symmetry

Article

Efficient Graph Algorithms in Securing Communication Networks

Syed Ahtsham Ul Haq Bokhary [1,*], Athar Kharal [1], Fathia M. Al Samman [2], Mhassen. E. E. Dalam [3] and Ameni Gargouri [4]

1 Centre for Advanced Studies in Pure and Applied Mathematics, Bahauddin Zakariya University, Multan 60000, Pakistan; atharkharal@bzu.edu.pk
2 Department of Mathematics, College of Science, Northern Border University, Arar 91431, Saudi Arabia; fathia.alsaman@nbu.edu.sa
3 Department of Mathematics, College of Sciences and Arts (Muhyil), King Khalid University, Muhyil 61421, Saudi Arabia; mdalam01@kku.edu.sa
4 Mathematics Department, College of Humanities and Science in Al Aflaj, Prince Sattam bin Abdulaziz University, Al-Kharj 11912, Saudi Arabia; a.gargouri@psau.edu.sa
* Correspondence: sihtsham@gmail.com; Tel.: +92-334-4141694

Abstract: This paper presents three novel encryption and decryption schemes based on graph theory that aim to improve security and error resistance in communication networks. The novelty of this work lies in the application of complete bipartite graphs in two of the schemes and the Cartesian product of graphs in the third, representing a unique approach to cryptographic algorithm development. Unlike traditional cryptographic methods, these graph-based schemes use structural properties of graphs to achieve robust encryption, providing greater resistance to attacks and corruption. Each scheme is illustrated with detailed examples that show how the algorithms can be successfully implemented. The algorithms are written in standard mathematical notation, making them adaptable for machine implementation and scalable for real-world use. The schemes are also rigorously analyzed and compared in terms of their temporal and spatial complexities, using Big $\mathcal{O}$ notation. This comprehensive evaluation focuses on their effectiveness, providing valuable insights into their potential for secure communication in modern networks.

Keywords: graph; encryption; decryption; complete graph; bipartite; Cartesian product graph; complexity; big $\mathcal{O}$

check for updates

Citation: Bokhary, S.A.U.H.; Kharal, A.; Samman, F.M.A.; Dalam, M.E.E.; Gargouri, A. Efficient Graph Algorithms in Securing Communication Networks. *Symmetry* **2024**, *16*, 1269. https://doi.org/10.3390/sym16101269

Academic Editor: Michel Planat

Received: 23 August 2024
Revised: 16 September 2024
Accepted: 23 September 2024
Published: 26 September 2024

1. Introduction

Cryptography, an indispensable tool for secure communication and authentication, encompasses the construction and analysis of systems dedicated to clandestine information exchange. This field is meticulously categorized into symmetric key cryptography, public key cryptography, and hash functions. While symmetric key cryptography employs a single key for encryption and decryption, its counterpart, public key cryptography, utilizes distinct keys for these operations. Of note, symmetric key cryptography, coupled with block ciphering, stands out for its remarkable efficiency and resilience against intrusion, owing much to its foundation in number theory and algebra. Graph theory emerges as a cornerstone in cryptographic endeavors [1,2], facilitating innovative approaches such as utilizing stars, paths, and bipartite graphs for enhanced security. Particularly, bipartite graphs find utility in cryptosystems for seamless data transfer. This synergy between graph theory and cryptography extends to the realm of network security research, where graph methods are deployed to scrutinize cryptographic performance and construct resilient cryptosystems, leveraging graphs with maximal girths. Moreover, cryptography's essence lies in safeguarding data integrity and confidentiality [3], a pursuit deeply ingrained in its transformative process from plain text to cipher text, intelligible solely to authorized recipients. The strategic use of special characters augments security, deterring potential hackers and intermediaries from deciphering encrypted information.

Graph theory-based encryption presents several advantages over traditional cryptographic methods, including enhanced security due to the intricate structures of underlying graphs, which make them more resistant to various attacks, and flexibility in accommodating diverse security needs tailored to specific application contexts. Moreover, graph-based algorithms can provide efficient encryption and decryption, particularly for large datasets. However, while graph theory's versatility allows it to be applied across a range of cryptographic tasks, such as key generation and data encryption, there are notable disadvantages. These include the inherent complexity of implementing graph-based schemes, which can be computationally intensive, efficiency trade-offs between speed and security, and the challenges associated with managing keys in large-scale systems. In essence, cryptography stands as a bulwark against unauthorized access [4,5], fostering secure data exchange across diverse domains, while its integration with discrete mathematics and graph theory heralds a future brimming with novel cryptographic paradigms poised to navigate the evolving landscape of digital security.

The initial message is converted into a code format through a process known as encryption, and the reverse procedure is called decryption [6]. The original contents are shielded against interception by encryption. Plain text is the term for an undisclosed message. By using a designated key, the data known as plain text in any encryption scheme are encoded or encrypted to create a cipher text. Then, through decryption, this cipher text is transformed into a readable original message. A key is a piece of information that is used to encode original data, which are then decrypted to produce actual text. The authorized recipient can easily access the hidden message with the provided key, but an interceptor cannot. Modern cryptography primarily uses two types of schemes: public key cryptography and symmetric key cryptography [7]. Symmetric key cryptography uses one key for both encryption and decryption, whereas public key cryptography uses one key for encryption and another key for decryption. We consider a simple connected graph $G = (V(G), E(G))$ with vertex set $V = V(G)$ and edge set $E(G)$ throughout this paper. For every vertex v that is contained in $V(G)$, the number of edges that intersect v is its degree, expressed as $deg(v)$. The symbols Δ and δ, respectively, represent the maximum and minimum degree of the graph.

A graph that has its vertex set V partitionable into two disjoint subsets, V_1 and V_2, such that xy is an edge if and only if $x \in V_1$ and $y \in V_2$, is called a bipartite graph. K_{mn}, where m and n are the cardinalities of the partite sets, represents a complete bipartite graph. It is easy to note that K_{mn} has $m + n$ vertices and mn edges.

Let $G_1(V_1, E_1)$ and $G_2(V_2.E_2)$ be two graphs with vertex sets V_1 and V_2 and edge sets E_1 and E_2, respectively. The Cartesian product graph of G_1 and G_2, denoted by $G_1 \Box G_2$, is a graph with vertex set $V_1 \times V_2 = \{(x, y) : x \in V_1, y \in V_2\}$, and there is an edge between two vertices, say (x_1, x_2) and (y_1, y_2), if one of the following conditions holds:
(i) Either $x_1 = y_1$ and $x_2y_2 \in E_2$; (ii) Or $x_2 = y_2$ and $x_1y_1 \in E_1$.

Vertex labeling for a graph G is a function of V to a set of labels; a graph that has a defined function of this kind is referred to as a vertex-labeled graph. Similarly, edge labeling is a function from the edge set to the set of labels. A graph with edge labeling is known as an edge-labeled graph. A total labeled graph is defined as a graph that has vertex and edge labeling.

Paper Organization

This paper is organized as follows: Next, Section 2 presents some of the current literature. It also locates and orientates the present work within contemporary research in the field. Section 3 presents the main three schemes. Section 3.1 leverages the Cartesian product of graphs and Sections 3.2 and 3.3 present the two schemes based upon complete bipartite graphs. Each scheme is followed by illustrative example(s) and respective algorithms. Section 4 discusses the obtained results and specifically addresses the aspects of algorithmic temporal and spatial complexity. Finally, Section 5 sums up the whole paper while also indicating some directions of our future work as well as some open problems.

2. Related Work

The exploration of graph theory's applications in cryptography has garnered significant attention from researchers, as evidenced by numerous papers on the subject. Lalitha and Vasu [8] proposed a Python-based encoding and decoding method to convert plain text into cipher text and vice versa. Similarly, Amudha, Sagayaraj, and Sheela [9], along with Chowdhury, Ghosh, and Jana [10], delve into the utility of graph theory in cryptography, highlighting its potential in securing communications.

A focus on utilizing complete graphs and Hamiltonian paths emerges in the works of Mohan, Rajendran, and Rajesh [11], as well as Gurjar and Krishnaa [12]. These papers detail encryption techniques leveraging graph structures for enhanced security and computational efficiency.

Moreover, innovative approaches such as employing Turan graphs in cryptosystems, proposed by Beaula, Venugopal, and Praba in [13], showcase the complexity and robustness achieved through graph-based encryption methods.

Graph theory's integration with image encryption is explored by Zhang et. al. [14], who devised algorithms for generating random Hamiltonian paths within digital images, enhancing security through pixel-bit shuffling and gray-level substitution.

