Int. J. Anal. Appl. (2025), 23:312

International Journal of Analysis and Applications

A Lightweight Symmetric Encryption Framework Using Homogeneous and Non-Homogeneous Caterpillar Graphs

D. Gomathi[1], **Sivakumar Nagarajan**[2,*]

[1]*Department of Mathematics, School of Advanced Sciences,Vellore Institute of Technology, Vellore, Tamil Nadu, India*

[2]*Department of Information Security, School of Computer science and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India*

[*]*Corresponding author:* nsivakumar@vit.ac.in

Abstract. In today's interconnected world, ensuring secure and efficient communication is of critical importance. Traditional cryptographic techniques often encounter challenges such as high computational costs and vulnerabilities to emerging attack strategies. This study proposes a novel encryption framework that leverages the structural properties of homogeneous and non-homogeneous caterpillar graphs to enhance the processes of encrypting and decrypting textual information. Plaintext characters are first mapped to numerical values based on their positions in the English alphabet, after which number-theoretic operations are applied to generate ciphertext. The encrypted values are embedded within caterpillar graph structures, where vertex assignments and coloring methods introduce additional layers of complexity. This integration not only increases resistance to brute-force and quantum-based attacks but also improves visualization and segmentation of encrypted blocks. Furthermore, efficient graph traversal algorithms are incorporated to optimize computational performance. The proposed framework significantly strengthens cryptographic security by combining graph theory, number theory, and coloring techniques, offering a scalable solution to modern cybersecurity challenges.

1. Introduction

The proliferation of Internet usage across various social sectors has forced internet service providers to adapt their offerings to facilitate remote access for professionals, collaborators, and general users. Although this pervasive connectivity yields substantial advantages, it concurrently introduces a host of security vulnerabilities. Information security concerns have been an intrinsic part of computing since its inception, and unauthorized system access continues to be a paramount issue for IT specialists. These challenges are compounded when users access websites from geographically dispersed locations. For instance, downloading media files, often perceived as a

Received: Aug. 22, 2025.

2020 *Mathematics Subject Classification.* 05C90, 68P25, 94A60, 05C15.

Key words and phrases. graph theory; encryption; decryption; caterpillar graph; block coloring.

https://doi.org/10.28924/2291-8639-23-2025-312
ISSN: 2291-8639

low-risk endeavor, can expose sensitive information to potential security risks [1]. This underscores the critical role of computational algorithms in identifying and mitigating threats. Upon detecting a potential risk, the system may proactively block downloads to preserve data integrity and completeness; the prevalence of computer viruses exacerbates these security challenges. Similarly, transactions conducted over information networks are susceptible to interception by malicious agents.

Cryptography plays a crucial role in the field of cryptology, involving the methods and principles that ensure the safe exchange of information. The word cryptology comes from the Greek terms "kriptos," meaning hidden, and "logos," meaning study. This discipline is generally split into two main areas: cryptography, which deals with creating secure communication methods, and cryptanalysis, which is dedicated to deciphering and analyzing these secure messages [2]. Although cryptography focuses on constructing secure communication systems, cryptanalysis aims to breach these systems by exploiting vulnerabilities [3] and [4]. Historically, cryptography has evolved from simple ciphers, such as the Caesar cipher, to complex machines employed during World War II, significantly influencing Alan Turing's groundbreaking work in codebreaking [5]. Building on these historical foundations, modern cryptography has evolved into a sophisticated field that relies on mathematical principles and algorithms. It employs encryption and decryption to secure communication channels. Encryption transforms plaintext into ciphertext, rendering it unintelligible to the unauthorized parties. Decryption reverses this process by restoring the original plain text. Beyond these fundamental operations, modern cryptography prioritizes confidentiality, integrity, authentication, and nonrepudiation to safeguard information in the digital age [6]. Cryptographic algorithms are categorized into two primary types: symmetric key cryptography and public key (asymmetric) cryptography. Symmetric-key cryptography, while efficient, faces challenges in securely distributing and managing the shared keys. Prominent examples include the Data Encryption Standard (DES) and Advanced Encryption Standard (AES). By contrast, public-key cryptography employs a pair of keys: a public key for encryption and a private key for decryption. This approach addresses key distribution issues but often sacrifices performance [7]. As digital communication and online transactions proliferate, the demand for robust security solutions has increased. Graph theory, a branch of mathematics, has emerged as a powerful tool for enhancing cryptographic systems. By leveraging graph structures, cryptographers can design more complex and resilient algorithms, thereby increasing the difficulty for attackers to compromise sensitive information [8], [9], [10]. A strong foundation in graph theory is crucial for developing and analyzing cryptographic protocols and ensuring the confidentiality and integrity of digital communications [11]. Graph theory, a fundamental area of discrete mathematics, studies the properties of graphs, which are mathematical structures comprising of vertices and edges. Graphs are versatile tools used to model the relationships between objects in various

domains including electrical, transportation, and social networks. They offer a systematic approach to problem-solving and have been instrumental in the development of numerous software applications that facilitate communication and complex technical processes [12].

This study proposes a novel encryption technique that leverages the unique structures of homogeneous and non-homogeneous caterpillar graphs. We exploited this structural variation to generate complex key patterns. A shared secret key, derived from the specific structure of a homogeneous and nonhomogeneous caterpillar graph, is used to guide the encryption process. The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 provides the preliminaries. Section 4 details the proposed encryption method. Section 5 provides a practical example. Section 6 presents an analysis of this security scheme. Section 7 presents the Results and Discussion. Finally, Section 8 concludes the research work.

2. Related work

Our work is shaped by recent advancements in graph-theoretic cryptography, particularly those exploring the intersection of structural graph properties and symmetric encryption. This section reviews the key contributions that have influenced the development of our proposed caterpillar graph-based encryption scheme.

Ali et al. [13] proposed a set of innovative symmetric encryption schemes that leverage special graph structures—including corona graphs, star graphs, and complete bipartite graphs—combined with algebraic properties to enhance secure message transmission. Their framework addresses the increasing need for non-standard encryption algorithms to counter traditional attack models in secure digital communication. Ajeena et al. [14] introduced a symmetric encryption method that utilizes the corona graph (CG). Their approach constructs a CG by linking each vertex of a base graph to two new vertices. Ciphertexts are then generated through the random mapping of plaintext letters onto this structure, resulting in distinct graph configurations that are transmitted securely. Shathir et al. [15] proposed a symmetric encryption scheme based on a triple-vertex path (TVP) graph. This method encodes plaintext messages within TVP configurations, leveraging their structural characteristics to strengthen the encryption security and increase the ciphertext complexity.

Kumar et al. [16] explored the use of balanced bipartite trees combined with labeling techniques, such as harmonic, graceful, sequential, and felicitous labeling, for encryption and decryption. Their study emphasized how equal vertex distribution across bipartite partitions supports secure graph-structured encoding. Ni et al. [17] introduced a graph-theoretic encryption model that integrates algebraic operations with specialized graphs, such as corona, bipartite, and star graphs. Their framework relies on shared secret keys and algebraic transformations to establish secure communication between the parties. Beaula et al. [18] proposed a symmetric encryption technique that used a combination of path and double-vertex graphs. Their method integrates shared key usage with computational graph structures to improve the unpredictability and security of

encrypted text. Akl [19] formulated an encryption strategy that incorporated three sequentially constructed graphs. These graphs are created via atypical mapping functions, which are considered to replicate trapdoor one-way functions, possibly providing some resistance to attacks aimed at decrypting information.

Kaur et al. [20] introduced an innovative technique for encryption and decryption that leverages the adjacency matrix of a graph. This method, which utilizes a "double-transposition column" approach, offers numerous advantages over more straightforward algorithms.

