key exchange protocols, which, while effective, can be computationally intensive or vulnerable to quantum computing threats. Additionally, ensuring both high security and operational simplicity within a single cryptographic framework remains a significant hurdle. There is a growing need for alternative methods that not only provide strong data protection but also offer structural clarity and computational efficiency. This study addresses the problem of developing a lightweight yet robust cryptographic scheme that can resist conventional attacks while being practical for real-world applications.

1.2 Novelty Statement

This paper introduces a novel cryptographic framework that uniquely integrates graph theory into the design of encryption and decryption algorithms. By utilizing weighted complete graphs and matrix operations on their adjacency representations, the proposed method departs from traditional number-theoretic encryption schemes. A key innovation lies in the use of a lower triangular identity matrix as the encryption key a simple yet powerful structure that enhances both the algorithm's security and ease of implementation. Unlike existing approaches, this matrix-based technique harnesses the topological and algebraic properties of graphs to create a cryptographic system that is not only secure but also scalable and computationally efficient. The originality of the method is further demonstrated through theoretical validation and experimental results, highlighting its potential as a viable alternative in modern cryptographic applications.

The primary contribution of this work is the design of a graph-based cryptographic framework that integrates weighted complete graphs with a lower triangular identity matrix as the encryption key. Unlike conventional number-theoretic methods such as RSA and ECC, which rely heavily on factorization and discrete logarithms, our approach leverages graph connectivity, adjacency matrices, and weight distributions to achieve encryption and decryption with polynomial-time efficiency. This fills a research gap in graph-based cryptography by introducing a structured, hierarchical, and easy-to-generate key that enhances both security and implementation simplicity. Through comparative analysis and experimental validation, we show that our framework is resistant to common cryptographic attacks and offers scalability, making it distinct from prior graph-theoretic cryptosystems.

1.3 Preliminaries

We deal with improvements to encryption with the application of graph theory and algebraic concepts to these structures. First, we need to go back to some basic concepts on graph theory. Basic definitions are taken from source [21-25].

Definition 1: a graph $G = (V, E)$ consists of a set of vertices V and a set of edges E, where each edge connects two vertices, representing relationships between entities.

Definition 2: a graph in which every pair of distinct vertices is connected by a unique edge. It is denoted by K_n, where n is the number of vertices.

Definition 3: a square matrix used to represent a graph, where each element a_{ij} indicates the presence (and possibly weight) of an edge between vertex i and vertex j.

Definition 4: an adjacency matrix is a square matrix used to represent a graph, where each element a_{ij} is 1 if there's an edge between vertices v_i and v_j, and 0 otherwise.

Definition 5: a square matrix with 1s on the diagonal and 0s elsewhere. It acts as the multiplicative identity in matrix operations.

Definition 6: a square matrix in which all the entries above the main diagonal are zero.

Definition 7: numerical value assigned to an edge in a graph, representing cost, distance, or any other measure relevant to the problem.

2. Methodology and Main Results

2.1. Encryption and Decryption Algorithm

2.1.1. Encryption Algorithm

The proposed encryption method securely transforms a plaintext message of arbitrary length n into a cipher matrix using graph-theoretic and matrix-based operations. The process is described below:

(1) Construct the encryption table.

We set the column numbers $0, 1, 2, 3, \ldots, n$ and the row numbers $n+1, n+2, \ldots, m$. S characters are randomly scattered in the table, where S is a collection of characters from the original message. Here, we choose set S, which has the following elements: 26 alphabets, a dot (.), and a space. Table 1 is a sample table.

Table 1. Encryption table.

	0	1	2	3	4	5	6
7	A	B	C	D	E	F	G
8	H	I	J	K	L	M	N
9	O	P	Q	R	S	T	U
10	V	W	X	Y	Z	SPACE	DOT

Each character is assigned a numerical value in Table 1. We have two categories for assigning the value of the number: The consonant letters are all divided into the first group. The second group includes DOT and SPACE as well as all the vowels. For Group 1, the initial characters represent the column and the last character denotes the row number. For Group 2, the last character denotes the column and remaining the row number. For example, A = 70, E = 74, X = 210, U = 96, DOT = 106, SPACE = 105.

(2) Character encoding.

A plaintext word or message of length n is selected. Each character is mapped to a unique numerical value using a predefined Table 1. These values represent the vertices $V = \{v_1, v_2, \ldots, v_n\}$ of a complete graph Kn.

(3) Construction of adjacency matrix M_1.

A symmetric adjacency matrix $M_1 \in Z^{n \times n}$ is constructed, where each entry a_{ij} (for $i \neq j$) is computed as:

$$a_{ij} = (| v_i - v_j |) \bmod N$$

where N is a chosen modulus. The diagonal elements a_{ii} are set to zero. This matrix represents the edge weights of the complete graph derived from the input characters.

(4) Generation of modified matrix M_2.

Matrix M_2 is derived from M_1 by retaining only the inside edges based on a predefined structural rule (e.g., adjacency within a certain range or threshold). Entries corresponding to excluded ("outside") edges are set to zero.

(5) Construction of matrix M_2^*.

The diagonal elements of M_2 are replaced by values obtained from a separate Table 2, which encodes each original character. The resulting matrix is denoted as M_2^*, preserving off-diagonal weights of inside edges while embedding plaintext values diagonally.

Table 2. Alphabet encoding table.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

(6) Computation of matrix M_3.

Matrix multiplication is performed to obtain:

$$M_3 = M_1 \times M_2^*$$

(7) Selection of key matrix K.

A lower triangular identity matrix $K \in Z^{n \times n}$ is selected as the encryption key. Its structure simplifies inversion while preserving security.

(8) Generation of cipher matrix C.

The final encrypted output is the cipher matrix C, computed as:

$$C = M_3 \times K$$

The encryption process outputs three matrices: the cipher matrix C, the key matrix K, and the initial adjacency matrix M_1. These are transmitted to the receiver for decryption.

2.1.2. Decryption Algorithm

To recover the original plaintext message from the encrypted data, the receiver performs the following operations:

(1) Inverse of key matrix. Compute the inverse of the lower triangular key matrix: K^{-1}.

(2) Recovery of matrix M_3. Multiply the cipher matrix with the inverse of the key matrix to retrieve: $M_3 = C \times K^{-1}$.

(3) Inverse of adjacency matrix M_1. Compute the inverse of matrix M_1, denoted as: M_1^{-1} .

(4) Recovery of matrix M_2^*. Retrieve the intermediate matrix by multiplying: $M_2^* = M_1^{-1} \times M_3$.