Furthermore, Kumari, in [15], emphasizes the role of cryptography in securing network communications, introducing a modified affine cipher algorithm that encrypts data plotted onto graphs and converted into images, ensuring confidentiality in cloud storage environments.

The authors of [16] underscore the pivotal role of graph theory in modern cryptography, proposing efficient domination and total magic labeling techniques for encrypting and decrypting confidential information.

Lastly, Perera and Wijesiri [17] bridge the connection between graph theory and symmetric cryptography, presenting methodologies that utilize matrices as secret keys to convert plain text into cipher text, enhancing data protection against unauthorized access.

3. Main Work

In this work, complete graphs and Cartesian product graphs are used. These graphs can be useful in cryptographic algorithms due to their structural properties. Every pair of vertices in a complete graph is joined by an edge. To ensure redundancy and robustness in cryptographic techniques, particularly in network design and secure communication, this high degree of connectedness might be helpful. The Cartesian product of two graphs combines the vertex sets and edge sets in a structured way, leading to graphs with predictable properties. This can be useful for designing certain types of cryptographic protocols or systems where predictable structures are advantageous. The structure of these graphs is used in cryptographic algorithms to enhance the computational complexity.

3.1. Data Security Using Cartesian Product Graphs

This cryptographic algorithm uses a Cartesian product graph $G = P_n \Box P_n$ to encrypt plain text of length n. The plain text is first encoded using a standard table, after which the vertices of the graph are labeled with the encoded values. Vertex weights are calculated by adding the labels of the incident edges to the vertex itself. These weights are then organized into a matrix, which is flattened to create the encrypted message. To recover the original plain text, decryption requires reconstructing the matrix and reversing the weight calculations.

3.1.1. Encryption (CPG)

Let K be a plain text of length n and f is a total labeling. This scheme uses the standard encoding table (Table 1). The encryption scheme is stated as follows.

1. Draw encryption Table 1 and let $k_1, \ldots, k_n$ be the numbers corresponding to each letter.
2. Construct the Cartesian product graph $G = P_n \Box P_n$, where n is the length of the word. The vertex set of $P_n \Box P_n$ is

$$V(G) = \{x_{ij} : 1 \leq i, j \leq n\}$$

and edge set

$$E(G) = \{x_{ij}x_{i+1j} : 1 \leq i, j \leq n\} \cup \{x_{ij}x_{ij+1} : 1 \leq i, j \leq n\},$$

respectively.

3. Label the vertices as follows: for $k_j < n$

$$f(x_{ij}) = \begin{cases} k_j & \text{if } i = 1 \\ 0 & \text{if } i \neq 1 \end{cases}$$

otherwise

$$f(x_{ij}) = \begin{cases} \lfloor \frac{k_j}{n} \rfloor + r_j & \text{if } i = 1 \\ \lfloor \frac{k_j}{n} \rfloor & \text{if } i \neq 1, \end{cases}$$

where $0 \leq r_i \leq n - 1$ is a remainder when k_i is divided by n. The $n \times n$ square matrix B is constructed with entries $f(x_{ij})$. It is important to note that the matrix obtained can be singular.

4. As follows, define edge labeling: $f(x_{ij}x_{ij+1}) = ij = f(x_{ij}x_{i+1j})$. It is important to note that $(i+1)j \neq ij + j$.

5. Calculate the weight of each vertex by adding the label of the vertex and all the edges incident on that vertex, i.e.,

$$\begin{aligned} wt(x_{ij}) &= f(x_{ij}) + \sum_{xx_{ij} \in E} f(xx_{ij}) \\ &= f(x_{ij}) + f'(ij) + g(ij) + h(ij) + k(ij) \end{aligned}$$

where

$$f'(ij) = \begin{cases} 0 & \text{if } i = 1 \\ (i-1)j & otherwise \end{cases} \qquad g(ij) = \begin{cases} 0 & \text{if } j = 1 \\ i(j-1) & otherwise \end{cases}$$

$$h(ij) = \begin{cases} 0 & \text{if } i = n \\ ij & otherwise \end{cases} \qquad k(ij) = \begin{cases} 0 & \text{if } j = n \\ ij & otherwise \end{cases}$$

Further explanation: To compute the weight of each vertex x_{ij}, we add:
1. The label of the vertex: $f(x_{ij})$.
2. The weights of all edges incident on the vertex: This involves four terms based on the position of the vertex:

$$\text{wt}(x_{ij}) = f(x_{ij}) + f'(ij) + g(ij) + h(ij) + k(ij)$$

where:

- $f'(ij)$: Adds weight based on the edge to the vertex above x_{ij} (zero if in the top row, $i = 1$).
- $g(ij)$: Adds weight based on the edge to the vertex to the left of x_{ij} (zero if in the first column, $j = 1$).
- $h(ij)$: Adds weight based on the edge to the vertex below x_{ij} (zero if in the bottom row, $i = n$).
- $k(ij)$: Adds weight based on the edge to the vertex to the right of x_{ij} (zero if in the last column, $j = n$).

Each term $f'(ij), g(ij), h(ij), k(ij)$ represents the weight of the respective incident edge, which is zero if the vertex is on a boundary where that edge does not exist.

6. Create the matrix A by filling its entries with the weights of each vertex. In other words, $A = (wt(x_{ij}))_{ij}$.
7. The message is encrypted as follows: $wt(x_{11}), \dots, wt(x_{1n}), wt(x_{21}), \dots, wt(x_{2n}), \dots, wt(x_{n1}), \dots, wt(x_{nn})$.

The encryption algorithm of this scheme is given as Algorithm 1.

Table 1. Standard encoding table.

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

Algorithm 1: CPG Encryption

```
INPUT  : $txt \leftarrow$ plaintext given by user
OUTPUT : flat list of length $(length(txt))^2$
1  $alphabets \leftarrow "ABCDEFGHIJKLMNOPQRSTUVWXYZ"$;
2  $txtlength \leftarrow length(txt)$;
3  $encodedtxt \leftarrow$position of each character within $alphabets$ ;
4  $toprow, otherrows \leftarrow [\ ]$;
5  FOR $i$ from 1 to $txtlength$ DO
6      $toprow \leftarrow toprow.append(\lfloor \frac{encodedtxt(i)}{txtlength} ) \rfloor + remainder(\frac{encodedtxt(i)}{txtlength}))$;
7      $otherrow \leftarrow otherrow.append(quotient(\frac{encodedtxt(i)}{txtlength})))$
8  END
9  $vertexwts \leftarrow Matrix(toprow, (txtlength - 1) copies of otherrows)$;
10 FOR $i$ from 1 to $txtlength$ DO
11     FOR $j$ from 1 to $txtlength$ DO
12         $Amat(i,j) \leftarrow vertexwts(i,j) + \begin{cases} 0 & \text{if } i = 1 \\ (i-1)j & otherwise \end{cases} + \begin{cases} 0 & \text{if } j = 1 \\ i(j-1) & otherwise \end{cases} + \begin{cases} 0 & \text{if } i = n \\ ij & otherwise \end{cases} + \begin{cases} 0 & \text{if } j = n \\ ij & otherwise \end{cases}$
13     END
14 END
15 RETURN flat list of $M_{txtlength \times txtlength}$
```

3.1.2. Decryption (CPG)

1. Message is received as an array of n^2 numbers.
2. Construct the $n \times n$ matrix $A = (a)_{ij}$ by arranging the numbers in rows.
3. Construct the matrix $B = (b)_{ij}$, where the values of b_{ij} are computed by using the following formula:

$$b_{ij} = \begin{cases} a_{ij} - ij - ij & \text{if } i = 1 \ \wedge \ j = 1 \\ a_{ij} - (i-1)j - i(j-1) & \text{if } i = n \ \wedge \ j = n \\ a_{ij} - i(j-1) - ij & \text{if } i = 1 \ \wedge \ j = n \\ a_{ij} - (i-1)j - ij & \text{if } i = n \ \wedge \ j = 1 \\ a_{ij} - i(j-1) - ij - ij & \text{if } i = 1 \ \wedge \ 2 \leq j \leq (n-1) \\ a_{ij} - (i-1)j - ij - ij & \text{if } 2 \leq i \leq (n-1) \ \wedge \ j = 1 \\ a_{ij} - (i-1)j - i(j-1) - ij & \text{if } i = n \ \wedge \ 2 \leq j \leq (n-1) \\ a_{ij} - i(j-1) - (i-1)j - ij & \text{if } 2 \leq i \leq (n-1) \ \wedge \ j = n \\ a_{ij} - i(j-1) - (i-1)j - ij - ij & \text{otherwise} \end{cases} \tag{1}$$

Note may be taken here that the expression ij cannot be simplified as normal terms of an algebraic expression because these are positions determining the values.