Sabharwal et al. [21] introduced a hybrid encryption approach that integrates graph-theoretic cryptography with steganographic techniques. In this method, information is encrypted using association schemes defined on finite abelian groups and then hidden within a cover image chosen at random, enhancing both security and data concealment.

Ranasinghe et al. [22] introduced a symmetric encryption scheme based on snake vertex labeling over the super magic covering wheel graphs. By combining graph labeling with an enhanced shift cipher and a secret key pair (k, l), their scheme provides strong resistance against impersonation and replay attacks.

Despite these advancements in graph-based symmetric encryption, existing schemes often rely on rigid graph structures, such as fixed-size corona or bipartite graphs, and lack flexibility in encoding messages of varying lengths. Additionally, they under utilize structural and visual obfuscation strategies like vertex coloring, which can significantly enhance security. To the best of our knowledge, no prior work has leveraged both homogeneous and non-homogeneous caterpillar graphs in a unified framework for encryption, particularly with integrated color-block mapping and modular inverse operations. This gap motivates the present study, which proposes a lightweight, dual-mode encryption framework capable of achieving semantic security while being adaptable to resource-constrained applications.

3. Mathematical preliminaries

3.1. **Graph-Theoretic Concepts.**

3.1.1. *Graph.* A graph G is a triple consisting of a vertex set $V(G)$, an edge set $E(G)$, and a relation that associates with each edge two vertices (not necessarily distinct), called its endpoints.

3.1.2. *Caterpillar Graph.* A caterpillar graph is a tree graph with a central path (spine) and leaf nodes branching off from the spine. Alternatively, it is a tree in which all vertices are within distance one of a central path.

3.1.3. *Homogeneous and Non-Homogeneous Caterpillar Graphs.* A homogeneous caterpillar is a tree in which each vertex on the spine has an equal number of leaves. In contrast, a non-homogeneous caterpillar allows varying numbers of leaves per spine vertex, increasing structural diversity.

3.1.4. *Graph Coloring.* Graph coloring is a method of assigning labels, traditionally called "colors," to elements of a graph subject to certain constraints—most commonly, ensuring no two adjacent vertices share the same color.

3.1.5. *Block Coloring.* Block coloring is a variant of graph coloring in which vertices are grouped into blocks, and each block is assigned a unique color. Unlike traditional coloring, block coloring does not enforce adjacency constraints but follows application-specific rules, such as encryption mappings.

3.2. **Cryptographic Elements.**

3.2.1. *English Alphabet Values (EAVs).* Each English letter, both uppercase and lowercase, is assigned a unique numeric value according to its position in the alphabet: $A \to 1$, $B \to 2$, $\ldots$, $Z \to 26$. This mapping serves as the basis for converting plaintext characters into their numerical equivalents for encryption.

3.2.2. *Modular Arithmetic:* This includes operations such as modular addition and modular inversion. For a prime number $p > 26$, the modular inverse of a (where $\gcd(a, p) = 1$) is an integer a^{-1} satisfying: $a \cdot a^{-1} \equiv 1 \pmod{p}$

3.2.3. *Color Mapping Rule.* Each encrypted block is assigned a unique color based on a reversible mapping function known to both sender and receiver, used for secure graph segmentation during encryption and decryption.

3.3. **Color Assignment Rule.** The Color Assignment Rule consists of two key components: palette selection and the block–color mapping function.

Palette Selection: In the palette selection phase, a global set of C colors is predefined, with $C = 100$ recommended for practical cryptographic applications. From this global palette $\mathcal{P} = \{c_1, c_2, \ldots, c_C\}$, a subset of k active colors $\mathcal{P}_{\text{active}} = \{c_{a1}, c_{a2}, \ldots, c_{ak}\}$ is selected corresponding to the number of plaintext blocks. Maximum perceptual separation among these colors is ensured using standard color-difference metrics such as CIEDE2000 or HSL thresholding, minimizing ambiguity and enhancing security.

Block–Color Mapping Function: The mapping function deterministically assigns each plaintext block m_i a color $c_i \in \mathcal{P}_{\text{active}}$ through a reversible function f. This ensures that each block is associated with a unique color, while incorporating both the graph structure and the secret prime number to enhance security.

Implementation Formula: The mapping function f is implemented as:

$$c_i = f(m_i, \text{pos}_i, p) = \left(i \times \text{pos}_i \times p + \sum_{j=1}^{r_i} c_{ij} \right) \bmod C \tag{3.1}$$

where:

- i is the block index ($i = 1, 2, \ldots, k$),

- pos_i is the position of the first vertex of block i in the caterpillar graph,
- p is the shared secret prime number,
- $\sum_{j=1}^{r_i} c_{ij}$ is the sum of all ciphertext values in block i,
- C is the size of the color palette.

Security Properties: By combining deterministic assignment, reversibility, and cryptographic enhancement through the secret prime p and graph structure dependency, the color assignment contributes both to the uniqueness of ciphertext representation and to resistance against frequency and structural attacks.

Conflict Resolution: Since the direct application of f may result in collisions (two blocks mapping to the same color), a collision-handling strategy is employed. Specifically, linear probing is used: if the calculated color index is already assigned, the index is incremented modulo C until a free color is found. This guarantees a one-to-one mapping between blocks and active colors, ensuring reversibility while keeping the expected resolution cost negligible due to the large palette size relative to the number of blocks.

4. Proposed Methodology

This paper presents a novel cryptographic framework designed to improve data security in public networks by ensuring data integrity, confidentiality, and privacy. The core of the technique employs a unique graph structure, namely homogeneous and non-homogeneous caterpillar graphs, which serve as shared secret keys for secure communication.Caterpillar graphs are selected due to their favorable structural and cryptographic properties. Their backbone-based spine supports efficient encoding and decoding with linear complexity, while flexible leg arrangements introduce significant combinatorial variability. Non-homogeneous variants further enhance security by increasing irregularity, thereby improving resistance to pattern-based attacks. Compared to other graph classes such as general trees, bipartite graphs, or Hamiltonian paths, caterpillar graphs offer an effective balance between computational efficiency and key space complexity, making them suitable for secure and scalable cryptographic applications.

In the proposed method, plaintext characters are first converted into numerical values using number-theoretic transformations. These numerical values are then encrypted and mapped onto the vertices of the caterpillar graph. At this stage, the **Color Assignment Rule** (defined in Section 3.3) is applied: each plaintext block is assigned a unique color selected from the active palette. This block-wise coloring not only enables clear differentiation and structured mapping of ciphertext blocks but also enhances security by introducing an additional obfuscation layer that complicates frequency analysis and structural prediction attacks. The color assignment is guided by the shared secret prime key p, ensuring that the mapping remains reversible during decryption while strengthening confidentiality. Only authorized senders and receivers, possessing the secret prime key and graph structure, can correctly decode the ciphertext. The complete encryption workflow is illustrated in Fig.1.