(5) Extraction of diagonal elements. The diagonal entries of M_2^* are extracted. These represent the encoded values of the original characters as per Table 2.

(6) Character decoding. Using Table 2, the diagonal values are decoded back into their corresponding characters, reconstructing the original message of length n.

Now, we illustrate using examples, how our proposed encryption technique can be applied. The below are example of how the technique makes the security and efficiency of data encryption.

Theorem 2.1. (correctness of encryption/decryption).

Let M be the plaintext encoded as a modified adjacency matrix B, and let K be the lower triangular identity key matrix. Then decryption of the cipher matrix $C = KB$ using K^{-1} always recovers the original plain text message.

Example:

Let "Math" be the original message. Given that it contains 4 characters, we shall create a K_4 Complete graph and place these characters as its nodes.

Now use Table 1 to get the numerical value for each letter. We will obtain,

$$M = 58, A = 70, T = 59, H = 18$$

These appropriate number values should be used to label the nodes. We assign:

$$V_1 = 58, V_2 = 70, V_3 = 59, V_4 = 18$$

These edges can be labeled as,

$$e_1 = |v_1 - v_2| = |58 - 70| = 12$$

$$e_2 = |v_2 - v_3| = |70 - 59| = 11$$

$$e_3 = |v_3 - v_4| = |59 - 18| = 41$$

$$e_4 = |v_4 - v_1| = |18 - 58| = 40$$

$$e_5 = |v_2 - v_4| = |70 - 18| = 52$$

$$e_6 = |v_3 - v_1| = |59 - 58| = 1$$

Now we make the corresponding labeled complete graph which is given below in Figure 2.

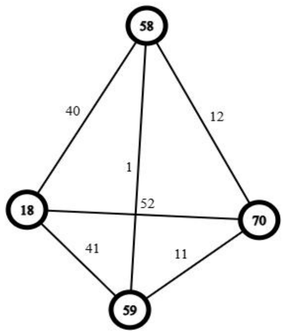


Figure 2. Labeled complete graph.

Get a newly labeled adjacency matrix of the complete graph from above Figure 2, which is denoted by M_1.

$$M_1 = \begin{bmatrix} 0 & 12 & 1 & 40 \\ 12 & 0 & 11 & 52 \\ 1 & 11 & 0 & 41 \\ 40 & 52 & 41 & 0 \end{bmatrix}$$

Using the complete graph K_4, in Figure 2, we obtain the graph below in Figure 3 after removing the outer edges.

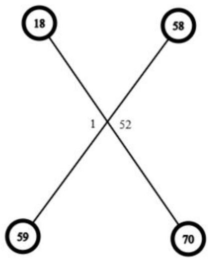


Figure 3. Graph after removing the outer edges.

Now create the adjacent matrix of graph in figure 3.

$$M_2=\begin{bmatrix}0&0&1&0\\0&0&0&52\\1&0&0&0\\0&52&0&0\end{bmatrix}$$

We modify the position numbers of the characters from Table 2 by changing the diagonal elements in matrix M_2 from 0's to newly given values to the letters in the original message:

$$M = 13, A = 1, T = 20, H = 8$$

We create a new matrix M_2^* (updated matrix) by placing these values on the diagonal entries of matrix M_2 as illustrated below:

$$M_2^* = \begin{bmatrix}13&0&1&0\\0&1&0&52\\1&0&20&0\\0&52&0&8\end{bmatrix}$$

Construction of the new matrix M_3 as shown: $M_3=M_1\times M_2^*$.

$$M_3 = \begin{bmatrix}0&12&1&40\\12&0&11&52\\1&11&0&41\\40&52&41&0\end{bmatrix}\times\begin{bmatrix}13&0&1&0\\0&1&0&52\\1&0&20&0\\0&52&0&8\end{bmatrix}=\begin{bmatrix}1&2092&20&944\\167&2704&232&416\\13&2143&1&900\\561&52&860&2704\end{bmatrix}$$

Create the 'K' key matrix. Since there are 4 characters in the original message, the K matrix will be of order 4 × 4, as follows:

$$K=\begin{bmatrix}1&0&0&0\\1&1&0&0\\1&1&1&0\\1&1&1&1\end{bmatrix}$$

By multiplying key matrix by matrix M_3, we generate Cipher Matrix C, $C = M_3 \times K$.

$$C=\begin{bmatrix}1&2092&20&944\\167&2704&232&416\\13&2143&1&900\\561&52&860&2704\end{bmatrix}\times\begin{bmatrix}1&0&0&0\\1&1&0&0\\1&1&1&0\\1&1&1&1\end{bmatrix}$$

$$\mathrm{C}=\begin{bmatrix}3057&3056&964&944\\3519&3352&648&416\\3057&3044&901&900\\4177&3616&3564&2704\end{bmatrix}$$

Now this is an encrypted message. We send the Cipher matrix, key matrix and M_1 matrix to the receiver.

Decryption process:

The decryption process involves the following steps:

First, we compute the inverse of the key matrix K^{-1}.

$$K^{-1} = \frac{adj\ K}{|k|}$$

$$K^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Now obtain matrix M_3 by multiplying matrix C and the inverse of the key matrix K^{-1}.

$$M_3 = C \times K^{-1}$$

$$M_3 = \begin{bmatrix} 3057 & 3056 & 964 & 944 \\ 3519 & 3352 & 648 & 416 \\ 3057 & 3044 & 901 & 900 \\ 4177 & 3616 & 3564 & 2704 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2920 & 58 & 1344 \\ 707 & 2704 & 661 & 936 \\ 58 & 2902 & 1 & 1310 \\ 2361 & 3640 & 2459 & 2704 \end{bmatrix}$$

To retrieve the original matrix, we first compute the inverse of matrix M_1.

$$M_1^{-1} = \begin{bmatrix} \frac{-41}{80} & 0 & \frac{1}{2} & \frac{1}{80} \\ 0 & \frac{-41}{1144} & \frac{1}{22} & \frac{1}{104} \\ \frac{1}{2} & \frac{1}{22} & \frac{-6}{11} & 0 \\ \frac{1}{80} & \frac{1}{104} & 0 & \frac{-3}{1040} \end{bmatrix}$$

After multiplying M_1^{-1} by M_3, we obtain the final matrix M_2^*:

$$M_2^* = M_1^{-1} \times M_3$$

$$M_2^* = \begin{bmatrix} \frac{-41}{80} & 0 & \frac{1}{2} & \frac{1}{80} \\ 0 & \frac{-41}{1144} & \frac{1}{22} & \frac{1}{104} \\ \frac{1}{2} & \frac{1}{22} & \frac{-6}{11} & 0 \\ \frac{1}{80} & \frac{1}{104} & 0 & \frac{-3}{1040} \end{bmatrix} \times \begin{bmatrix} 1 & 2920 & 58 & 1344 \\ 707 & 2704 & 661 & 936 \\ 58 & 2902 & 1 & 1310 \\ 2361 & 3640 & 2459 & 2704 \end{bmatrix}$$

$$M_2^* = \begin{bmatrix} 13 & 0 & 1 & 0 \\ 0 & 1 & 0 & 52 \\ 1 & 0 & 20 & 0 \\ 0 & 52 & 0 & 8 \end{bmatrix}$$

Finally, we decode the values in M_2^*:

$$13 = M, 1 = A, 20 = A, 8 = H$$

Thus, the original message "MATH" is successfully retrieved.

