Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 28 (2025), No. 1, pp. 143–159
DOI : 10.47974/JDMSC-2101

Exploring secure communication through artistic graph creation

H. R. Medini [†]
Sabitha D'Souza *
C. Devadas Nayak [§]
Pradeep G. Bhat [‡]
Department of Mathematics
Manipal Institute of Technology
Manipal Academy of Higher Education
Manipal 576104
Karnataka
India

Abstract

Creating art through mathematics can provide a unique way to communicate complex ideas, patterns and concepts in an aesthetically pleasing manner. Translating complex data into a visual representation is a valuable way to communicate information effectively. Also, secured communication is a highly challenging task in the contemporary world. There will always be a continuous search for better algorithms to prevent data from breaching. This paper aims to transform each word into unique artistic graphs, facilitating communication through visually appealing representation. A novel algorithmic art approach is introduced to communicate securely. Graph labeling and generalized complement of a graph are used in encryption and decryption. The proposed algorithm uses vertex even mean labeling and k(i)-complement of a graph to convert plaintext into cipher graph and vice versa. This algorithm is a symmetric key algorithm as both encryption and decryption use the same cryptographic key and is applicable for all plaintexts including special characters.

Subject Classification: *(2010) 05C78, 05C85, 05C90, 90C35.*

Keywords: *Encryption and decryption, Communication, Message, Graph labeling, Generalized complement.*

[†] *E-mail:* **medinihr@gmail.com**
* *E-mail:* **sabitha.dsouza@manipal.edu (Corresponding Author)**
[§] *E-mail:* **devadas.nc@manipal.edu**
[‡] *E-mail:* **pg.bhat@manipal.edu**

1. Introduction

Mathematics explores structures, patterns and relationships in which graph theory has existed for ages. Art is everywhere around us. It surrounds us in every aspect of life. Within the realm of graph theory, graphs can be viewed as artistic expressions. It can be quite beautiful and intricate, often resembling patterns that can be seen in traditional art forms like Rangoli or Pookalam, which are colourful and decorative designs created on the ground using coloured powders or flower petals, especially during festive occasions in India. The bridge between graph theory and art lies in their structures, patterns and interconnections.

The graphs of graph theory bear resemblance to the designs of rangoli art, where the vertices serve as the points and the edges symbolize the lines or curves observed in rangoli patterns. The algorithm produces distinct cipher graphs for each plaintext which resemble rangoli. This comparison highlights the artistic intricacies embedded within the encrypted structure, introducing a creative artistic dimension to secure digital communication.

Graph theory acts as a bridge between the realms of mathematics and art. The origins of graph theory date back to 1735, when Euler resolved the Königsberg bridge problem. Graph theory is a branch of mathematics focused on the study of graphs, which are structures made up of vertices (points) and edges (lines connecting those points). These graphs can be used to model and analyze relationships, connections, and patterns in various systems. The versatility of graphs goes beyond their aesthetic appeal, as they find applications in diverse fields, including computer science and programming to represent networks, social connections, data structures, and algorithms. It finds many applications in cryptography [9, 7, 6]. In 2021, Kapoor and Gupta [11] investigated the integration of symmetric and asymmetric cryptographic algorithms to enhance data security in web communications by utilizing Blowfish, RC6 and RSA algorithms, the proposed hybrid system encrypts data with symmetric methods and secures the key with an asymmetric algorithm. Sangwan et al. [2] explored the methods to enhance the security and efficiency of communication within volunteer cloud computing environments.

The concept of graph labeling was introduced by A.Rosa [1] in 1960. Almost all the graph labeling methods are incorporated in the Gallian survey of graph labeling [5]. Somasundaram and Ponraj [10] introduced the mean labeling of graphs in 2003. Mean labeling was later developed into various forms, including vertex even mean, vertex odd mean, super

mean, k-super mean labelings and other related forms. For the definitions of mean labeling, vertex even mean labeling (VEML) and vertex odd mean labeling (VOML), refer to the Gallian survey [5]. In 2015, N Revathi [8] investigated VEML and VOML for certain graphs. The proposed algorithm applies VEML for coding plaintexts as it is most suitable for obtaining cipher graphs.

A graph G is defined as a vertex even mean graph if there exists an injective function $f: V(G) \longrightarrow \{2, 4, 6, \ldots, 2|E(G)|\}$, such that the edge values are calculated by taking the mean of their adjacent vertices. This function f is referred to as VEML [5].

The notion of the generalized complement of a graph was introduced by Sampathkumar and Pushpalatha [3] and they named it as k-complement of a graph. It is denoted as G_k^P. Another generalization of the complement of a graph was also found by Sampathkumar et al. [4] and it was named as k(i)-complement of a graph.

The k(i)-complement of G is obtained by removing the edges that exist within each partite in G and adding the edges between those vertices which are absent in G. It is denoted as $G_{k(i)}^P$.

The k(i)-complement of a graph is used for encryption and decryption. The generalized complement is an emerging area of research. The proposed work finds its application in cryptography for secure communication. In this paper, the graph labeling method is used for the encryption of plaintexts which includes the assignment of numerical values for the data and the retention of these values as edge labels in the general graph G. The main purpose of the generalized k(i)-complement of a graph is to add and delete additional independent edges in the process of encryption and decryption, respectively. The idea of adding these independent edges is to make the process of decryption cumbersome. The partition of the vertex set and keys sent by the sender are used to decode the message. The general graph can be used for all possible plaintexts which helps to generate different cipher graphs. Symmetric key cryptography is applied since the same keys are used for both encryption and decryption processes.