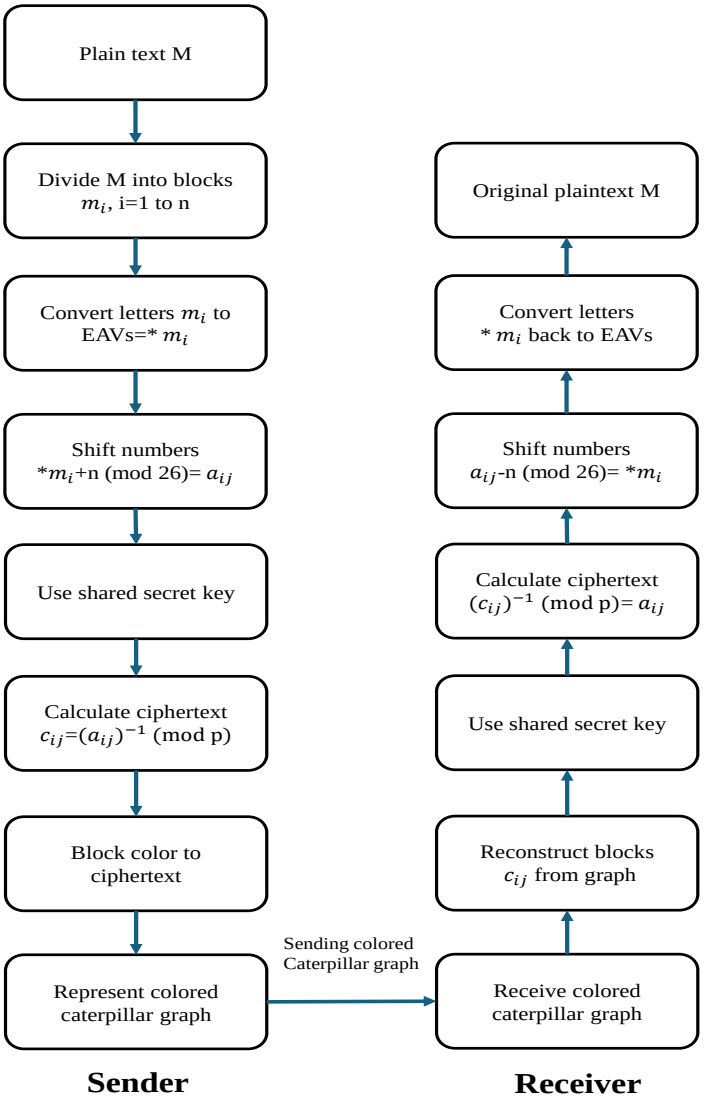

FIGURE 1. Workflow of the proposed encryption scheme.

4.1. **Secure Data transfer using Homogeneous Caterpillar Graph.** Consider a plaintext message M which is English word or sentence. The message consists of 26 or fewer letters from the English alphabet. Each letter is assigned a unique numerical value based on its Position in the English alphabet, known as English Alphabet Values (EAVs), where A $\rightarrow$ 1, B $\rightarrow$ 2, ..., Z$\rightarrow$ 26. This numerical assignment serves as the foundation for converting the plaintext into its corresponding numerical representation. Plaintext M is divided into blocks denoted as $M = m_i = (m_1, m_2, \ldots, m_n)$, where n is the total number of blocks. Each block m_i represents a subset of the plaintext message, containing a fixed number(length) of r letters, labeled as m_{i1}, m_{i2}, ..., m_{ir}. To increase encryption complexity, the blocks are distinguished by color assignment rule. Once the blocks are formed, each letter within the block is converted into its corresponding numerical equivalent by applying

EAVs, denoted as $*m_i$. To further strengthen the encryption process, the numerical values in each block are adjusted by adding the total number of blocks n to each letter's numerical value,resulting in the following equation:

$$\begin{aligned}
{}^{*}m_1 + n \pmod{26} &\equiv (a_{11}, a_{12}, \ldots, a_{1r}),\\
{}^{*}m_2 + n \pmod{26} &\equiv (a_{21}, a_{22}, \ldots, a_{2r}),\\
{}^{*}m_3 + n \pmod{26} &\equiv (a_{31}, a_{32}, \ldots, a_{3r}),\\
&\vdots\\
{}^{*}m_n + n \pmod{26} &\equiv (a_{n1}, a_{n2}, \ldots, a_{nr}).
\end{aligned}$$

The sender uses a shared secret key p, which is the prime number greater than 26 ($p > 26$). To compute the ciphertext C for the colored block elements, the sender calculates the modular multiplicative inverse of each numerical element with respect to p. The resulting ciphertext is represented as $C = (c_1, c_2, \ldots, c_n)$, where each block c_i is given by $c_1 = (c_{11}, c_{12}, \ldots, c_{1r}), c_2 = (c_{21}, c_{22}, \ldots, c_{2r}), c_3 = (c_{31}, c_{32}, \ldots, c_{3r}), \ldots, c_n = (c_{n1}, c_{n2}, ...c_{nr})$. The encryption is performed by computing the modular multiplicative inverse for each numerical value in the block as follows:

$$\begin{aligned}
c_{11} &\equiv (a_{11})^{-1} \pmod{p},\\
c_{12} &\equiv (a_{12})^{-1} \pmod{p},\\
c_{13} &\equiv (a_{13})^{-1} \pmod{p},\\
&\vdots\\
c_{nr} &\equiv (a_{nr})^{-1} \pmod{p}.
\end{aligned}$$

Thus, the ciphertext C is represented as a homogeneous caterpillar graph, where each block is randomly colored using block coloring to further obfuscate the plaintext structure. This encrypted message is then transmitted to the receiver, as illustrated in fig 2.

Upon receiving the homogeneous caterpillar graph, the receiver identifies the colored vertices and reconstructs ciphertext blocks $c_1, c_2, \ldots, c_n$. Using the shared secret key p, the receiver then calculates the modular inverses of the elements c_{ij} for each block. This process mathematically expressed as $(c_{ij})^{-1} \pmod{P} \equiv a_{ij}$, where i=1,2,...,$n$ and j=1,2,...,r. Once the values a_{ij} are obtained,the receiver computes the original adjusted values by subtracting n(the total number of blocks) and applying modulo 26.

This step is represented as, $a_{ij} - n \pmod{26} \equiv *m_i$ for i=1,2,...n. Finally, using the English Alphabet values(EAVs), the receiver converts the numerical values $*m_i$ back into their corresponding letters m_i, thereby reconstructing the original plaintext message M. This process ensures that only the intended receiver,with knowledge of the shared secret key, can successfully decode the encrypted message.

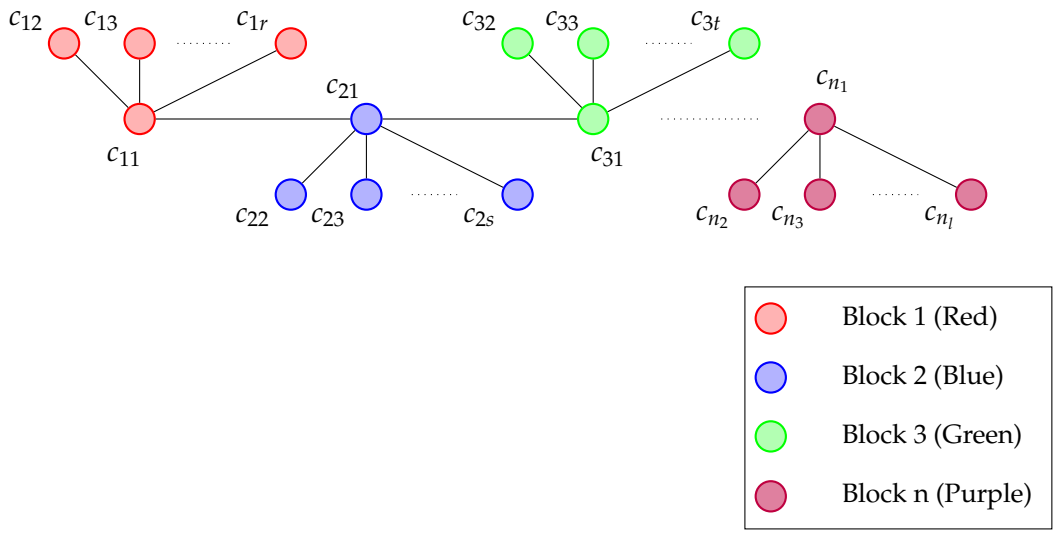

FIGURE 2. Homogeneous Caterpillar Graph: Each spine vertex has uniform leg distribution per block. Colors encode ciphertext blocks. Dotted lines indicate continuation of nodes.