4. Add all the entries of the j^{th} column to have an entry $k_j = \sum_{i=1}^{n} b_{ij}$.
5. Find the letter from Table 1 corresponding to each number k_i.
6. The original plain text K is decrypted.

The decryption algorithm of this scheme is given as Algorithm 2.

Algorithm 2: CPG Decryption

```
INPUT  : $a \leftarrow$ flat list of integers, of length $txtlength^2$
OUTPUT : plaintext string
1  Declare variables: $a, txtlength, b$;
2  $a \leftarrow$ reshape $a$ into a square matrix of order $txtlength$;
3  $b \leftarrow$ Empty square matrix of size $= Rank(a)$;
4  FOR $i$ from 1 to $txtlength$ DO
5      FOR $j$ from 1 to $txtlength$ DO
6          IF $i = 1$ AND $j = 1$ THEN
7              $b_{i,j} \leftarrow a_{i,j} - (10 * i + j) - (10 * i + j)$
8          ELSE IF $i = N$ AND $j = N$ THEN
9              $b_{i,j} \leftarrow a_{i,j} - (10 * (-1) + j) - (10 * i + (j - 1))$
10         ELSE IF $i = 1$ AND $j = N$ THEN
11             $b_{i,j} \leftarrow a_{i,j} - (10 * i + j - 1) + (-10 * i - j)$
12         ELSE IF $i = N$ AND $j = 1$ THEN
13             $b_{i,j} \leftarrow a_{i,j} - (10 * (i - 1) + j) + (-10 * i - j)$
14         ELSE IF $i = 1$ AND $2 \leq j \leq (N - 1)$ THEN
15             $b_{i,j} \leftarrow a_{i,j} - ((10 * i + j - 1) + (-10 * i - j)) + (-10 * i - j)$
16         ELSE IF $2 \leq i \leq (N - 1)$ AND $j = 1$ THEN
17             $b_{i,j} \leftarrow a_{i,j} - ((10 * (i - 1) + j) + (-10 * i - j)) + (-10 * i - j)$
18         ELSE IF $i = N$ AND $2 \leq j \leq (N - 1)$ THEN
19             $b_{i,j} \leftarrow a_{i,j} - ((10 * i + j - 1) - (10 * i - j)) - (10 * i - j)$
20         ELSE IF $2 \leq i \leq (N - 1)$ AND $j = N$ THEN
21             $b_{i,j} \leftarrow a_{i,j} - ((10 * i + j - 1) - (10 * (i - 1) - j)) - (10 * i - j)$
22         ELSE
23             $b_{i,j} \leftarrow a_{i,j} - (((10 * i + j - 1) + (-10 * (i - 1) - j)) + (-10 * i - j)) +$
               $(-10 * i - j)$
24     END
25 END
26 $dcypos \leftarrow [\ ]$;
27 FOR $j$ from 1 to $txtlength$ DO
28     $dcypos(j) \leftarrow dcypos.append(\sum(Row_j(bmat)))$
29 END
30 $plaintext \leftarrow$ string of alphabets from inverse image of $dcypos$;
31 RETURN $plaintext$
```

Example 1. *In this example, the CPG algorithm for encryption/decryption is applied on the letter "MATH". The example give the step by step explanation of the encryption/decryption algorithm. The text to be ciphered is "MATH", which by using the standard alphabetic positions, as indicated in Table 1, becomes* 13, 1, 20, 8.

1. *We have* $n = 4,\ k_1 = 13,\ k_2 = 1,\ k_3 = 20,\ k_4 = 8.$
2. *Vertex labeling:*

$$\begin{aligned} f(x_{11}) = 4, f(x_{21}) = f(x_{31}) = f(x_{41}) = 3 \\ f(x_{12}) = 1, f(x_{22}) = f(x_{32}) = f(x_{42}) = 0 \\ f(x_{13}) = f(x_{23}) = f(x_{33}) = f(x_{43}) = 5 \\ f(x_{14}) = f(x_{24}) = f(x_{34}) = f(x_{44}) = 2 \end{aligned}$$

Hence, the required matrix B is constructed as

$$B = \begin{pmatrix} 4 & 1 & 5 & 2 \\ 3 & 0 & 5 & 2 \\ 3 & 0 & 5 & 2 \\ 3 & 0 & 5 & 2 \end{pmatrix}$$

3. *Edge labeling:*

$$\begin{array}{lll} f(x_{14}x_{24}) = 14 & f(x_{14}x_{24}) = 24 & f(x_{34}x_{44}) = 34 \\ f(x_{11}x_{12}) = 11 & f(x_{12}x_{13}) = 12 & f(x_{13}x_{14}) = 13 \\ f(x_{11}x_{21}) = 11 & f(x_{21}x_{31}) = 21 & f(x_{31}x_{41}) = 31 \\ f(x_{21}x_{22}) = 21 & f(x_{22}x_{23}) = 22 & f(x_{23}x_{24}) = 23 \\ f(x_{12}x_{22}) = 12 & f(x_{22}x_{32}) = 22 & f(x_{32}x_{42}) = 32 \end{array}$$

4. *Vertex-wise weight calculations: The vertex and edge labeling of the graph* $P_4 \square P_4$ *is shown in Figure 1*

$$\begin{array}{ll} wt(x_{11}) = 11 + 22 + 4 = 26 & wt(x_{21}) = 11 + 21 + 21 + 3 = 56 \\ wt(x_{12}) = 11 + 1 + 12 + 12 = 36 & wt(x_{22}) = 12 + 0 + 22 + 22 + 21 = 77 \\ wt(x_{13}) = 12 + 5 + 13 + 13 = 43 & wt(x_{23}) = 13 + 5 + 22 + 23 + 23 = 86 \\ wt(x_{14}) = 13 + 2 + 14 = 29 & wt(x_{24}) = 23 + 14 + 2 + 24 = 63 \\ wt(x_{31}) = 21 + 31 + 3 + 31 = 86 & wt(x_{41}) = 31 + 3 + 41 = 75 \\ wt(x_{32}) = 31 + 22 + 0 + 32 + 32 = 117 & wt(x_{42}) = 41 + 42 + 32 + 0 = 115 \\ wt(x_{33}) = 32 + 23 + 33 + 33 + 5 = 126 & wt(x_{43}) = 42 + 33 + 43 + 5 = 123 \\ wt(x_{34}) = 24 + 33 + 2 + 34 = 93 & wt(x_{44}) = 43 + 34 + 2 = 79 \end{array}$$

Thus, the matrix A is obtained as

$$A = \begin{pmatrix} 26 & 36 & 43 & 29 \\ 56 & 77 & 86 & 63 \\ 86 & 117 & 126 & 93 \\ 75 & 115 & 123 & 79 \end{pmatrix}$$

5. *Encrypted message:*
   26, 36, 43, 29, 56, 77, 86, 63, 86, 117, 126, 93, 75, 115, 123, 79 *is our encrypted message.*

Decryption

1. *The encrypted message* 26, 36, 43, 29, 56, 77, 86, 63, 86, 117, 126, 93, 75, 115, 123, 79 *is received.*
2. *Construct matrix A from the encrypted message.*

3. *Using the formula in Equation (1), we calculate*

$$\begin{aligned}
b_{11} &= 26 - 11 - 11 = 4 & b_{21} &= 56 - 11 - 21 - 21 = 3\\
b_{12} &= 36 - 11 - 12 - 12 = 1 & b_{22} &= 77 - 21 - 12 - 22 - 22 = 0\\
b_{13} &= 43 - 12 - 13 - 13 = 5 & b_{23} &= 86 - 22 - 13 - 23 - 23 = 5\\
b_{14} &= 29 - 13 - 14 = 2 & b_{24} &= 63 - 23 - 14 - 24 = 2\\
b_{31} &= 36 - 21 - 31 - 31 = 3 & b_{41} &= 75 - 31 - 41 = 3\\
b_{32} &= 117 - 31 - 22 - 32 - 32 = 0 & b_{42} &= 115 - 32 - 41 - 42 = 0\\
b_{33} &= 126 - 32 - 23 - 33 - 33 = 5 & b_{43} &= 123 - 33 - 42 - 43 = 5\\
b_{34} &= 93 - 33 - 24 - 34 = 2 & b_{44} &= 79 - 34 - 43 = 2
\end{aligned}$$

4. *Obtain the matrix*

$$B = \begin{pmatrix} 4 & 1 & 5 & 2\\ 3 & 0 & 5 & 2\\ 3 & 0 & 5 & 2\\ 3 & 0 & 5 & 2 \end{pmatrix}$$

5. *Using column sums and the reverse of the encoding Table 1, we obtain*

$$\begin{aligned}
k_1 &= b_{11} + b_{21} + b_{31} + b_{41} = 4 + 3 + 3 + 3 = 13 && \mapsto M\\
k_2 &= b_{12} + b_{22} + b_{32} + b_{42} = 1 + 0 + 0 + 0 = \ 1 && \mapsto A\\
k_3 &= b_{13} + b_{23} + b_{33} + b_{43} = 5 + 5 + 5 + 5 = 20 && \mapsto T\\
k_4 &= b_{14} + b_{24} + b_{34} + b_{44} = 2 + 2 + 2 + 2 = \ 8 && \mapsto H
\end{aligned}$$

and hence, the recovered text is M, A, T, H.