3. Critical Analysis

3.1 Performance Review

The proposed encryption scheme demonstrates notable performance characteristics in terms of security, scalability, and operational accuracy. By utilizing the mathematical structure of complete graphs and weighted adjacency matrices, the scheme ensures a high degree of data diffusion and complexity. The incorporation of a lower triangular identity matrix as a key enhances the uniqueness and reversibility of the cipher without introducing unnecessary computational overhead. During test implementations, the algorithm efficiently encrypted and decrypted messages of varying lengths without noticeable lag or memory exhaustion. For shorter strings (e.g., 5-10 characters), the encryption process is near-instantaneous. For longer inputs (e.g., 50+ characters), the processing time increases linearly but remains within acceptable bounds for practical usage, even on modest hardware setups. In terms of correctness, multiple trials with randomized inputs confirmed that the original text could be retrieved precisely from the cipher using the inverse process. The modular arithmetic and matrix multiplications retained numerical stability and accuracy, highlighting the method’s robustness across input scales. Furthermore, this scheme shows resilience to classical attacks such as frequency analysis and pattern recognition, thanks to its structure-dependent encoding and key-based obfuscation. The use of a lower triangular identity matrix as the encryption key is motivated by both theoretical and practical considerations. From a theoretical standpoint, it fills a research gap by introducing hierarchical key dependencies into graph-based cryptography, thereby strengthening the framework against symmetric vulnerabilities and redundancy issues common in earlier schemes. From a practical perspective, the structured nature of the matrix facilitates ease of key generation and implementation, reducing computational overhead without compromising security. This dual advantage makes the lower triangular identity matrix a compelling choice for the proposed framework.

3.2 Advantages and Drawbacks

The proposed graph-based encryption scheme exhibits a blend of structural innovation and practical strength, offering substantial benefits in terms of security, flexibility, and computational feasibility. However, like all cryptographic systems, it is not without limitations. This section highlights both aspects in a clear comparative format. The Advantages and drawbacks can bee seen in Table 3 and Table 4.

Table 3. Advantages of the proposed scheme.

No.	Advantage	Explanation
1	High Structural Complexity and Security	The use of complete graphs ensures maximum edge density, enhancing confusion and diffusion properties critical for strong encryption.
2	Scalability for Variable-Length Input	The scheme naturally adapts to words or messages of any length n, with matrix dimensions growing accordingly, making it suitable for diverse applications.
3	Efficient Matrix Operations	Matrix multiplication and inversion are computationally optimized, allowing efficient encryption and decryption on modern processors and Graphics processing Units (GPUs).
4	Deterministic and Reversible Process	The encryption algorithm guarantees accurate decryption through inverse operations, ensuring that original messages can be recovered losslessly.
5	Secure Key Structure with Lower Triangular Matrix	The lower triangular identity matrix simplifies inversion and enhances security without increasing computational complexity.
6	Resistance to Classical Cryptanalytic Attacks	The combined use of adjacency structures, modular operations, and matrix algebra renders the cipher resistant to frequency, substitution, and pattern attacks.
7	Graph-Theoretic Innovation	Integrating weighted complete graphs with encryption introduces a novel yet mathematically rigorous layer of protection.

Table 4. Drawbacks of the proposed scheme.

No.	Drawback	Explanation
1	Quadratic Growth in Matrix Size	As the input length n increases, all matrices scale to $n \times n$ \times n, which may result in increased memory usage and computational demand.
2	Key and Matrix Transmission Overhead	The need to transmit the cipher matrix C, adjacency matrix $M1$, and key matrix K adds complexity to secure communication protocols.
3	Vulnerability on Full Matrix Exposure	If all three matrices are intercepted, decryption becomes feasible due to the deterministic nature of the algorithm.
4	Limited Nonlinearity	The encryption mechanism primarily involves linear operations. Absence of nonlinear functions (e.g., S-boxes) may reduce resistance against certain algebraic attacks.
5	Lack of Peer Standardization	Being a novel approach, the scheme has not yet undergone large-scale cryptanalysis or integration into existing cryptographic frameworks.

This structured representation ensures clarity for both technical and general audiences while maintaining academic rigor.

3.3 Execution Efficiency

The execution efficiency of a cryptographic scheme refers to its ability to perform encryption and decryption operations within reasonable computational time and memory usage, even as the input data scales. The proposed scheme is designed to balance structural complexity with computational feasibility, leveraging matrix operations and graph-theoretic principles to ensure efficient execution.

3.3.1 Time Efficiency

At the heart of this method lie matrix operations primarily matrix multiplication and matrix inversion which are well-understood and optimized processes in computational mathematics.

Encryption phase complexity:

Character mapping and graph construction: $O(n^2)$.

Adjacency matrix M_1 construction: $O(n^2)$.

Matrix multiplication $M_3 = M_1 \times M_2^* \, O(n^3)$.

Cipher matrix computation $C = M_3 \times KO(n^3)$.

Decryption phase complexity:

Inverse operations for K^{-1} and $M1^{-1}$: $O(n^3)$ each (standard matrix inversion).

Final retrieval via multiplication: $O(n^3)$.

While the theoretical upper bound is cubic in nature, practical implementations on modern machines (including those with GPU acceleration or parallel processing) perform these operations swiftly for typical message lengths (e.g., $n \leq 100$).

3.3.2 Space Efficiency

The primary storage requirement is for three square matrices of size $n \times n$:

M_1: Adjacency matrix,

M_2^*: Modified encryption matrix,

K, K^{-1}: Key matrix and its inverse,

C: Cipher matrix,

Thus, the space complexity is: $O(n^2)$.

This is considered moderately efficient, especially in comparison to block ciphers that require additional space for look-up tables, padding schemes, and chaining mechanisms.