4.2. **Secure Data Transfer Using Non-Homogeneous Caterpillar Graph.** Consider a plaintext message M, which is an English word or sentence consisting of 26 or fewer letters from the English alphabet. Each letter is assigned a unique numerical value based on its positions in the English alphabet, known as English Alphabet Values(EAVs), where A $\rightarrow$ 1, B $\rightarrow$ 2, ..., Z$\rightarrow$ 26. This numerical assignment serves as the foundation for converting the plaintext into its corresponding numerical representation. Plaintext M is divided into blocks denoted as $M = m_i = (m_1, m_2, \ldots, m_n)$, where n is the total number of blocks. Each block m_i represents a subset of the plaintext message, containing a different numbers(lengths) of $r,s,t \ldots l$ letters, labeled as m_{i1}, m_{i2}, ..., m_{ir},..., m_{il}. To increase encryption complexity, the blocks are distinguished by color assignment rule. Once the blocks are formed, each letter within the block is converted into its corresponding numerical equivalent by applying EAVs, denoted as $*m_i$. To further strengthen the encryption process, the numerical values in each block are adjusted by adding the total number of blocks n to each letter's numerical value, resulting in the following equation:

$$
\begin{aligned}
*m_1 + n \pmod{26} &\equiv (a_{11}, a_{12}, \ldots, a_{1r}) \\
*m_2 + n \pmod{26} &\equiv (a_{21}, a_{22}, \ldots, a_{2s}) \\
*m_3 + n \pmod{26} &\equiv (a_{31}, a_{32}, \ldots, a_{3t}) \\
&\vdots \\
*m_n + n \pmod{26} &\equiv (a_{n1}, a_{n2}, \ldots, a_{nl})
\end{aligned}
$$

The sender uses a shared secret key, which is a prime number $p > 26$. The ciphertext C for the colored block elements is computed using the modular multiplicative inverse of the elements with respect to p, resulting in $C = (c_1, c_2, \ldots, c_n)$, where each block is represented as $c_1 = (c_{11}, c_{12}, \ldots, c_{1r}), c_2 = (c_{21}, c_{22}, \ldots, c_{2s}), c_3 = (c_{31}, c_{32}, \ldots, c_{3t}), \ldots, c_n = (c_{n1}, c_{n2}, \ldots c_{nl})$. The calculations are carried out as follows.

$$\begin{aligned} c_{11} &\equiv (a_{11})^{-1} \pmod{p} \\ c_{12} &\equiv (a_{12})^{-1} \pmod{p} \\ c_{13} &\equiv (a_{13})^{-1} \pmod{p} \\ &\vdots \\ c_{nl} &\equiv (a_{nl})^{-1} \pmod{p} \end{aligned}$$

Thus, ciphertext C is then represented as a nonhomogeneous caterpillar graph with random block coloring assigned to each block. The sender transmits this graph to the receiver, as shown in fig 3.

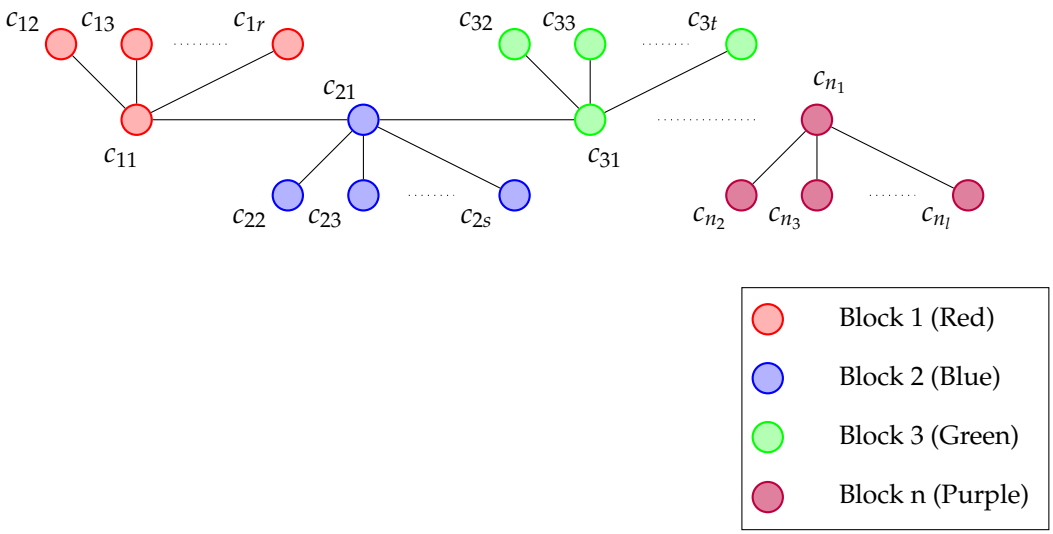

FIGURE 3. Non-Homogeneous Caterpillar Graph used as a shared secret key, where each block is represented by a distinct color. Dotted lines indicate continuation of nodes.

Upon receiving the non-homogeneous caterpillar graph, the receiver begins by identifying the colored vertices, which assists in reconstructing the ciphertext blocks $c_1, c_2, \ldots, c_n$. For each element c_{ij} within these blocks, the receiver computes its modular inverse relative to the shared secret prime number p, thereby retrieving a_{ij}. To reverse the initial block shift applied during encryption, the receiver subtracts the total number of blocks *(n)* from each value a_{ij} and computes the modulo 26 result. This produces $*m_i$ for $i = 1, 2, \ldots n$. By applying the English Alphabet Values (EAVs), each numerical value $*m_i$ is converted back into its corresponding letter. Finally, the receiver assembles the blocks to reconstruct the original plaintext message, $M = (m_1, m_2, \ldots, m_i)$. This approach

ensures a secure and structured encryption mechanism by leveraging the properties of modular arithmetic and graph-based transformations.

5. Practical Example

5.1. **Example of Homogeneous caterpillar graph.** Suppose M is a plain text message that is given by the sentence **"Practice makes perfect"** . Using English alphabet values (EAVs), we converted the letters of the plaintext into numbers A $\rightarrow$ 1, B $\rightarrow$ 2, ..., Z$\rightarrow$ 26. English Alphabet values conversion of the plaintext is "Practice makes perfect" is P $\rightarrow$ 16, r $\rightarrow$ 18, c $\rightarrow$ 3, t $\rightarrow$ 20, i $\rightarrow$ 9, c $\rightarrow$ 3, e $\rightarrow$ 5, m $\rightarrow$ 13, a $\rightarrow$ 1, k $\rightarrow$ 11, e $\rightarrow$ 5, s $\rightarrow$ 19, p $\rightarrow$ 16, e $\rightarrow$ 5, r $\rightarrow$ 18, f $\rightarrow$ 6, e $\rightarrow$ 5, c $\rightarrow$ 3, t $\rightarrow$ 20, Here we divided into 4 number of colored blocks $M = (m_1, m_2, m_3, m_4)$, such that

$$m_1 = Pract = (16, 18, 1, 3, 20)$$
$$m_2 = \text{icema} = (9, 3, 5, 13, 1)$$
$$m_3 = \text{kespe} = (11, 5, 19, 16, 5)$$
$$m_4 = \text{rfect} = (18, 6, 5, 3, 20)$$

As there are 4 blocks, the numbers within these blocks are modified by adding 4 to them through a shift cipher, specifically.