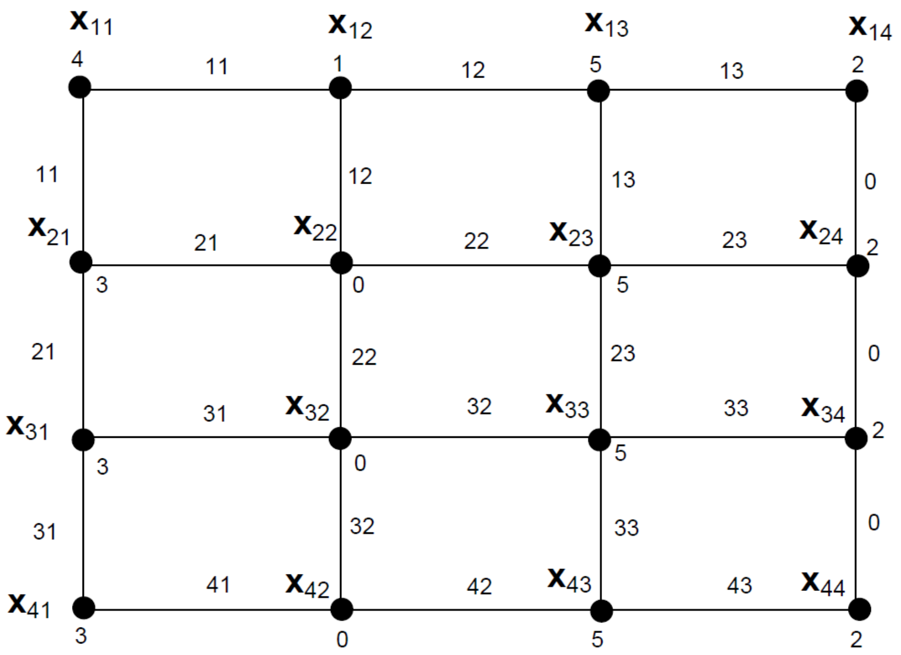

Figure 1. Labeling of vertices and edges of the graph obtained from Algorithm 1 for the word MATH.

3.2. Data Security Using Complete Bipartite Graphs—I

This cryptographic algorithm uses a complete bipartite graph to encode plain text P of length n. The plain text characters are first converted to numerical values by using the Pythagorean table (Table 2) and vertex and edge labeling. To represent each character, a bipartite graph $K_{1,n}$ is created with one vertex representing the character and n vertices in the other set. Vertex and edge labels are assigned based on the character's numerical value and graph positions. The edge weights are used to construct a matrix, which is then multiplied by a symmetric key matrix to yield a new matrix. The final encrypted message is a sequence derived from the remainders of the matrix when divided by n^2. Decryption involves reversing these steps to recover the original plain text using two symmetric keys and solving linear equations.

Valuable contributions in this area may also be seenin [4,18]. Table 2 displays the Pythagorean numbers corresponding to the letters and Table 3 illustrates how the numbers are connected to the letters in order to allow for uniqueness in vertex labeling of a graph.

Table 2. Pythagorean numbers corresponding to each letter.

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

Table 3. Adjusted table used in encryption/decryption scheme.

A	B	C	D	E	F	G	H	I	J	K	L	M
11	21	31	41	51	61	71	81	91	12	22	32	42
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
52	62	72	82	92	13	23	33	43	53	63	73	83

3.2.1. Encryption (CBG-I)

Let P be plain text of length n consisting of letters and f is a total labeling. The encryption scheme is stated as follows.

1. Draw encryption Table 2.
2. Let $h_1, \ldots, h_n$ be the numbers corresponding to each letter using Table 3.
3. Construct n copies of the complete bipartite graph $K_{,n}$. Let $V_{1,i} = \{a_i\}$ $V_{2,i} = \{b_{i,j} : 1 \leq j \leq n\}$ be the two partite sets in the i^{th} copy of $K_{1,n}$.
4. Label the vertices as follows:
   $f(a_i) = h_i$, where $1 \leq i \leq n$.

$$f(b_{ij}) = \begin{cases} \lfloor \frac{h_i}{n} \rfloor + r_i & \text{if } j = 1 \\ \\ \lfloor \frac{h_i}{n} \rfloor & \text{if } j \neq 1, \end{cases}$$

5. Label the edges as follows:
   $f(a_i b_j) = ij$, where $1 \leq i, j \leq n$. Note may be taken here that the expression ij cannot be simplified as normal terms of an algebraic expression as these are positions determining the values.
6. Calculate the weight of each edge to construct the matrix $M = (m)_{ij}$, where

$$m_{ij} = h_i + ij + f(b_{ij})$$

7. Fix the symmetric key $K_1 = (k)_{ij}$, where

$$k_{ij} = \begin{cases} 0 & \text{if } 1 \leq j \leq n \text{ and } j < i < n \\ j - i + 1 & \text{otherwise,} \end{cases}$$

Thus, K_1 is an upper triangular $n \times n$ non-singular matrix.
8. Find matrix $N = M \times K_1$.
9. Divide each entry of the matrix N by n^2 to obtain

$$n_{ij} = n^2 q_{ij} + r_{ij}$$

10. $K_2 = n^2$ is the fixed symmetric key.
11. Construct the matrix $K_3 = (q)_{ij}$ and $R = (r)_{ij}$.
12. For the plain text P, the cipher will be the array of n^2 numbers:

$$r_{11}, \ldots, r_{1n}, r_{21}, \ldots, r_{2n}, \ldots, r_{n1}, \ldots, r_{nn}$$

3.2.2. Decryption (CBG-I)

1. Message is received as an array of n^2 numbers with two keys as matrices $K_1 = (k)_{ij}$ and $K_3 = (q)_{ij}$ and the fixed key $K_2 = n^2$. It is important to note that K_1 and K_2 are fixed and K_3 varies.
2. Find the matrix $N = (n)_{ij}$, where $n_{ij} = n^2 q_{ij} + r_{ij}$.
3. Find K_1^{-1} and use it to obtain the matrix $M = (m)_{ij} = NK_1^{-1}$.
4. Corresponding to each entry m_{ij}, construct the following linear equations:

$$\begin{aligned} m_{ij} &= x_i + y_{ij} + ij \\ x_i &= y_{i1} + y_{i2} + \cdots + y_{in} \\ y_{i2} &= y_{i3} + \cdots + y_{in} \end{aligned}$$

5. By solving the above equations, the values of y_{ij} are

$$y_{ij} = \begin{cases} \frac{2m_{i2} - m_{i1} - i1 - 2}{n+1} & \text{if } j \neq 1 \\ y_{i2} + m_{i1} - m_{i2} + 1 & \text{if } j = 1, \end{cases}$$

Note may be taken here that the expression $i1$ cannot be simplified as normal terms of an algebraic expression because these are positions determining the values.
6. Obtain h_i by adding all y_{ij}, i.e., $h_i = \sum_{i=1}^{n} y_{ij}$.
7. Find the letter from Table 3 corresponding to each number h_i.
8. The original plain text P is decrypted.

For CBG-I, the encryption and decryption algorithms are given as Algorithms 3 and 4, respectively.