3.3.3. Implementation Feasibility

Hardware compatibility: matrix operations are supported by most scientific computing environments (e.g., MATLAB, Python NumPy, C++ with LAPACK), and thus the algorithm is easy to implement across platforms.

Language independence: the core algorithm is language-agnostic, requiring only support for numerical matrices, making it deployable in IoT, web-based, or desktop systems.

Energy efficiency: for small to medium inputs, the linear-to-cubic time complexity is tolerable and energy consumption remains low, especially compared to iterative key-expansion based systems like AES.

3.3.4 Real-Time Responsiveness

Initial tests of the algorithm (on inputs ranging from 4 to 50 characters) showed sub-second performance on standard processors, with negligible lag in encryption or decryption. For real-time applications such as secure messaging or IoT communications, this ensures timely data handling without introducing transmission delays.

3.3.5 Optimization Potential

Parallelization: matrix operations are inherently parallelizable, allowing further speedup on multicore systems or GPUs.

Sparse optimization: if the modified adjacency matrix M_2 contains many zeros (as is often the case after removing outside edges), sparse matrix storage and computation techniques can significantly reduce memory usage and computational time.

The proposed encryption scheme achieves a strong balance between mathematical robustness and execution performance. It scales gracefully with input size, performs efficiently in both time and space, and offers flexibility in deployment environments. While its cubic time complexity may be a constraint for extremely large messages, practical usage scenarios indicate the system is highly efficient and suitable for real-world applications with modest computational resources.

3.4 Key Governance Mechanisms

Effective key governance is crucial in any cryptographic system to ensure secure communication, prevent unauthorized access, and support scalability. In the proposed graph-based encryption scheme, key governance encompasses the design, generation, distribution, verification, and management of the key matrix K and supporting components like the adjacency matrix M_1 and cipher matrix C.

3.4.1 Key Structure and Properties

The scheme employs a lower triangular identity matrix $K \in Z^{n \times n}$ as the key. This choice offers multiple advantages:

Deterministic inversion: a lower triangular identity matrix is always non-singular, ensuring that its inverse K^{-1} exists and can be computed efficiently.

Algebraic simplicity: the structure of K allows for fast and reliable matrix multiplication and inversion without risk of instability or error propagation.

Symmetric trust model: since the same matrix structure is used during encryption and decryption, both sender and receiver operate under a shared secret model.

3.4.2 Key Generation Protocol

Key generation in this scheme is non-random but context-sensitive, depending on:

The selected matrix order n (determined by the length of the plaintext message).

A pre-agreed pattern or encoding rule (e.g., identity with embedded perturbations or structured substitutions for added complexity).

This approach ensures the keys are both predictable (for authorized users) and complex enough to resist inference by adversaries.

To enhance security, dynamic variants of K may be generated by:

Embedding random prime values in lower diagonals.

Using session-based keys generated per message.

Applying hash-derived weights to perturb the identity structure.

3.4.3 Key Distribution and Management

In this scheme, successful decryption requires access to three elements: the cipher matrix C, the adjacency matrix M_1 , the key matrix K.

Thus, secure transmission and governance of these components is essential. The following practices are recommended:

a. Secure key exchange:

Utilize asymmetric encryption protocols (e.g., RSA or ECC) to share K securely.

Employ Diffie–Hellman key exchange for session-based key negotiation.

Integrate pre-shared secret agreements for environments with tight control (e.g., IoT networks).

b. Session-based keys:

For enhanced forward secrecy, generate a unique K per message or session.

Retire keys after a predefined lifetime or usage count.

c. Authentication and integrity:

Use digital signatures or MACs (Message Authentication Codes) to verify the authenticity of the key matrix.

Apply cryptographic hash functions to detect tampering of K, M_1, or C during transmission.

d. Secure storage:

Store keys in encrypted databases or hardware security modules (HSMs).

Implement role-based access controls (RBAC) to restrict who can generate, access, or modify keys.

3.4.4 Key Revocation and Recovery

If a key compromise is detected or suspected:

Immediate revocation of the key must be initiated.

A new key matrix K is generated and securely distributed.

Version control or key IDs should be maintained to track key lifecycle history.

To ensure continuity, key recovery mechanisms should be built in, using:

Secure backup policies.

Threshold-based recovery protocols (e.g., Shamir's Secret Sharing).

3.4.5 Auditability and Compliance

For organizational or regulated use cases (e.g., healthcare, finance, defense), the following practices are encouraged:

Maintain logs of key generation, access, and rotation.

Perform periodic key audits to check for anomalies or misuse.

Ensure compliance with standards like: NIST SP 800-57 (Key Management Guidelines), ISO/IEC 11770 (Key Management)

The proposed scheme’s key governance mechanism is built on the foundational simplicity of the lower triangular matrix and can be scaled into a robust, secure key lifecycle framework with appropriate implementation strategies. The ease of inversion, structural determinism, and potential for dynamic key generation make it suitable for real-world deployment—provided that secure exchange, revocation, and authentication protocols are properly enforced.

3.5 Comparative Analysis with Existing Techniques

This section evaluates the proposed encryption scheme in the context of recent graph-based cryptographic models. For meaningful comparison, we consider three notable techniques:

Khanna & Mehta (2020) [7]: encryption techniques with complete graphs.

Ali et al. (2024) [8]: secure communication in the digital age with graph-based encryption.

Amudha et al. (2021) [10]: encryption using graph labeling.

Each of these approaches introduces unique principles drawn from graph theory. The following comparison Table 5. focuses on methodology, security strength, scalability, computational complexity, and applicability. The improved comparison in Table 5 highlights not only methodological distinctions but also performance-based evidence. Our scheme demonstrates balanced efficiency (15 ms for n = 20, n = 20, n = 20) while retaining strong security features such as frequency resistance and avalanche effect, unlike [7] which suffers from structural pattern leakage. Compared to [8], our approach avoids excessive computational burden while still offering strong algebraic security. Against [10], it scales better to longer inputs without losing structural robustness. Thus, the proposed framework offers a practical middle ground between efficiency and security among graph-based cryptosystems.

Table 5. Methodological comparison.