$$(16, 18, 1, 3, 20) + 4 \pmod{26} = (20, 22, 5, 7, 24) = a_1$$
$$(9, 3, 5, 13, 1) + 4 \pmod{26} = (13, 7, 9, 17, 5) = a_2$$
$$(11, 5, 19, 16, 5) + 4 \pmod{26} = (15, 9, 23, 20, 9) = a_3$$
$$(18, 6, 5, 3, 20) + 4 \pmod{26} = (22, 10, 9, 7, 24) = a_4$$

Now sender then uses the shared secret key, which is a prime number $p = 29$, where $29 > 26$. The ciphertext $C = (c_1, c_2, c_3, c_4)$, where $c_1 = (c_{11}, c_{12}, c_{13}, c_{14}, c_{15})$, $c_2 = (c_{21}, c_{22}, c_{23}, c_{24}, c_{25})$, $c_3 = (c_{31}, c_{32}, c_{33}, c_{34}, c_{35})$ and $c_4 = (c_{41}, c_{42}, c_{43}, c_{44}, c_{45})$. The calculations were performed as follows:

$$c_1 = ((20)^{-1}, (22)^{-1}, (5)^{-1}, (7)^{-1}, (24)^{-1}) \pmod{29} = (5, 4, 6, 25, 5)$$
$$c_2 = ((13)^{-1}, (7)^{-1}, (9)^{-1}, (17)^{-1}, (5)^{-1}) \pmod{29} = (9, 25, 13, 12, 6)$$
$$c_3 = ((15)^{-1}, (9)^{-1}, (23)^{-1}, (20)^{-1}, (9)^{-1}) \pmod{29} = (2, 13, 4, 15, 13)$$
$$c_4 = ((22)^{-1}, (10)^{-1}, (9)^{-1}, (7)^{-1}, (24)^{-1}) \pmod{29} = (4, 3, 13, 25, 26)$$

$\therefore$ The ciphertext values are represented as vertices in a caterpillar graph with colored vertices assigned to each block using color assignment rule, as illustrated in fig 4, and are then transmitted to the recipient. The recipient receives a homogeneous caterpillar graph and extracts the vertices from the colored blocks c_1, c_2, c_3, c_4. The extract colored blocks are (5,4,6,25,5), (9,25,13,12,6), (2,13,4,15,13) and (4,3,13,25,26). The receiver calculates the modular inverse of each ciphertext element c_{ij} ,i=1,2,3,4 and j=1,2,3,4,5. The receiver knows the shared secret key $p = 29$ to calculate the inverse

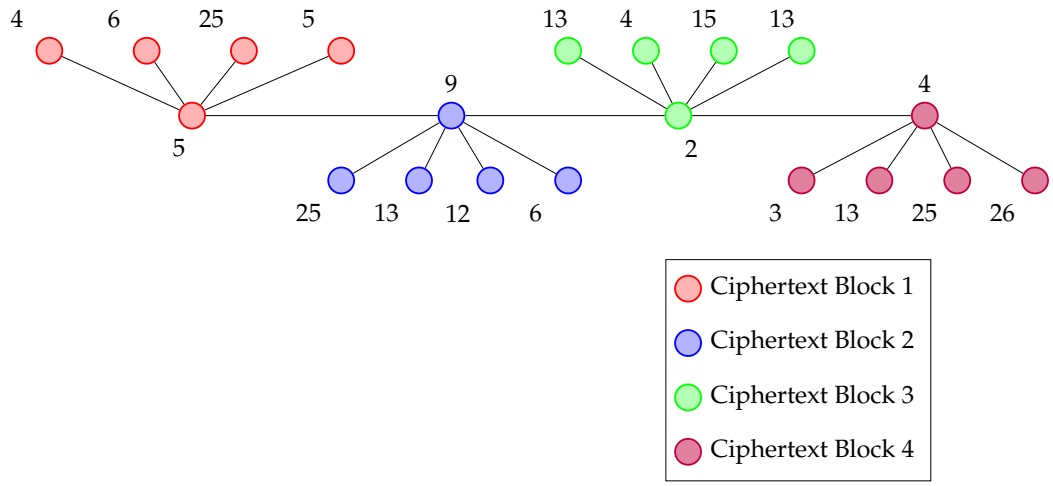

FIGURE 4. Homogeneous caterpillar graph with block coloring. The legend indicates ciphertext blocks.

numbers of c_1, c_2, c_3, c_4 as follows

$$\begin{aligned}
a_1 &= ((5)^{-1}, (4)^{-1}, (6)^{-1}, (25)^{-1}, (5)^{-1}) \pmod{29} = (20, 22, 5, 7, 24) \\
a_2 &= ((9)^{-1}, (25)^{-1}, (13)^{-1}, (12)^{-1}, (6)^{-1}) \pmod{29} = (13, 7, 9, 17, 5) \\
a_3 &= ((2)^{-1}, (13)^{-1}, (4)^{-1}, (15)^{-1}, (13)^{-1}) \pmod{29} = (15, 9, 23, 20, 9) \\
a_4 &= ((4)^{-1}, (3)^{-1}, (13)^{-1}, (25)^{-1}, (26)^{-1}) \pmod{29} = (22, 10, 9, 7, 24)
\end{aligned}$$

Given that the plaintext consists of 4 blocks, the receiver subtracts 4 from each block using modulo 26. The calculations are performed as follows.

$$\begin{aligned}
(20, 22, 5, 7, 24) - 4 \pmod{26} &= (16, 18, 1, 3, 20) = a_1 \\
(13, 7, 9, 17, 5) - 4 \pmod{26} &= (9, 3, 5, 13, 1) = a_2 \\
(15, 9, 23, 20, 9) - 4 \pmod{26} &= (11, 5, 19, 16, 5) = a_3 \\
(22, 10, 9, 7, 24) - 4 \pmod{26} &= (18, 6, 5, 3, 20) = a_4
\end{aligned}$$

Finally the numerical values are converted back to their correspond letters.

$$\begin{aligned}
(16, 18, 1, 3, 20) &\to (P, r, a, c, t) \\
(9, 3, 5, 13, 1) &\to (i, c, e, m, a) \\
(11, 5, 19, 16, 5) &\to (k, e, s, p, e) \\
(18, 6, 5, 3, 20) &\to (r, f, e, c, t)
\end{aligned}$$

$\therefore$ The rearranged sentence is "Practice makes perfect".

5.2. **Example of Non-Homogeneous caterpillar graph.** Suppose M is a plaintext message that is given by the sentence **"I like to eat pizza"**. Here we divide into 5 blocks $M = (m_1, m_2, m_3, m_4, m_5)$,

such that

$$m_1 = I, m_2 = like, m_3 = to, m_4 = eat, m_5 = pizza$$

. Using the EAVs, the plaintext message blocks m_i, where i = 1, 2, 3,4,5 can be transformed into numerical values in the following way:

$$\begin{aligned}
m_1 &= I = 9 \\
m_2 &= like = (12, 9, 11, 5) \\
m_3 &= to = (20, 15) \\
m_4 &= eat = (5, 1, 20) \\
m_5 &= \text{pizza} = (16, 9, 26, 26, 1)
\end{aligned}$$

As there are 5 blocks, the numbers within these blocks are modified by adding 5 to them through a shift cipher, specifically.

$$\begin{aligned}
9 + 5 \pmod{26} &= 14 = a_1 \\
(12, 9, 11, 5) + 5 \pmod{26} &= (17, 14, 16, 10) = a_2 \\
(20, 15) + 5 \pmod{26} &= (25, 20) = a_3 \\
(5, 1, 20) + 5 \pmod{26} &= (10, 6, 25) = a_4 \\
(16, 9, 26, 26, 1) + 5 \pmod{26} &= (21, 14, 5, 5, 6) = a_5
\end{aligned}$$