Algorithm 3: CBG-I Encryption

```
INPUT  : $txt \leftarrow$ plaintext string from user
OUTPUT : $cypher$ :: list of positive integers
Declare variables :: $toprow,\ otherrows$;
$encodedtxt \leftarrow$ value of each symbol in $txt$ according to Table 3 ;
$toprow,\ otherrows \leftarrow [\ ]$;
FOR $i$ from 1 to $txtlength$ DO
    $toprow \leftarrow toprow.append\left(\lfloor \frac{encodedtxt(i)}{txtlength} \rfloor + remainder\left(\frac{encodedtxt(i)}{txtlength}\right)\right)$;
    $otherrows \leftarrow otherrows.append\left(\lfloor \frac{encodedtxt(i)}{txtlength} \rfloor\right)$
END
$f \leftarrow$ Matrix of $[toprow, (txtlength - 1)$ copies of $otherrows\ ]$;
$Mmat,\ K_1 \leftarrow [\ ]$;
FOR $i$ from 1 to $txtlength$ DO
    FOR $j$ from 1 to $txtlength$ DO
        $Mmat \leftarrow Mmat.append(encodedtxt(i) + ij + f(i,j))$;
        $K_1 \leftarrow if(i < j, 0, i - j + 1)$
    END
END
$Nmat \leftarrow Mmat \times K_1$;
$K_2 \leftarrow txtlength^2$;
$K_3 \leftarrow quotient(Nmat, txtlength^2)$;
$cypher \leftarrow remainder(Nmat, txtlength^2)$;
RETURN $cypher, K_1, K_2, K_3$;
```

Algorithm 4: CBG-I Decryption

```
INPUT  : $cypher, K_1, K_2, K_3$
OUTPUT : plaintext
$Ndcy \leftarrow K_2 \times K_3 + cypher$;
$Mdcy \leftarrow Ndcy \times K_1^{-1}$;
$Bmat \leftarrow [\ ]$;
FOR $i$ from 1 to $\sqrt{K_2}$ DO
    FOR $j$ from $\sqrt{K_2}$ to 1 DO
        $Bmat(i,j) \leftarrow \begin{cases} Bmat(i,2) + Mdcy(i,1) - Mdcy(i,2) + 1 & ;\ if\ j = 1 \\ \frac{2Mdcy(i,2) - Mdcy(i,1) - (10i+1) - 2}{\sqrt{K_2}+1} & ;\ otherwise \end{cases}$
        $encodedtxt \leftarrow \sum_i Row_i(Bmat)$
        $rslt \leftarrow$ Table $3^{-1}(encodedtxt(i))$
    END
END
RETURN $rslt$;
```

The following example explains the encryption and decryption algorithms.

Example 2. *The text is "MAY", which, when coded using Table 3, is* 42, 11, 73, *and the calculations are*

$$\begin{array}{ccc} b_{11} = \lfloor \frac{42}{3} \rfloor = 14 & b_{21} = \lfloor \frac{11}{3} \rfloor + 2 = 5 & b_{31} = \lfloor \frac{73}{3} \rfloor + 1 = 25 \\ b_{12} = 14 & b_{22} = \lfloor \frac{11}{3} \rfloor = 3 & b_{32} = 24 \\ b_{13} = 14 & b_{23} = \lfloor \frac{11}{3} \rfloor = 3 & b_{33} = 24 \end{array}$$

Considering Figure 2, we have

$$M = \begin{pmatrix} 42+11+14 & 42+12+14 & 42+13+14 \\ 11+21+5 & 11+22+3 & 11+23+3 \\ 73+31+25 & 73+32+24 & 73+33+24 \end{pmatrix}$$

$$= \begin{pmatrix} 67 & 68 & 69 \\ 37 & 36 & 37 \\ 129 & 129 & 130 \end{pmatrix}$$

$N = MK_1$, *where* $K_1 = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}$. *Divide each entry by* $n^2 = K_2$. *Find* K_3 *and* R *from* $n_{ij} = q_{ij}n^2 + r_{ij}$. *R is the cipher.*

$$K_3 = \begin{pmatrix} 4 & 1 & 3 \\ 5 & 0 & 3 \\ 6 & 1 & 4 \end{pmatrix}$$

$R = [4,5,6,1,0,1,3,3,4]$ *is the cipher.*

Decryption:

1. $R = [4,5,6,1,0,1,3,3,4]$ *is the received message with key* $[K_1,\ K_2,\ K_3]$.
2. *Obtain the matrix*

$$N = \begin{pmatrix} 67 & 202 & 406 \\ 37 & 110 & 220 \\ 129 & 387 & 775 \end{pmatrix}$$

and

$$M = \begin{pmatrix} 67 & 68 & 69 \\ 37 & 36 & 37 \\ 129 & 129 & 130 \end{pmatrix}$$

3. *Find* $b_{ij} = \dfrac{2m_{12} - m_{i1} - 13}{n+1}$.

$$\begin{aligned} b_{13} = b_{12} &= \frac{2(68) - 67 - 13}{4} = \frac{56}{4} = 14 \\ b_{11} &= b_{12} + 67 - 68 + 1 = 14 - 1 + 1 = 14 \\ b_{22} &= \frac{2m_{22} - m_{21} - 23}{4} = \frac{2(36) - 37 - 23}{4} = 3 \\ b_{21} &= b_{22} - 36 + 37 + 1 = 3 + 2 = 5 \\ b_{32} &= \frac{1}{4}(2(129) - 129 - 31 - 2) = 24 \\ b_{31} &= 24 + 129 - 129 + 1 = 25 \end{aligned}$$

$$\begin{aligned} a_1 &= b_{11} + b_{12} + b_{13} = 14 + 14 + 14 && = 42 && \mapsto M \\ a_2 &= b_{21} + b_{22} + b_{23} = 5 + 3 + 3 && = 11 && \mapsto A \\ a_3 &= b_{31} + b_{32} + b_{33} = 25 + 24 + 24 && = 73 && \mapsto Y \end{aligned}$$

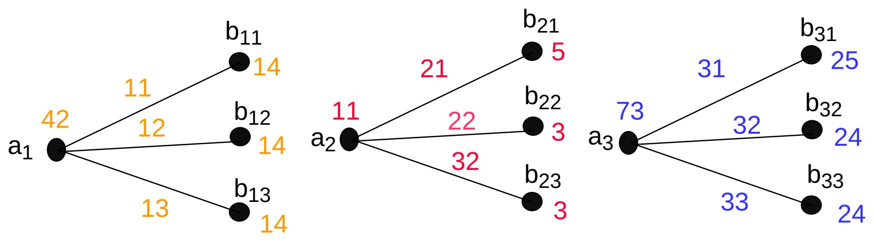

Figure 2. Labeling of vertices and edges of the graph obtained from CBG-I for the word MAY.

3.3. Data Security Using Complete Bipartite Graphs—II

This cryptographic algorithm encodes a plain text message by converting each character to a numerical value from a predefined table. Each value is divided by 10, resulting in two arrays: one containing the quotient and another containing the remainder. A bipartite graph $K_{n,n}$ is created, with vertices and edges labeled based on their values. The encryption is carried out by calculating the weights of specific vertices and creating a cipher text from these weights. To decrypt, the receiver applies a formula to recover the remainders, and then, reconstructs the original numbers. The encryption table is used to find the corresponding characters, allowing the original plain text to be recovered. The detailed algorithm is stated below.

3.3.1. Encryption (CBG-II)

Let the plain text be $K_1, K_2, \ldots, K_n$, where K_i are the letters and n is the length of the plain text. Further, suppose that f is a total labeling. The encryption algorithm is as follows:

1. Draw the encryption Table 2.
2. Write the value of K_i according to the table.
3. Let a_i be the values corresponding to each letter K_i from Table 3.
4. Divide each a_i by 10 and let $a_i = 10q_i + r_i$.
5. Construct the arrays $Q = [q_1, q_2, \ldots, q_n]$ and $R = [r_1, r_2, \ldots, r_n]$.
6. Construct the graph $K_{n,n}$ with partite sets $V_1 = \{b_1, b_2, \ldots, b_n\}$ and $V_2 = \{c_1, c_2, \ldots, c_n\}$.
7. Label the vertices and edges as follows:
   For $1 \le i \le n$
$$f(b_i) = i \quad , \quad f(c_i) = r_i$$
$$f(b_j c_i) = j r_i, 1 \le j \le n$$
8. Compute the weight of each vertex c_i by adding labels of all the edges incident on c_i, i.e.,
$$w(c_i) = \sum f(b_i c_i)$$
9. Construct
$$C = (w(c_1), w(c_2), \ldots, w(c_n))$$

3.3.2. Decryption (CBG-II)

Let $w_1, w_2, \ldots, w_n$ be the received cipher, where each $w_i \equiv 0(mod\ n)$; the decryption method is as follows:

1. Find r_i from the following formula:
$$r_i = \frac{w_i - 5n(n+1)}{n}$$
2. Obtain the array $R = [r_1, r_2, \ldots, r_n]$.
3. Find a_i by using the key $Q = [q_1, q_2, \ldots, q_n]$ from the formula
$$a_i = 10q_i + r_i$$

4. Find the letter from Table 3 corresponding to each number a_i.
5. The plain text $K_1, K_2, \ldots, K_n$ has been deciphered.