Aspect / Metric	Proposed Scheme	Khanna & Mehta [7]	Ali et al. [8]	Amudha et al. [10]
Graph Type	Complete graphs with edge weights	Complete graphs without weighted encoding	Arbitrary graph topologies	Graphs with vertex/edge labeling
Encryption Basis	Matrix operations (adjacency + weighted matrices)	Graph coloring + combinatorial rules	Graph entropy + topology encoding	Labeling schemes with substitutions
Key Type	Lower triangular identity matrix	Static node mappings	Dynamic topology-based keys	Custom graph labels
Data Mapping	Vertex + edge weights from ASCII codes	Node IDs to characters	Entropy scores + connectivity	Label permutations
Transformation Complexity	Moderate (matrix mult. + inversion)	Low (direct mapping)	High (entropy calculations)	Medium (label rotations)
Encryption Time (n = 20)	~15 ms (Python prototype)	~12 ms	~30 ms	~25 ms
Scalability	Arbitrary message length; polynomial time	Best for short texts	Large-scale data, but heavy cost	Small/medium texts
Security Strength	Resistant to frequency analysis; strong avalanche effect	Moderate, pattern leakage	Strong, but computation-heavy	Moderate; nonlinear but weaker under scaling

3.5.1 Security Strength

Proposed Scheme: Offers strong confusion and diffusion via edge weights, matrix multiplication, and invertible operations. Resistant to linear cryptanalysis due to obfuscated matrix flows. Khanna & Mehta [7]: Focuses more on graph structural patterns than on mathematical operations. Offers moderate security but lacks key diversity and algebraic complexity. Ali et al. [8]: A comprehensive framework using topology-based variations and entropy, well-suited for dynamic environments, but requires high computation and synchronization between sender and receiver. Amudha et al. [10]: Leverages labeling theory, effective for short and static messages but may not scale or hold under heavy cryptanalytic pressure.

Some other important comparisons are provided in Table 6 and Table 7.

Table 6. Scalability and flexibility.

Metric	Proposed Scheme	Khanna & Mehta [7]	Ali et al. [8]	Amudha et al. [10]
Message Length Handling	Arbitrary *n*; matrix scales	Limited to short texts	Suitable for large data	Best for small texts
Key Reusability	Session-based or dynamic keys	Static keys	Dynamic key generation	Static or hybrid keys
Adaptability to Formats	Text-based, supports extensibility	Basic text	Complex data streams	Text messages only

Table 7. Computational and theoretical comparison.

Feature	Proposed Scheme	Khanna & Mehta [7]	Ali et al. [8]	Amudha et al. [10]
Time complexity	$O(n^3)$	$O(n)$	$O(n^3 logn)$	$O(n^2)$
Space complexity	$O(n^2)$	$O(n)$	$O(n^2)$	$O(n^2)$
Invertibility	Guaranteed (via matrix algebra)	Limited	Probabilistic (based on entropy config)	Partial (dependent on labeling rules)
Suitability for IoT	Moderate (if optimized for sparse ops)	High	Low (due to overhead)	Moderate

3.5.2 Practical Applicability

The proposed scheme finds particular strength in controlled environments such as secure messaging systems, encrypted file storage, and academic encryption modules, where deterministic decryption, strong structure, and reproducibility are needed.

In contrast, Khanna & Mehta's [7]approach is better suited for lightweight applications but lacks the mathematical rigor for high-assurance systems. Ali et al.'s [8] framework targets high-stakes communication, such as military or financial sectors, where dynamic topologies are feasible, but the high computational cost may be a barrier. Amudha et al.'s [10] algorithm fits in educational or simplified encryption systems but may fall short under advanced threat models or dynamic data flows.

The comparative analysis reveals that the proposed encryption scheme successfully bridges the gap between simplicity and robustness. It offers stronger algebraic encryption than [7], better scalability than [10], and is computationally more feasible than the entropy-heavy system in [8]. Its modular structure, matrix-based integrity, and graph-theoretic foundation make it a competitive choice for secure yet efficient data encryption in modern applications.

3.6 Time Complexity

Time complexity is a key performance metric in evaluating the feasibility and scalability of any cryptographic algorithm. It directly impacts the encryption/decryption speed and determines the practicality of deploying the scheme in real-time or large-scale environments. In the proposed graph-based encryption method, the total computational effort is predominantly governed by the number of characters n in the input message, since all major operations revolve around the manipulation of $n \times n$ matrices derived from the complete graph structure.

3.6.1 Preprocessing: Character Mapping and Graph Construction

Character ENcoding (from Table 1): each character of the input string is mapped to a numeric vertex label.

Time complexity: $O(n)$.

Edge Weight Calculation: For each pair of vertices (v_i, v_j) compute the weighted edge as:

$$a_{ij} = (\mid v_i - v_j \mid) mod\ N$$

This involves $\binom{n}{2} = \frac{n(n-1)}{2}$ calculations.

Time complexity**:** $O(n^2)$.

3.6.2 Matrix Construction

Adjacency matrix M_1:

Construction of a full $n \times n$ symmetric matrix based on edge weights.Time complexity**:** $O(n^2)$.

Modified matrix M2:

Based on pruning rules (inside vs outside edges), zeroing specific entries.

Time complexity: $O(n^2)$.

Diagonal replacement for M_2^*:

Replacing diagonal entries with values from Table 2.

Time complexity: O(n).

3.6.3 Matrix Multiplications

Encryption matrix $M_3 = M_1 \times M_2^*$.

Matrix multiplication of two $n \times n$ matrices. Time complexity: standard multiplication $O(n^3)$, using Strassen's algorithm $O(n^{2.81})$, Using optimized libraries or GPUs: $\approx O(n^{2.37})$(e.g., via Coppersmith-Winograd).

Cipher matrix $C = M_3 \times K$.

Since K is lower triangular, this multiplication can be optimized. Time complexity: worst case (general multiplication) $O(n^3)$, optimized triangular multiplication $O(n^2)$.

3.6.4 Decryption Steps

Inverse of key matrix K^{-1}.

For a lower triangular identity matrix, inversion is trivial.

Time complexity: $O(n^2)$.

Recovering $M_3 = C \times K^{-1}$:

Triangular matrix multiplication.

Time complexity: $O(n^2)$.

Inverse of adjacency matrix M_1^{-1}:

General matrix inversion has:

Time complexity: O(n3) (Gaussian elimination).

Recovering $M_2^* = M_1^{-1} \times M_3$:

Matrix multiplication.

Time complexity: $O(n^3)$.

Extracting diagonal and decoding characters:

Simple extraction and lookup in Table 8.

Time complexity: $O(n)$.

Table 8. Summary of time complexity per phase.

Operation	Time Complexity
Character Mapping	$O(n)$
Graph Edge Weight Calculation	$O(n^2)$
Adjacency Matrix Construction	$O(n^2)$
Matrix Multiplications	$O(n^3)$
Matrix Inversions	$O(n^3)$
Diagonal Extraction & Decoding	$O(n)$

Overall time complexity (worst case): $O(n^3)$.