Now, sender takes the shared secret key which is a prime number $p = 31$, where $31 > 26$. The ciphertext $C = (c_1, c_2, c_3, c_4, c_5)$, where $c_1 = (c_{11})$, $c_2 = (c_{21}, c_{22}, c_{23}, c_{24})$, $c_3 = (c_{31}, c_{32})$, $c_4 = (c_{41}, c_{42}, c_{43})$ and $c_5 = (c_{51}, c_{52}, c_{53}, c_{54}, c_{55})$. The calculations are carried out as follows.

$$\begin{aligned}
c_1 &= (14)^{-1} \pmod{31} = 20 \\
c_2 &= ((17)^{-1}, (14)^{-1}, (16)^{-1}, (10)^{-1}) \pmod{31} = (11, 20, 2, 28) \\
c_3 &= ((25)^{-1}, (20)^{-1}) \pmod{31} = (5, 14) \\
c_4 &= ((10)^{-1}, (6)^{-1}, (25)^{-1}) \pmod{31} = (28, 26, 5) \\
c_5 &= ((21)^{-1}, (14)^{-1}, (5)^{-1}, (5)^{-1}, (6)^{-1}) \pmod{31} = (3, 20, 25, 15, 26)
\end{aligned}$$

$\therefore$ The ciphertext values are subsequently designated as vertices in a caterpillar graph with colored vertices assigned to each block, as illustrated in fig 5, and are then transmitted to the recipient.

The recipient receives the caterpillar graph and first forms blocks of the ciphertext based on the colored vertices of the caterpillar graph. So the first block is 20, second is (11,20,2,28),third is (5,14),fourth is (28,26,5) and fifth is (3,20,25,15,26). The receiver knows the shared secret key $p = 31$

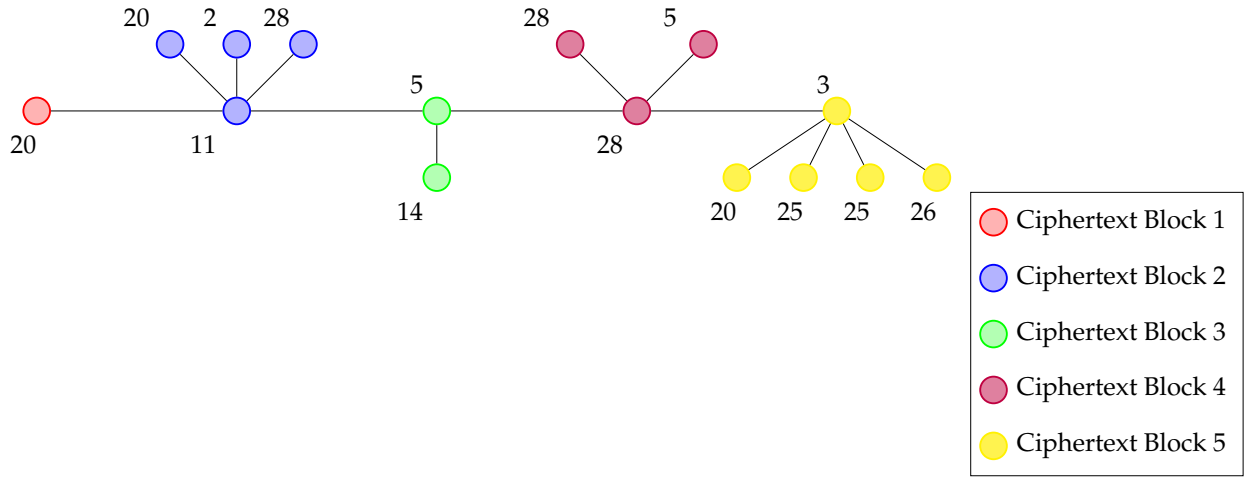

FIGURE 5. Non-homogeneous caterpillar graph with block-wise color encoding

to calculate the inverse numbers of c_1, c_2, c_3, c_4, c_5 as follows:

$$a_1 = (20)^{-1} \pmod{31} = 14$$
$$a_2 = ((11)^{-1}, (20)^{-1}, (2)^{-1}, (28)^{-1}) \pmod{31} = (17, 14, 16, 10)$$
$$a_3 = ((5)^{-1}, (14)^{-1}) \pmod{31} = (25, 20)$$
$$a_4 = ((28)^{-1}, (26)^{-1}, (5)^{-1}) \pmod{31} = (10, 6, 25)$$
$$a_5 = ((3)^{-1}, (20)^{-1}, (25)^{-1}, (25)^{-1}, (26)^{-1}) \pmod{31} = (21, 14, 5, 5, 6)$$

Given that the plaintext consists of 5 blocks, the receiver subtracts 5 from each block using modulo 26. The calculations are performed as follows.

$$14 - 5 \pmod{26} = (9) = a_1 = a_{11}$$
$$(17, 14, 16, 10) - 5 \pmod{26} = (12, 9, 11, 5) = a_2$$
$$(25, 20) - 5 \pmod{26} = (20, 15) = a_3$$
$$(10, 6, 25) - 5 \pmod{26} = (5, 1, 20) = a_4$$
$$(21, 14, 5, 5, 6) - 5 \pmod{26} = (16, 9, 26, 26, 1) = a_5$$

∴ The corresponding EAV's are

$$(9) \to (I)$$
$$(12, 9, 11, 5) \to (l, i, k, e)$$
$$(20, 15) \to (t, o)$$
$$(5, 1, 20) \to (e, a, t)$$
$$(16, 9, 26, 26, 1) \to (p, i, z, z, a)$$

∴ The recovered original plaintext is "I like to eat pizza"

6. Security scheme

The proposed encryption techniques utilizing both homogeneous and nonhomogeneous Caterpillar Graphs offer enhanced security compared with conventional encryption methods. In these techniques, ciphertext is represented in the form of homogeneous and nonhomogeneous Caterpillar Graphs. The security of this approach is based on the random generation of these graphs, making it significantly challenging for an attacker to decrypt ciphertext. To successfully retrieve the plaintext, an attacker must not only comprehend the structure of the Caterpillar Graph, but also accurately identify its vertices.

In Scenario I, the construction of a homogeneous Caterpillar Graph required the selection of a subset of vertices from a given set. The number of possible configurations for selecting v vertices from n total nodes can be determined by using the following combination formula:

$$nc_v = \frac{n!}{v!(n-v)!}$$

where v is the number of vertices in the homogeneous Caterpillar Graphs. When n = 26 and v = 20, the total number of combinations can be determined by the following formula:

$$26c_{20} = \frac{26!}{20!(26-20)!} = 230230$$

Among these 230,230 potential configurations, only one corresponds to the correct key, thereby highlighting the strength of this encryption scheme. To further enhance security, a color-based encoding scheme was introduced. Colors are a fundamental aspect of human visual perception and are created when light reflects off objects and enters the eye. Although humans can perceive approximately 10 million distinct colors for encryption purposes, a limited subset of 100 colors is considered. For the first encryption case, only four colors are selected from this set of 100, which follows the combination formula

$$100c_4 = \frac{100!}{100!(100-4)!} = 3921225$$

Thus, the total number of possible encryption configurations incorporating both vertex selection and color encoding is: 230,230×3,921,225=902,783,631,750 ways. This immense number of possible encryption schemes renders brute-force attacks computationally infeasible, further reinforcing the robustness of the proposed encryption technique.