The algorithms for the encryption and decryption of CBG-II are given as Algorithms 5 and 6, respectively.

Algorithm 5: CBG-II Encryption

```
INPUT   : $txt \leftarrow$ plaintext given by user
OUTPUT  : $cypher, Q$ : matrix integers
1  $encodedtxt \leftarrow$ value of each symbol in $txt$ according to Table 3 ;
2  $Qarray \leftarrow quotient(encodedtxt, 10)$;
3  $Rarray \leftarrow remainder(encodedtxt, 10)$;
4  $cypher \leftarrow [\ ]$;
5  FOR $i$ from 1 to $txtlength$ DO
6      FOR $j$ from 1 to $txtlength$ DO
7          $rslt \leftarrow 10j + Rarray(i)$
8      END
9      $rslt \leftarrow +rslt$ ;   /* Thesymbol  $\leftarrow +$   denotes cumulative addition */
10     ;
11 END
12 $cypher \leftarrow rslt$;
13 RETURN $cypher, Qarray$
```

Algorithm 6: CBG-II Decryption

```
INPUT   : $cypher, Qarray \leftarrow$ as received
OUTPUT  : $plaintext$ : recovered text string
1  $Rdcy \leftarrow [\ ]]$;
2  FOR $i$ from 1 to $txtlength$ DO
3      $Rdcy \leftarrow Rdcy.append\left(\frac{cypher(i)-5txtlength(txtlength+1)}{txtlength}\right)$;
4      $plaintext(i) \leftarrow$ Table $3^{-1}(10Qarray + Rdcy)$
5  END
6  RETURN $plaintext$
```

The following two examples explain this encryption/decryption scheme.

Example 3. *Using Table 3, the text "KEY" becomes*

$$a_1 = 22,\ a_2 = 51,\ a_3 = 73$$

Using $a_i = 10q_i + r_i$, we have

$$Q = [2, 5, 7]$$
$$R = [2, 1, 3]$$

Consider a complete bipartite graph $K_{3,3}$ with vertex sets $V_1 = \{b_1, \ldots, b_3\}$ and $V_2 = \{c_1, \ldots, c_3\}$ and labeling as

$$f(b_i) = i \qquad f(c_i) = r_i \qquad f(b_i c_i) = 10 \cdot j + r_i$$

The weights of the V_2 set of vertices of $K_{3,3}$ (shown in Figure 3) are calculated as

$$wt(c_1) = 12 + 22 + 32 = 66$$
$$wt(c_2) = 11 + 21 + 31 = 63$$
$$wt(c_3) = 13 + 23 + 33 = 69$$

$$\begin{aligned} C &= [wt(c_1), \dots, wt(c_{13})] \\ &= [66, 63, 69] \end{aligned}$$

Decipher

$$\begin{aligned} r_i &= \frac{w_i - 5n(n+1)}{15} \\ r_1 &= \frac{66 - 5 \cdot 3 \cdot 4}{3} = 2 \\ r_3 &= \frac{63 - 5 \cdot 3 \cdot 4}{3} = 1 \\ r_3 &= \frac{69 - 5 \cdot 3 \cdot 4}{3} = 3 \end{aligned}$$

which gives the deciphered array as $[2, 1, 3]$ *and the public key is* $Q = [2, 5, 7]$. *By using the formula*

$$a_i = 10q_i + r_i$$

one obtains

$$[22, 51, 73] \mapsto KEY$$

Example 4. *Using Table 3, the text "METAMATHEMATICS" becomes*

$$\begin{aligned} &a_1 = 42,\ a_2 = 51,\ a_3 = 23,\ a_4 = 11,\ a_5 = 42,\ a_6 = 11,\ a_7 = 23,\ a_8 = 81 \\ &a_9 = 51,\ a_{10} = 42,\ a_{11} = 11,\ a_{12} = 23,\ a_{13} = 91,\ a_{14} = 31,\ a_{15} = 13 \end{aligned}$$

Using $a_i = 10q_i + r_i$, *we have*

$$\begin{aligned} Q &= [4, 5, 2, 1, 4, 1, 2, 8, 5, 4, 1, 2, 9, 3, 1] \\ R &= [2, 1, 3, 1, 2, 1, 3, 1, 1, 2, 1, 3, 1, 1, 3] \end{aligned}$$

Consider a complete bipartite graph $K_{13,13}$ *with vertex sets* $V_1 = \{b_1, \dots, b_{13}\}$ *and* $V_2 = \{c_1, \dots, c_{13}\}$ *and labeling as*

$$f(b_i) = i \qquad f(c_i) = r_i \qquad f(b_i c_i) = 10 \cdot j + r_i$$

The weights of the V_2 *set of vertices of* $K_{13,13}$ *are calculated as*

$$\begin{aligned} wt(c_1) &= 12 + 22 + 32 + 42 + 52 + 62 + 72 + 82 + 92 + 102 + 112 + 122 + 132 + 142 + 152 = 1230 \\ wt(c_2) &= 11 + 21 + 31 + 41 + 51 + 61 + 71 + 81 + 91 + 101 + 111 + 121 + 131 + 141 + 151 = 1215 \\ &\vdots \\ wt(c_{15}) &= 13 + 23 + 33 + 43 + 53 + 63 + 73 + 83 + 93 + 103 + 113 + 123 + 133 + 143 + 153 = 1245 \end{aligned}$$

$$\begin{aligned} C &= [wt(c_1), \dots, wt(c_{15})] \\ &= [1230, 1215, 1245, 1215, 1230, 1215, 1245, 1215, 1215, 1230, 1215, 1245, 1215, 1215, 1245] \end{aligned}$$

Decipher

$$
\begin{aligned}
r_i &= \frac{w_i - 5n(n+1)}{15} \\
r_1 &= \frac{1230 - 5 \cdot 15 \cdot 16}{15} = 2 \\
&\vdots \\
r_{15} &= \frac{1245 - 5 \cdot 15 \cdot 16}{15} = 3
\end{aligned}
$$

which gives the deciphered array as $[2,1,3,1,2,1,3,1,1,2,1,3,1,1,3]$ *and the public key is* $Q = [4,5,2,1,4,1,2,8,5,4,1,2,9,3,1]$. *By using the formula*

$$a_i = 10q_i + r_i$$

one obtains

$$[42,51,23,11,42,11,23,81,51,42,11,23,91,31,13] \mapsto METAMATHEMATICS$$

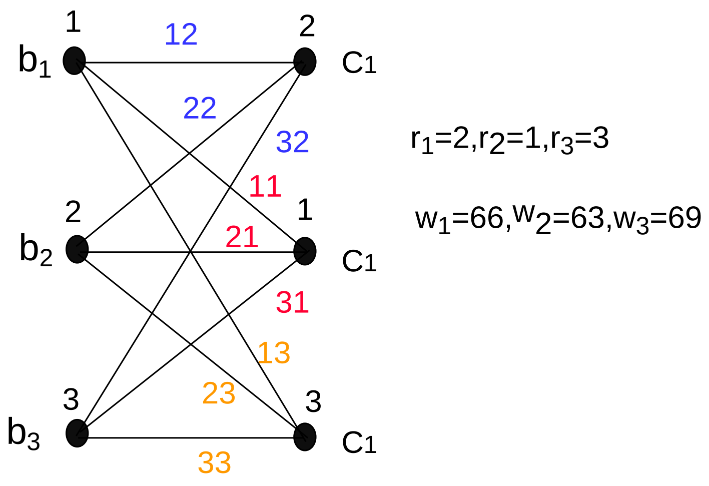

Figure 3. Labeling of vertices and edges of the graph obtained from Algorithm 3 for the word Key.

4. Discussion

The complexity of algorithms, error resistance (in both transmission, and subsequently, the decryption), and key sizes are a few of the sterling metrics to measure the strengths and weaknesses of cryptography systems. The algorithms presented herein were implemented using Computer Algebra System Mathematica 12.0 on an Intel(R) Core(TM) i7-7500U CPU with a clock speed of 2.7 GHz.