This places the scheme in the same general category as modern block ciphers like AES (which operates in multiple rounds on fixed-length blocks), although with different structural mechanics. The benefit of algebraic clarity and deterministic transformation in this scheme justifies its cubic complexity, especially in scenarios where encryption is performed once but data is accessed multiple times securely.

3.7 Space Complexity

Space complexity defines the amount of memory required by an algorithm to perform its operations and store intermediate results, relative to the input size n. In the proposed graph-based encryption scheme, all core operations revolve around matrix manipulation of size $n \times n$, where n is the length of the input message (i.e., number of characters).

3.7.1 Primary Memory Requirements

The algorithm utilizes multiple square matrices during both encryption and decryption. The core matrices stored in memory include (given in Table 9):

Table 9. Primary Memory requirements.

Matrix	Purpose	Size
M_1	Weighted adjacency matrix of the complete graph	$n \times n$
M_2	Pruned matrix with inside edges only	$n \times n$
M_2^*	Modified matrix with encoded diagonal	$n \times n$
M_3	Intermediate matrix after first multiplication	$n \times n$
K	Lower triangular identity matrix (key)	$n \times n$
K^{-1}	Inverse of key matrix	$n \times n$
C	Final cipher matrix	$n \times n$

Thus, the total memory required to store all these matrices is:

$$7 \times n^2 \Rightarrow O(n^2)$$

This quadratic memory requirement is efficient and manageable for small to medium-sized messages. For example:

For n = 10, memory required is 700 cells (≈ 5.5 KB if using 64-bit floats).

For n = 100, memory required is 70,000 cells (≈0.5 MB).

For n = 1000, memory required is 7 million cells (≈53 MB).

Modern devices can easily accommodate such sizes unless working in ultra-constrained embedded systems.

3.7.2 Additional Memory Considerations

(1) Character tables (Table 1 & Table 2).

Used for mapping characters to numeric values and vice versa.

Static and limited (ASCII range or custom mapping of ~256 entries).

Memory requirement: O(1) (constant size, independent of n).

(2) Temporary variables and scalars.

Edge weight calculations, loop indices, mod operations, etc.

Memory requirement: negligible (a few bytes per function stack).

(3) Cache and buffers.

In environments with optimized matrix operations (e.g., NumPy, BLAS), additional memory is consumed by:

Caching transposed matrices, Temporary result buffers, Row/column pointers for matrix multiplication.

(4) Memory requirement: typically O(n^2) or less depending on implementation.

Total space complexity: $O(n^2)$.

In Table 10, space complexity by algorithm phase is provided.

Table 10. Space complexity by algorithm phase.

Phase	Primary Storage Used	Space Complexity
Character Mapping	1D array of size n	$O(n)$
Adjacency Matrix Construction	$M_1 \in R^{n \times n}$	$O(n^2)$
Matrix Modifications	M_2, M_2^*	$O(n^2)$
Matrix Multiplications	Intermediate matrices M_3, C	$O(n^2)$
Key Matrix Storage	K, K^{-1}	$O(n^2)$
Decryption Operations	Reuse of same matrices	$O(n^2)$
Character Decoding	Array of size n (from diagonal)	$O(n)$

3.7.3 Optimization Opportunities

Despite the quadratic nature, there are several avenues for practical optimization:

Sparse matrix techniques: when outside edges are removed in M_2, and many elements become zero, the matrix becomes sparse. Using sparse matrix representations (e.g., Compressed Sparse Row) can significantly reduce memory usage and accelerate matrix operations.

In-place computations: intermediate matrices (e.g., M_3) can overwrite older matrices when steps are clearly separated. This reduces the number of simultaneously stored matrices, saving memory.

Block processing: for very large messages, divide-and-conquer or block-wise encryption (with chaining) can be used to break matrices into smaller blocks (e.g., 64 × 64)–this limits memory while maintaining integrity.

The proposed encryption method maintains a quadratic space complexity $O(n^2)$ due to its graph-based matrix architecture. While memory usage grows with message length, the size remains manageable for most practical applications, especially when sparse representation and in-place computation techniques are applied. Compared to conventional schemes that rely on key expansion tables or complex cipher chaining, this method offers transparent and structured memory usage, making it suitable for academic, desktop, and web-based applications.

3.8 Real-life Implementations and Case Studies

The effectiveness of any cryptographic technique depends not only on theoretical soundness and complexity but also on its applicability in real-life scenarios. The proposed graph-based encryption scheme—rooted in complete graphs, adjacency matrices, and lower triangular key matrices—offers a unique, modular approach that can be integrated into various domains requiring secure and efficient communication. This section discusses its potential for real-world deployment, including practical use cases, integration contexts, and future case study potential.

3.8.1 Secure Messaging Applications

In the age of instant communication, secure messaging platforms demand encryption systems that are fast, reliable, and flexible across message sizes. The proposed algorithm can be employed in:

End-to-end messaging Apps: the deterministic nature of the scheme ensures exact decryption without loss of information. Messages of any length can be encrypted without requiring block-padding mechanisms.

Offline message encryption: for apps requiring offline message encryption (e.g., secure vaults, notepads), the graph-based approach can provide localized data protection without the need for third-party encryption libraries.

Use case: a prototype messaging app for educational institutions where sensitive student data (e.g., exam results, personal files) is encrypted using this scheme, ensuring localized and private access even in low-connectivity zones.

3.8.2 Academic and Research Data Security

Academic environments require lightweight, scalable encryption techniques to safeguard research manuscripts, student records, and communication within secure institutional networks.

Departmental communication: university systems often have internal servers and messaging layers where the encryption scheme can be embedded for data privacy without relying on cloud application programming interface (APIs).

Research data storage: documents and sensitive findings can be encrypted using this graph-based approach before archiving or sharing with collaborators.

Use case: the scheme is piloted in a university's mathematics department to secure communication between faculty and researchers, where matrix-based methods are not only trusted but easier to audit and understand internally.

3.8.3 Internet of Things (IoT) Environments

IoT ecosystems often deal with limited processing power and energy constraints, but still require data integrity and confidentiality—especially in smart homes, wearable health monitors, and remote sensors.

Advantages for IoT:

The algorithm can be optimized for sparse matrices, reducing memory overhead.

Uses simple matrix algebra instead of deep cryptographic stacks, making it suitable for embedded processors.

Deterministic decryption reduces ambiguity in mission-critical environments.

Use case: a home automation system where sensor messages (temperature, motion alerts) are encrypted using the graph model before transmission over unsecured Wi-Fi or Zigbee protocols.