In Scenario II, the security of the proposed encryption technique was further reinforced by modifying the parameters used in the graph-based encryption scheme. Specifically, the nonhomogeneous Caterpillar Graph in this scenario was constructed using v=15 vertices selected from a total of n=26. The number of possible configurations for selecting these nodes can be determined by using the following combination formula:

$$26c_{15} = \frac{26!}{15!(26-15)!} = 7726160$$

Among the 7,726,160 possible configurations, only one corresponds to the correct encryption key, demonstrating the robustness of the method against brute-force decryption attempts. To further enhance security, a color-based encoding scheme was introduced, similar to Scenario I, but with increased complexity. In this case, five colors are selected from a total of 100 available colors, which follow the combination formula

$$100c_5 = \frac{100!}{100!(100-5)!} = 75287520$$

Thus, considering both vertex selection and color encoding, the total number of possible encryption configurations is. $75,287,520 \times 7,726,160 = 58,168,342,552,320$. This immense number of possible encryption schemes makes brute-force attacks virtually impossible, ensuring a high level of security for encrypted data. The combined complexity of graph-based vertex selection and color-based encoding significantly strengthens the resilience of encryption techniques against cryptanalysis.

7. Results and Discussion

To demonstrate the effectiveness of the proposed lightweight symmetric encryption framework, a comparative study was performed against existing graph-based cryptographic techniques. The assessment focused on key security and performance aspects, including structural complexity, key space size, encryption and decryption time complexity, memory consumption, scalability, resistance to quantum attacks, frequency analysis, brute-force attacks, and practical usability. The results, summarized in Table 1, show that the proposed caterpillar graph-based scheme provides improved security due to its large key space, color-based obfuscation, and combinatorial complexity, while maintaining an efficient time complexity of $O(n \log n)$. Compared to other graph-based models such as general trees, bipartite graphs, Hamiltonian paths, and corona graphs, the caterpillar framework achieves an excellent balance between security robustness and computational efficiency.

7.1. **Key Management Strategies.**

Effective key management is crucial to maintaining the confidentiality and integrity of the proposed encryption system that employs caterpillar graphs. The security of this method depends on properly managing three key elements: (i) a prime number $p > 26$, (ii) the caterpillar graph structure G, and (iii) the block–color mapping function f. This section presents two main strategies for securely managing these keys while ensuring an efficient time complexity of $O(n \log n)$. Compared to other graph-based models like general trees, bipartite graphs, Hamiltonian paths, and corona graphs, the caterpillar graph approach offers an excellent balance between robust security and computational efficiency.

7.1.1. *Key Generation.*

The secret key $K = (p, G, f)$ is generated by combining three components: a prime number p, a caterpillar graph G, and a block–color mapping function f. First, a prime number $p > 26$ is randomly selected from a secure set of primes. The size of p is chosen to balance

TABLE 1. Comparative Analysis: Caterpillar Graph-Based Encryption vs. Other Graph Models

Criterion	**Caterpillar Graphs (Proposed)**	**General Trees**	**Bipartite Graphs**	**Hamiltonian Paths**	**Corona Graphs**
Structural Complexity	High (Spine with variable legs and color encoding)	Medium (Hierarchical branching)	Medium (Two-partition sets)	High (Complete path traversal)	Very High (Composite structure)
Key Space Size	$\sim 2^{150}$ $(\binom{n}{v} \times \binom{100}{k} \times 2^{\lambda})$	$\sim 2^{128}$	$\sim 2^{140}$	$\sim 2^{145}$	$\sim 2^{155}$
Encryption Time	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	$O(n!)$	$O(n^2 \log n)$
Decryption Time	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	$O(n^2)$	$O(n^2 \log n)$
Memory Requirements	Moderate	Low	Moderate	High	High
Scalability	Excellent	Good	Moderate	Poor	Moderate
Quantum Resistance	High	Medium	Medium	Low	High
Visual Obfuscation	Excellent (Color coding)	Good	Moderate	Poor	Excellent
Frequency Analysis Resistance	High	Medium	Medium	Low	High
CPA Resistance	High (IND-CPA secure under assumptions)	Medium	Medium	Low	High
Brute Force Resistance	Excellent (Exponential keyspace)	Good	Good	Poor	Excellent
Implementation Complexity	Moderate	Low	Moderate	High	High
Error Propagation	Low (Block-wise containment)	Medium	Medium	High	Low
Parallel Processing Support	Excellent	Good	Good	Poor	Moderate
Storage Efficiency	Good	Excellent	Good	Poor	Moderate
Adaptability	High (Homogeneous and Non-Homogeneous modes)	Medium	Medium	Low	High
Cryptanalysis Resistance	High	Medium	Medium	Low	High
Real-World Applicability	High	Medium	Medium	Low	Medium

computational efficiency with resistance to brute-force attacks, ensuring both practicality and security. Then, the caterpillar graph G is constructed either homogeneously or non-homogeneously. Random variations in the spine length and leg distribution introduce structural diversity, making each graph correspond to a unique key space. This variability increases the unpredictability of the secret key. Finally, the block–color mapping function f assigns a unique color to each block from the palette $\mathcal{P}$. This assignment depends on both the vertex position and the prime p, creating a reversible mapping that enhances security. This method further protects against frequency and structural attacks, as ciphertext blocks cannot be easily correlated without knowledge of the shared prime and mapping function.

7.1.2. *Key Exchange Protocol.* For secure communication, both the sender and receiver need to share the secret key $K = (p, G, f)$. The prime number p can be safely exchanged using a standard symmetric key exchange method, such as Diffie–Hellman or its post-quantum alternatives. The caterpillar graph structure G and the mapping function f are either shared beforehand via a trusted channel or sent encrypted along with p. To prevent replay attacks and guarantee forward secrecy, the secret key K should be periodically updated by regenerating p, G, and f after a set session duration. This approach, combining secure key generation with a dependable exchange protocol, ensures that the proposed system upholds confidentiality, authenticity, and resilience against attacks.

7.1.3. *Resistance Against Man-in-the-Middle (MITM) Attacks.* A major risk to key exchange protocols is the man-in-the-middle (MITM) attack, in which an attacker intercepts and may modify the communication between the sender and receiver. To prevent this, the proposed framework incorporates mutual authentication, requiring both parties to confirm each other's identities before creating a shared key. Techniques such as digital signatures or hash-based message authentication codes (HMACs) can be used to link exchanged values to the participants' identities, preventing attackers from impersonating legitimate users. This authentication layer ensures that even if the communication channel is compromised, an attacker cannot insert or alter keys without being detected.

7.1.4. *Resistance Against Side-Channel Attacks.* Side-channel attacks exploit unintended information leaks that occur during encryption or decryption, such as differences in timing, power consumption, or electromagnetic emissions. The suggested framework incorporates several defenses against these threats. First, critical operations are performed in constant time to prevent timing-based leaks. Second, the sequence of computations is randomized, and masking methods are applied to hide intermediate values, lowering the chances of successful power analysis and electromagnetic attacks. These safeguards make it computationally infeasible for an attacker with physical access to the cryptographic device to obtain sensitive information about the secret key or internal state.

7.1.5. *Limitations and Considerations.* While our comparative analysis demonstrates the effectiveness of the proposed framework, it is important to acknowledge certain limitations. Although the proposed caterpillar graph-based encryption framework offers notable benefits in security and computational efficiency, several limitations should be recognized. The system currently supports only the 26-letter English alphabet, restricting its use in multilingual environments requiring Unicode and extended character sets. The security approach relies on managing three separate components (prime p, graph structure G, and color mapping function f), making key distribution more complex than traditional single-key systems. Graph structure complexity increases memory usage, communication bandwidth demands, and creates scalability challenges for large datasets or resource-constrained environments. Practical implementations may be susceptible to side-channel attacks including timing analysis of modular inverse computations and power analysis during graph construction. The color assignment method faces potential collisions due to fixed palette limitations and color space inconsistencies across display devices. Additionally, the framework lacks industry standardization and certification protocols, limiting enterprise adoption. Implementation challenges include computational overhead from extended Euclidean algorithms, platform compatibility issues, and integration difficulties with existing cryptographic systems. Despite these limitations, the framework remains valuable for research, educational applications, and specialized visual encryption scenarios, though production deployment requires addressing these considerations while preserving core advantages in combinatorial security and visual obfuscation.