4.1. Statistical Analysis

A statistical analysis has also been carried out for these schemes, the results of which are given in Table 4.

The entropy of the plain text is a measure of the uncertainty or randomness of the original data. Generally speaking, higher entropy indicates greater unpredictability in the data. We may evaluate the degree to which the encryption process adds uncertainty by comparing the entropies of the plain text and cipher text. In Table 4, the average entropies of the plain text, cipher text, and key are computed. The results show that the average entropy ratios are 2.545, 3.3872, and 0.3448 for the encryption algorithm for the CPG, CBG-I, and CBG-II schemes, respectively. The analysis reveals that both the CPG and

CBG-I encryption schemes generate cipher text with significantly higher entropy than the plain text, indicating strong security and effective obfuscation of the message. Although CBG-II shows a lower entropy ratio, the incorporation of a robust key compensates for this, ensuring sufficient security. Overall, these encryption schemes demonstrate varying levels of protection, with key strength playing a critical role in enhancing security.

Table 4. Entropy analysis of the encryption schemes for the plain text of length between 100 and 1000.

Encryption	Entropy of Plain Text	Entropy of Cipher Text	Entropy Ratio	Entropy of Key
CPG	4.5566	11.96377778	2.545255556	NA
CBG-I	15.45045833	4.3581	3.387248983	9.407766667
CBG-II	1.57179	4.558676667	0.344853954	3.097433333

The complexity analysis results are given in the next section.

4.2. Time Complexity

For CPG encryption, we observe the growth of execution time with input size. Based on the data (Table 5) and corresponding Figure 4, the execution time seems to grow faster than linearly but not as fast as quadratically. Hence, the complexity lies between linear and quadratic, potentially $\mathcal{O}(n\,log(n))$ or even higher depending on the particular implementation. The major time consuming loop is at lines 10–14 of Algorithm 1 as it contains two **FOR** loops and four **IF–THEN** statements. Similar considerations determine the complexity of CBG-I (Algorithm 3) to be of order $\mathcal{O}(n)$ and CBG-II (Algorithm 5) of order $\mathcal{O}(1)$. While comparing the decryption times [3], CBG-I decryption has the most time-expensive statement at lines 4–10 of Algorithm 4, as it comprises three assignments and one **IF–THEN** statement, which amount to a time complexity of order $\mathcal{O}(n^2)$. Comparatively speaking, CBG-I's decryption is most expensive followed by CPG of order $\mathcal{O}(\sqrt{n})$ and CBG-II of order $\mathcal{O}(1)$. Table 6 summarizes the whole discussion.

Table 5. Time usage data.

	Encryption			Decryption		
n	**CPG**	**CBG-I**	**CBG-II**	**CPG**	**CBG-I**	**CBG-II**
4	0	0	0.016	0	0	0
24	0	0	0	0	0.015	0
80	0.015	0.125	0	0.047	0.469	0
160	0.078	0.437	0.016	0.156	3.656	0
480	1.047	12.282	0.047	1.39	99.656	0
1000	4.547	125.468	0.219	8.031	1253.547	0

A pictorial representation of the time-usage data is shown in Figure 4.

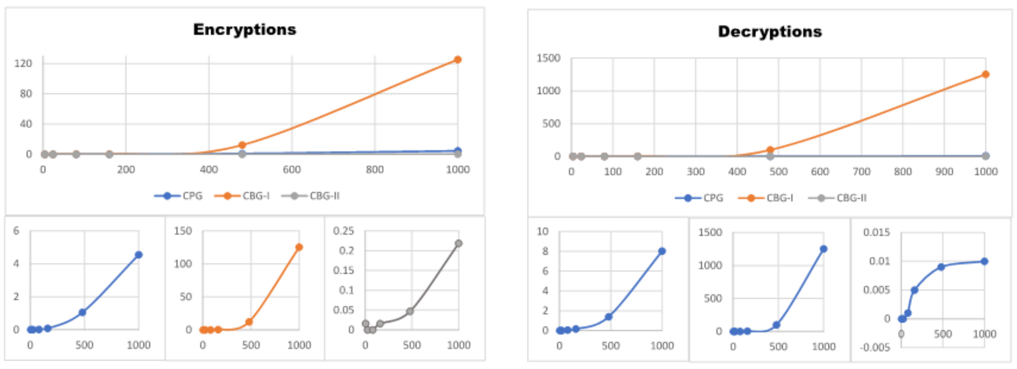

Figure 4. Time complexity plots.

Table 6. Time complexity comparison: CBG-II is clear winner in terms of both encryption and decryption.

Algorithm	Complexity	Major Responsible Part
CBG-II Encryption	$\mathcal{O}(1)$	Lines 5–11: 1 loop, zero decisions and 1 assignment
CBG-II Decryption	$\mathcal{O}(1)$	Lines 2–5: Only 2 assignments
CPG Encryption	$\mathcal{O}(n\ log(n))$	Lines 10–14: 2 FOR loops, 4 decisions
CPG Decryption	$\mathcal{O}(\sqrt{n})$	Lines 4–25: 2 FOR loops, 9 decisions
CBG-I Encryption	$\mathcal{O}(n)$	Lines 10–15: 1 FOR loop, 1 decision
CBG-I Decryption	$\mathcal{O}(n^2)$	Lines 4–10: 3 assignments, 1 decision, populating of B matrix

4.3. Space Complexity

Considering the data (Table 7) and the charts (Figure 5), once again CBG-II consumed much less memory. Its space complexity is of order $\mathcal{O}(1)$. Considering the main loops and decisions involved, the space complexity results are summed up in Table 8.

Table 7. Memory usage data.

	Encryption			Decryption		
n	**CPG**	**CBG-I**	**CBG-II**	**CPG**	**CBG-I**	**CBG-II**
4	27,976	365,128	165,920	16,432	588,248	148,896
24	379,024	732,656	227,952	247,792	2,106,592	150,624
80	3,609,592	10,096,760	527,776	2,400,144	35,762,872	156,000
160	13,999,584	71,819,056	1,238,160	9,327,504	246,394,448	163,680
480	123,039,256	1,777,324,352	7,456,744	82,502,544	5,921,769,112	194,016
1000	527,699,576	16,081,460,432	29,198,304	354,140,808	52,601,997,776	244,032

Space complexity is visualized in Figure 5.

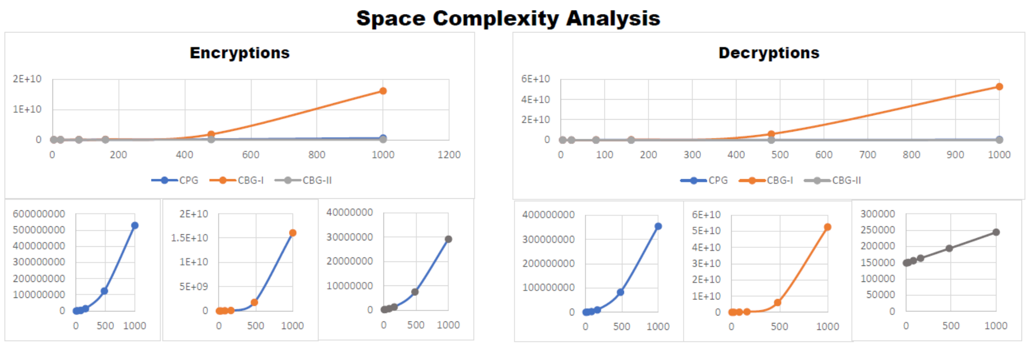

Figure 5. Space complexity plots.

Table 8. Space complexity comparison: CBG-II uses considerably less memory resources, followed by CPG, while CBG-I is the most expensive algorithm.

Algorithm Name	Complexity	Major Responsible Part
CBG-II Encryption	$\mathcal{O}(1)$	Lines 5–11: 1 loop, zero decisions, and 1 assignment
CBG-II Decryption	$\mathcal{O}(n)$	Lines 2–5: Only 2 assignments
CPG Encryption	$\mathcal{O}(\sqrt{n})$	Lines 10–14: 2 FOR loops, 4 decisions
CPG Decryption	$\mathcal{O}(n^2)$	Lines 4–25: 2 FOR loops, 9 decisions
CBG-I Encryption	$\mathcal{O}(n)$	Lines 10–15: 1 FOR loop, 1 decision
CBG-I Decryption	$\mathcal{O}(n^3)$	Populating of matrix B

It is clear from the above complexity data and results that CBG-II is a clear winner in terms of efficiency, followed by CPG and CBG-I. CBG-I, despite being strong in terms of error resistance, is the worst performer, as its decryption module takes a long time and uses a large amount of memory. Key size consideration for these schemes remains of little importance as the keys are symmetric and public in most cases. It may be noted that these schemes have ensured error resistance by repeating certain information in the weight matrices. So, if there is an error, chances are good that it can be recognized and rectified with a high probability of success. While considering the future potential directions, these algorithms may be coupled or extended with compression of matrix schemes. We are currently developing a few such schemes and shall report on them subsequently.