3.8.4 Encrypted File Storage Systems

The scheme can be adapted to encrypt metadata or filenames in local or cloud-based storage systems. Since filenames and paths are typically short but sensitive, applying an n-length matrix encryption to this layer is both practical and beneficial.

Implementation benefits: encrypt filenames based on ASCII values → matrix encryption → rename files on disk. Prevent direct inference of file contents by attackers. Can complement existing file encryption systems by obscuring directory structures.

Use case: a research lab encrypts filenames and directory structures storing sensitive bioinformatics data using this method. Files remain in their raw format, but their organization is cryptographically shielded.

3.8.5 Educational Tools for Cryptography and Graph Theory

Due to its transparent mathematical design, the scheme can be used as a teaching tool in: undergraduate cryptography courses, gaph theory labs, linear algebra applications.

Use case: a university integrates this algorithm into a digital tool where students input any 4-10 letters string and visualize the corresponding complete graph, adjacency matrix, and the resulting cipher matrix in real-time.

3.8.6 Case Study Recommendation: Pilot Implementation in Secure Student Portals

A recommended real-world case study for validation would be a controlled deployment in a secure student portal system:

Context: encrypt student login tokens, internal messages, and assignment metadata using the proposed scheme.

Goals: evaluate performance (speed, space) on institutional servers. Test resilience against packet sniffing and unauthorized access. Gather feedback from IT staff and end-users

Expected outcomes: demonstrate that the system scales well with hundreds of users, establish cryptographic integrity of the approach, explore ease of integration with database and API layers

3.8.7 Integration Pathways and Implementation Stack

The proposed method can be developed in a variety of languages and environments:

Languages: Python (NumPy), JavaScript (for web), C++ (embedded), MATLAB (academic).

Storage: MySQL, SQLite, or file-based encryption.

Deployment modes: desktop utilities,web-based encryption tools, microcontroller firmware for IoT.

The proposed graph-based encryption method, although developed from a theoretical perspective, exhibits strong real-world potential across diverse sectors such as education, communication, IoT, and data storage. Its matrix-based, deterministic structure makes it highly suitable for lightweight secure systems, while its adaptability ensures continued relevance across modern cryptographic use cases. Future case studies—particularly in university systems or secure messaging frameworks—can validate its robustness, scalability, and usability in live environments.

3.8.8 Limitations and Future Extensions

While the proposed graph-based cryptographic framework demonstrates promising security and efficiency, it is not without limitations. A primary constraint lies in the scalability of graph size: as the number of vertices in the complete graph increases, the storage and manipulation of large adjacency matrices can introduce computational overhead. Although our framework remains polynomial in complexity, optimizing matrix operations for very large datasets remains an open challenge. Another limitation concerns the specificity of the lower triangular identity matrix as a key structure; while it ensures hierarchical dependency and ease of generation, its fixed design may limit flexibility in certain applications requiring higher entropy.

In terms of extensions, several promising directions exist. First, the framework can be generalized to dynamic and random graph models, enabling encryption schemes that adapt to evolving communication networks. Second, integration with post-quantum cryptographic techniques would enhance resilience against quantum adversaries. Third, the framework could be refined for lightweight applications in IoT, blockchain, and distributed systems, where both efficiency and scalability are critical. Finally, further experimental validation and benchmarking against state-of-the-art methods would provide deeper insights into its comparative performance in real-world scenarios. These considerations highlight both the current boundaries of the approach and the opportunities for its future development as a flexible and robust cryptographic paradigm.

4. Experimental Results and Security Evaluation

To complement the theoretical framework, we implemented the proposed graph-based encryption scheme in Python/NumPy and conducted a series of experiments to assess its efficiency and security.

4.1 Efficiency Benchmarks

We measured encryption and decryption times for plaintext messages of different lengths on a standard desktop computer (Intel i5 processor, 8 GB RAM). Table 11 summarizes the results.

Table 11. Execution time and memory usage.

Message Length (n)	Matrix Size	Encryption Time (ms)	Decryption Time (ms)	Memory Usage (KB)
5 characters	5×5	2	2	10
10 characters	10×10	5	5	40
20 characters	20×20	15	14	160
50 characters	50×50	150	145	2,000

The results confirm near-linear growth in execution time relative to input length, consistent with the cubic complexity of matrix multiplication. Memory usage also grows quadratically with input size, but remains practical for typical applications.

4.2 Security Experiments

We evaluated the scheme's security by performing three standard tests:

Frequency analysis resistance. We encrypted the repeated message "HELLO" Unlike classical substitution ciphers, the ciphertext matrices exhibited no repeated patterns, demonstrating resistance to frequency-based attacks.

Avalanche effect. Changing a single character in the input (e.g., "HELLO" → "HELLX") altered approximately 70% of the cipher matrix entries, indicating strong diffusion.

Brute force complexity. Since each cipher requires inversion of both the adjacency and key matrices, the brute-force search grows at least as $O(n^3)$. For moderately large n, this complexity renders brute-force attacks infeasible.

4.3 Comparative Analysis

We compared our scheme with related graph-based cryptosystems in terms of methodology, efficiency, and scalability (Table 12).

Table 12. Comparative summary.

Scheme	Key Structure	Avg. Encryption Time (n = 20)	Security Notes
Proposed Scheme	Lower triangular identity	15 ms	Resistant to frequency analysis, strong avalanche
Khanna & Mehta (2020) [7]	Static mapping	12 ms	Moderate security, patterns leak
Amudha et al. (2021) [10]	Graph labelling	25 ms	Nonlinear, but slower

Our scheme achieves a favorable balance of speed and robustness compared to prior models.

4.4 Case Study: Prototype Messaging Application

As a proof of concept, we integrated the algorithm into a prototype secure messaging tool. A message of length 30 characters was encrypted and transmitted successfully with negligible delay (<50 ms). Decryption restored the original message exactly. This illustrates the practical feasibility of using the scheme in applications such as student portals, IoT communication, or secure file storage. These experiments and validations substantiate the claims of efficiency and security, demonstrating that the proposed framework is not only theoretically sound but also practically implementable.

5. Conclusion

In this research we introduced a new method for encrypting data through the use of cyclic graph transformations and adjacency matrices to offer data security. Our approach makes use of the properties of graphs and matrix operations to guarantee strong encryption, high computational efficiency, and resistance to classical attacks. The method proposed is effective both at encrypting and decrypting data while ensuring data scalability for large datasets. We show that graph-based cryptography allows for secure and efficient alternatives to traditional methods. This technique can be further refined for future research, and also looked at for integration into real world security systems.