8. conclusion

This paper introduces an innovative cryptographic framework that utilizes the structural characteristics of both homogeneous and non-homogeneous Caterpillar Graphs, combined with color-based encoding, to improve encryption security and computational efficiency. The method embeds ciphertext within graph structures, employing vertex selection and color assignment to create substantial combinatorial complexity, making decryption without prior knowledge computationally impractical. The randomness involved in graph construction, vertex choice, and color encoding, along with the extensive range of possible encryption configurations, provides strong protection against brute-force and quantum attacks. By integrating graph theory, combinatorial mathematics, and color-based encoding, the approach further strengthens cryptographic security. Both theoretical analysis and experimental results demonstrate the scheme's effectiveness in securing data transmission while maintaining computational efficiency. Its flexibility allows for customization through adjustments in graph parameters and encoding methods, making it suitable for secure communications, financial transactions, and cloud data protection. Future work may focus on optimizing the encryption process for resource-limited devices such as IoT gadgets and wireless sensor networks, as well as expanding its application to various data types including images and audio. Moreover, combining this technique with post-quantum cryptographic protocols and

hybrid encryption models could improve resilience against emerging cyber threats. By advancing graph-based cryptography integrated with color-based encoding, this study contributes to the development of secure, scalable, and efficient encryption solutions designed for the evolving cybersecurity environment.

Conflicts of Interest: The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] A. Singh, A New Approach to Enhance Avalanche Effect in Aes to Improve Computer Security, J. Inf. Technol. Softw. Eng. 05 (2015), 1. https://doi.org/10.4172/2165-7866.1000143.

[2] A. Abboud, Protecting Documents Using Visual Cryptography, Int. J. Eng. Res. Gen. Sci. 3 (2015), 464–470.

[3] A.I. Khamg, A.R. Ramli, Implementation and Evaluation of New Cryptography Algorithm for E-mail Applications, Int. J. Comput. Internet Manag. 17 (2009), 34–39.

[4] T. Kivinen, J. Snyder, Signature Authentication in the Internet Key Exchange Version 2 (IKEv2), RFC Editor, 2015. https://doi.org/10.17487/rfc7427.

[5] J. Sanchez, R. Correa, H. Buenano, S. Arias, H. Gomez, Encryption Techniques: A Theoretical Overview and Future Proposals, in: 2016 Third International Conference on eDemocracy & eGovernment (ICEDEG), IEEE, 2016, pp. 60-64. https://doi.org/10.1109/icedeg.2016.7461697.

[6] A.A. Soofi, I. Riaz, U. Rasheed, An Enhanced Vigenere Cipher For Data Security, Int. J. Sci. Technol. Res. 5 (2016), 141–145.

[7] W. Etaiwi, Encryption Algorithm Using Graph Theory, J. Sci. Res. Rep. 3 (2014), 2519–2527. https://doi.org/10.9734/jsrr/2014/11804.

[8] M. Yamuna, M. Gogia, A. Sikka, M.J.H. Khan, Encryption Using Graph Theory and Linear Algebra, Int. J. Comput. Appl. 5 (2012), 102–107.

[9] M. Yamuna, K. Karthika, Data Transfer Using Bipartite Graphs, Int. J. Adv. Res. Sci. Eng. 4 (2015), 128–131.

[10] P. Priyadarsini, A Survey on Some Applications of Graph Theory in Cryptography, J. Discret. Math. Sci. Cryptogr. 18 (2015), 209–217. https://doi.org/10.1080/09720529.2013.878819.

[11] W. Stallings, Cryptography and Network Security: Principles and Practice, Pearson, 2014.

[12] K. Bekkaoui, S. Ziti, F. Omary, Data Security: A New Symmetric Cryptosystem Based on Graph Theory, Int. J. Adv. Comput. Sci. Appl. 12 (2021), 742–750. https://doi.org/10.14569/ijacsa.2021.0120982.

[13] N. Ali, A. Sadiqa, M.A. Shahzad, M. Imran Qureshi, H.M.A. Siddiqui, et al., Secure Communication in the Digital Age: A New Paradigm with Graph-Based Encryption Algorithms, Front. Comput. Sci. 6 (2024), 1454094. https://doi.org/10.3389/fcomp.2024.1454094.

[14] R.K.K. Ajeena, F.A. Abdullatif, S.M. Aboud, The Corona Graph for Symmetric Encryption Schemes, in: 2023 Fifth International Conference on Electrical, Computer and Communication Technologies (ICECCT), IEEE, 2023, pp. 1-5. https://doi.org/10.1109/icecct56650.2023.10179762.

[15] M.K. Shathir, G.E. Arif, R.K.K. Ajeena, More Secure on the Symmetric Encryption Schemes Based on Triple Vertex Path Graph, J. Discret. Math. Sci. Cryptogr. 26 (2023), 1175–1182. https://doi.org/10.47974/jdmsc-1564.

[16] D.K. Gurjar, A. Krishnaa, Balanced Bipartite Trees in Cryptography, Indian J. Sci. Res. 12 (2022), 35–40. https://doi.org/10.32606/ijsr.v12.i2.00005.

[17] B. Ni, R. Qazi, S.U. Rehman, G. Farid, Some Graph-Based Encryption Schemes, J. Math. 2021 (2021), 6614172. https://doi.org/10.1155/2021/6614172.

[18] C. Beaula, P. Venugopal, Encryption Using Double Vertex Graph and Matrices, Solid State Technol. 64 (2021), 2486–2493.

[19] S.G. Akl, The Graph Is the Message: Design and Analysis of an Unconventional Cryptographic Function, in: From Parallel to Emergent Computing, CRC Press, 2019.

[20] D.G. Kaur, D.N. Tripathi, Applying Graph Theory to Secure Data by Cryptography, Int. J. Linguist. Comput. Appl. 8 (2021), 1–3. https://doi.org/10.30726/ijlca/v8.i1.2020.81001.

[21] A. Sabharwal, P. Yadav, K. Kumar, Graph Crypto-Stego System for Securing Graph Data Using Association Schemes, J. Appl. Math. 2024 (2024), 2084342. https://doi.org/10.1155/2024/2084342.

[22] P. Ranasinghe, R. Bandara, A. Athapaththtu, Symmetric Encryption Using Snake Graphs and Supermagic Covering, J. Natl. Sci. Found. Sri Lanka 52 (2025), 435–440. https://doi.org/10.4038/jnsfsr.v52i4.12196.

[23] W. Stallings, Cryptography and Network Security: Principles and Practice, Pearson, 2017.

[24] N. Koblitz, A Course in Number Theory and Cryptography, Springer, 1994.

[25] P.W. Shor, Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM Rev. 41 (1999), 303–332. https://doi.org/10.1137/s0036144598347011.

[26] L.K. Grover, A Fast Quantum Mechanical Algorithm for Database Search, in: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing - STOC '96, ACM Press, New York, 1996, pp. 212-219. https://doi.org/10.1145/237814.237866.

[27] D.J. Bernstein, T. Lange, Post-Quantum Cryptography, Nature 549 (2017), 188–194. https://doi.org/10.1038/nature23461.

[28] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.

[29] C.E. Shannon, A Mathematical Theory of Communication, Bell Syst. Tech. J. 27 (1948), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.

[30] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

[31] D.J. Bernstein, Cache-Timing Attacks on AES, University of Illinois at Chicago, 2005.