5. Conclusions and Future Directions

In conclusion, cryptography stands as an indispensable tool in fortifying the security landscape of the digital era. This research contributes to the advancement of cryptographic protocols [19–21] through the utilization of encryption and decryption techniques rooted in graph theory. Leveraging Cartesian product graphs and complete bipartite graphs exemplifies the potential for bolstering data confidentiality and integrity within communication systems. Rigorous assessment of time and space complexities corroborates the efficacy of these cryptographic algorithms, thereby laying the groundwork for fortified and secure digital communication channels.

Although data transmission errors are inevitable, the algorithms developed herein demonstrate significant error mitigation capabilities. For example, in Schemes 1 and 3 , the data are encrypted in the form of an array of n^2 numbers. In the decryption process, the entries in each column of the final matrix must be equal except possibly the first entry. If the entries are not equal, the receiver identifies the error. Thus, there is a high probability that the receiver identifies the discrepancies in the received data by using a receiver-driven matrix construction process, which allows self-correction mechanisms to correct any errors. By improving the robustness and dependability of cryptographic schemes, this inherent error-reducing feature advances the state of the art in secure data communication protocols.

Future research should look into several exciting directions for improving these algorithms.

1. Investigate how to improve the encryption and decryption processes for larger datasets. This includes looking into more efficient algorithms or data structures that could reduce computation time and resource usage, making these cryptographic methods suitable for real-world applications with large amounts of data.
2. Integration with modern cryptographic systems: Assess the algorithms' compatibility and security when combined with current cryptographic frameworks and systems. Research could focus on how these methods compare to established standards such as AES or RSA, as well as their potential for hybrid cryptographic systems that combine the strengths of multiple techniques.
3. Investigate these cryptographic algorithms' resilience to potential future quantum computing threats. This includes determining how well the algorithms perform against quantum algorithms, which have the potential to break traditional cryptographic methods. Research could concentrate on adapting these algorithms to be quantum-resistant or developing new cryptographic techniques that ensure security in the face of emerging quantum technologies.

Author Contributions: Conceptualization, S.A.U.H.B. and A.K.; methodology, S.A.U.H.B. and A.K.; software, S.A.U.H.B. and A.K.; validation, S.A.U.H.B., A.K. and F.M.A.S.; formal analysis, S.A.U.H.B. and A.K.; investigation, S.A.U.H.B. and A.K.; resources, F.M.A.S., M.E.E.D. and A.G.; writing—original draft preparation, S.A.U.H.B. and A.K.; writing—review and editing, S.A.U.H.B., F.M.A.S., M.E.E.D., A.G. and A.K.; supervision, S.A.U.H.B.; project administration, S.A.U.H.B.; funding acquisition, F.M.A.S., M.E.E.D. and A.G. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to the Deanship of Research and Graduate Studies at King Khalid University for funding this work through Large Research Project under grant number RGP2/126/45. The authors also express their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, Saudi Arabia for funding this research work through project number "NBU-FFR-2024-1324-05". And this study is supported via funding from Prince Sattam bin Abdulaziz University project number (PSAU/2024/R/1445).

Data Availability Statement: The datasets used and/or analyzed during the current study available from the corresponding author on reasonable request.

Acknowledgments: The authors want to acknowledge the valuable contributions made by the editor and the five anonymous reviewers.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Xue, Y; Chen, L.; Mu, Y.; Zeng, L.; Rezaeibagha, F.; Deng, R.H. Structured encryption for knowledge graphs. *Inf. Sci.* **2022**, *605*, 43–70. [CrossRef]
2. Neamah, A.A.; Shukur, A.A. A Novel Conservative Chaotic System Involved in Hyperbolic Functions and Its Application to Design an Efficient Colour Image Encryption Scheme. *Symmetry* **2023**, *15*, 1511. [CrossRef]
3. Heru, E.K.; Faisal; Pranoto, H. File Encryption Application using Menezes-Vanstone Elliptic Curve Cryptography Based on Python. *Procedia Comput. Sci.* **2023**, *227*, 651–658. [CrossRef]
4. Zhang, H.; Lin, L.; Xu, L.; Wang, X. Graph partition based privacy-preserving scheme in social networks. *J. Netw. Comput. Appl.* **2021**, *195*, 103214. [CrossRef]
5. Senturk, I.; Gursoy, N.K. An Algorithmic Observation of Directed Graphs on Lattices. *J. Int. Math. Virtual Inst.* **2022**, *12*, 17–32. [CrossRef]
6. Rosen, K.H. *Elementary Number Theory and Its Applications*, 5th ed.; Addison-Wesley: Boston, MA, USA, 2005.
7. Stinson, D.R. *Cryptography: Theory and Practice*, 4th ed.; Chapman and Hall/CRC: Boca Raton, FL, USA, 2018.
8. Lalitha, M.; Vasu, S. A Study On Graph Theory in Cryptography Using Python. *J. Eng. Technol. Innov. Res.* **2023**, *10* .
9. Amudha, P.; Sagayaraj, A.C.C.; Sheela, A.C.S. An Application of Graph Theory in Cryptography. *Int. J. Pure Appl. Math.* **2018**, *119*, 375–383.
10. Chowdhury, S.; Ghosh, P.; Jana, M. An Approach of Graph Theory for Solving Cryptographic Problem. *BKGC Sch.* **2020**, *1*, 64–68.
11. Mohan, P.; Rajendran, K.; Rajesh, A. An Encryption Technique Using a Complete Graph with a Self-Invertible Matrix. *J. Algebr. Stat.* **2022**, *13*, 1821–1826.
12. Gurjar, D.K.; Krishnaa, A. Complete Graph and Hamiltonian Cycle in Encryption and Decryption. *Int. J. Math. Trends Technol.* **2021**, *67*, 62–71. [CrossRef]
13. Beaula, C.; Venugopal, P.; Praba, B. Block Encryption and Decryption of a Sentence Using Decomposition of the Turan Graph. *J. Math.* **2023**, *2023*, 7588535. [CrossRef]
14. Zhang, W.; Wang, S.; Han, W.; Yu, H.; Zhu, Z. An Image Encryption Algorithm Based on Random Hamiltonian Path. *Entropy* **2020**, *22*, 73. [CrossRef] [PubMed]
15. Kumari, M.; Kirubanad, V. Data Encryption And Decryption Using Graph Plotting. *Int. J. Civ. Eng. Technol.* **2018**, *9*, 36–46.
16. Meenakshi, A.; Kannan, A.; Cep, R.; Elangovan, M. Efficient Graph Network Using Total Magic Labeling and Its Applications. *Mathematics* **2023**, *11*, 4132. [CrossRef]
17. Perera, P.A.S.D.; Wijesiri, G.S. Encryption and Decryption Algorithms in Symmetric Key Cryptography Using Graph Theory. *Psychol. Educ. J.* **2021**, *58*, 3420–3427. [CrossRef]
18. Wu, Y.; Chen, L. Structured encryption for triangle counting on graph data. *Future Gener. Comput. Syst.* **2023**, *145*, 200–210. [CrossRef]
19. Huang, Z.; Lai, J.; Chen, W.; Li, T.; Xiang, Y. Data security against receiver corruptions: SOA security for receivers from simulatable DEMs. *Inf. Sci.* **2019**, *471*, 201–215. [CrossRef]
20. Das, R.; Khan, A.; Arya, R.; Ilkhom, B.; Bakhtiyor, A.; Safoyev, N.; Khudoykulov, Z. SSKA: Secure symmetric encryption exploiting Kuznyechik algorithm for trustworthy communication. *Int. J. Syst. Assur. Eng. Manag.* **2024**, *15*, 2391–2400. [CrossRef]
21. Gursoy, A.; Gursoy, N.K.; Oner, T.; Senturk, I. An alternative construction of graphs by associating with algorithmic approach on MV-algebras. *Soft Comput.* **2021**, *25*, 13201–13212. [CrossRef]