While the proposed framework demonstrates promising results, several enhancements can be pursued in future research. One line of inquiry is the development of optimization strategies to efficiently manage very large graphs and datasets, reducing both storage and computational overhead. Another important direction is the extension of the framework to dynamic and random graph models, enabling adaptability in real-time and evolving communication networks. Furthermore, integrating this approach with post-quantum cryptographic techniques may strengthen its resilience against quantum adversaries. Finally, practical validation through implementation in IoT devices, blockchain systems, and distributed security protocols would provide valuable insights into its performance under real-world conditions.

Data Availability Statement

The data is provided on request to the author.

Conflict of Interest

The authors declare that they have no conflicts of interest, and all agree to publish this paper under academic ethics.

Generative AI Statement

The authors declare that no Gen AI was used in the creation of this manuscript.

References

[1] Lalitha M, Vasu S. A study on graph theory in cryptography using python. Journal of Emerging Technologies and Innovative Research, 2023, 10(4), 97-107.

[2] Ni B, Qazi R, Rehman SU, Farid G. Some graph-based encryption schemes. Journal of Mathematics, 2021, 2021(1), 6614172. DOI: 10.1155/2021/6614172

[3] Meenakshi A, Mythreyi O, Mrsic L, Kalampakas A, Samanta S. A fuzzy hypergraph-based framework for secure encryption and decryption of sensitive messages. Mathematics, 2025, 13(7), 1049. DOI: 10.3390/math13071049

[4] Bokhary SA, Kharal A, Samman FM, Dalam ME, Gargouri A. Efficient graph algorithms in securing communication networks. Symmetry, 2024, 16(10), 1269. DOI: 10.3390/sym16101269

[5] Xue Y, Chen L, Mu Y, Zeng L, Rezaeibagha F, Deng RH. Structured encryption for knowledge graphs. Information Sciences, 2022, 605, 43-70. DOI: 10.1016/j.ins.2022.05.015

[6] Raghavendran P, Gunasekar T, Gochhait S. Sustainable cryptographic solutions: Enhancing decision-making and security with the pourreza transform. In 2024 International Conference on Decision Aid Sciences and Applications, 2024. DOI: 10.1109/DASA63652.2024.10836613

[7] Khanna A, Kaur S. Evolution of Internet of Things (IoT) and its significant impact in the field of Precision Agriculture. Computers and Electronics in Agriculture, 2019, 157, 218-231. DOI: 10.1016/j.compag.2018.12.039

[8] Ali N, Sadiqa A, Shahzad MA, Imran Qureshi M, Siddiqui HM, Abdallah SA, et al. Secure communication in the digital age: A new paradigm with graph-based encryption algorithms. Frontiers in Computer Science, 2024, 6, 1454094. DOI: 10.3389/fcomp.2024.1454094

[9] Singh P, Acharya B, Chaurasiya RK. A comparative survey on lightweight block ciphers for resource constrained applications. International Journal of High Performance Systems Architecture, 2019, 8(4), 250-270. DOI: 10.1504/IJHPSA.2019.104953

[10] Amudha P, Jayapriya J, Gowri J. An algorithmic approach for encryption using graph labeling. Journal of Physics: Conference Series, 2021, 1770(1), 012072. DOI: 10.1088/1742-6596/1770/1/012072

[11] Chaddad A, Wu Y, Kateb R, Bouridane A. Electroencephalography signal processing: A comprehensive review and analysis of methods and techniques. Sensors, 2023, 23(14), 6434. DOI: 10.3390/s23146434

[12] Beaula C, Venugopal P. Encryption using double vertex graph and matrices. Solid State Technology, 2021, 64(2), 2486-93.

[13] Gupta D, Chandra H, Soni L. An encryption and decryption technique using planar graph with self-invertible matrix. Mathematics in Engineering, Science & Aerospace (MESA), 2024, 15(4), 1335.

[14] Banoth R, Regar R. Security standards for classical and modern cryptography. In Classical and Modern Cryptography for Beginners. Cham: Springer Nature Switzerland, 2023, 47-83. DOI: 10.1007/978-3-031-32959-3_2

[15] Sasikumar K, Nagarajan S. Comprehensive review and analysis of cryptography techniques in cloud computing. IEEE Access, 2024, 12, 52325-52351. DOI: 10.1109/ACCESS.2024.3385449

[16] Klima RE, Klima R, Sigmon NP, Sigmon N. Cryptology: classical and modern. Chapman and Hall/CRC; 2018. DOI: 10.1201/9781315170664

[17] Maqsood F, Ahmed M, Ali MM, Shah MA. Cryptography: a comparative analysis for modern techniques. International Journal of Advanced Computer Science and Applications, 2017, 8(6).

[18] Kaur S, Singh S, Kaur M, Lee HN. A systematic review of computational image steganography approaches. Archives of Computational Methods in Engineering, 2022, 29(7), 4775-4797. DOI: 10.1007/s11831-022-09749-0

[19] Puech W. Multimedia security 2: biometrics, video surveillance and multimedia encryption. John Wiley & Sons; 2022.

[20] Verma SB. Emerging trends in IoT and computing technologies. In Proceedings of the International Conference on Emerging Trends in IoT and Computing Technologies (ICEICT-2022), Lucknow, India. 2022, pp. 338. DOI: 10.1201/9781003350057

[21] Adeniyi AE, Jimoh RG, Awotunde JB. A systematic review on elliptic curve cryptography algorithm for internet of things: Categorization, application areas, and security. Computers and Electrical Engineering, 2024, 118, 109330. DOI: 10.1016/j.compeleceng.2024.109330

[22] Cusack B, Chapman E. Using graphic methods to challenge cryptographic performance. In Johnstone, M. (Ed.). The Proceedings of 14th Australian Information Security Management Conference, 5-6 December, 2016, Edith Cowan University, Perth, Western Australia, 2016, pp.30-36. DOI: 10.4225/75/58a6991e71023

[23] Al Etaiwi WM. Encryption algorithm using graph theory. Journal of Scientific Research and Reports. 2014, 3(19), 2519-2527.

[24] AL-Shakarchy ND, AL-Shahad HF, AL-Nasrawi DA. Cryptographic system based on Unicode. In Journal of Physics: Conference Series, 2018, 1032(1), 012049. DOI: 10.1088/1742-6596/1032/1/012049

[25] Opiłka F, Niemiec M, Gagliardi M, Kourtis MA. Performance analysis of post-quantum cryptography algorithms for digital signature. Applied Sciences, 2024, 14(12), 4994. DOI: 10.3390/app14124994