Example: Consider C_5 (Fig. 1). Let $P = \{V_1, V_2\}$, where $V_1 = \{1, 4, 5\}$ and $V_2 = \{2, 3\}$. Then, k-complement G_k^P (Fig. 2) and k(i)-complement $G_{k(i)}^p$ (Fig. 3) of C_5 are shown below.

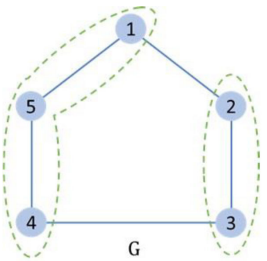


Figure 1
Cycle C_5

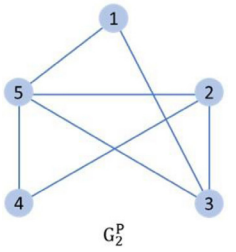


Figure 2
k-complement of G

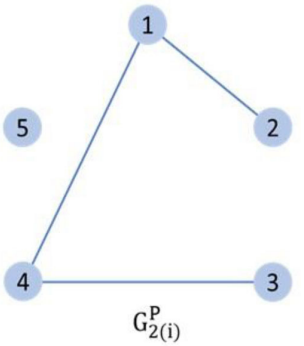


Figure 3
k(i)-complement of G

2. Proposed work

A new multi-level crypto-system is presented that employs VEML and generalized k(i)-complement of a graph where each word is transformed into distinctive graphical representations.

By delving into the applications of graph theory, the potential of utilizing diverse graph structures for the encryption and decryption of text is unravelled, thereby presenting a compelling and aesthetically engaging avenue for digital communication. The process of encrypting plaintext starts with the encoding table.

2.1 *Encoding Table*

The characters provided on the 'QWERTY' keyboard layout are considered for assigning values in the table. The standard keyboard layout

typically consists of a total of 69 characters which includes 26 letters (A-Z), 1 space, 10 digits (0-9) and 32 unique special characters. So, 69 characters are considered in Table 1. The numbers from 1 to 26 are assigned to English alphabets and the space between the words is assigned the label 27. The numerals 0,1,2,...,9 are assigned values from 28 to 37. The special characters are assigned the numbers from 38 to 69.

One can easily get a numerical representation of the given plaintext using Table 1. These values are taken as the first encrypted values (FEV). The distinct values of the first encrypted set of values represent the edge values of the cipher graph, which is explained in subsection 2.2.

Table 1

Encryption of plaintext characters

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

V	W	X	Y	Z	space	0	1	2	3	4	5	6	7	8	9	~	`
22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39

!	@	#	$	%	^	&	*	(	)	_	-	+	=	{	}	[	]	|
40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58

\	:	;	“	’	<	>	,	.	?	/
59	60	61	62	63	64	65	66	67	68	69

2.2 *Creation of a general graph*

The vertices of the general graph *G* are assigned even numbers from 0 to 76 as shown in Fig. 4. The edges are drawn in such a way that the vertices 0 and 76 are adjacent to all the other vertices. The edge values ranging from 1 to 75 are obtained through VEML. The resultant graph is a general graph which can be applied to all sorts of plaintexts. It is evident from Table 1 that 69 characters are assigned the values. So, *G* should consist of at least 69 edges. If more characters are needed, one can extend *G* by adding extra vertices and edges. The general graph given here consists of 75 edges and the remaining values can be used for additional characters if needed.

2.3 Procedure to obtain cipher graphs

The process for generating a cipher graph from a plaintext is outlined below.

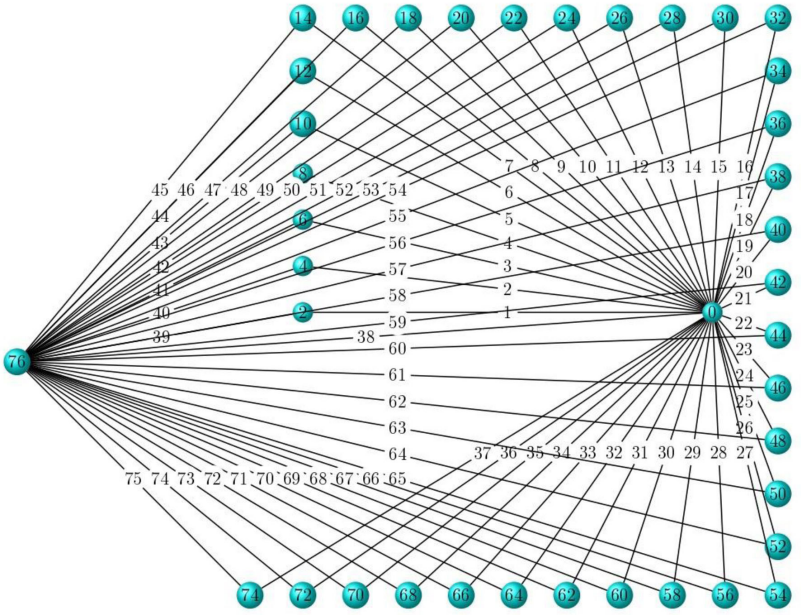


Figure 4

General graph

Let x represent any edge value.

Case 1: x < 39.

That is, if 2x < 78, then draw an edge from '0' to 2x.

Case 2: x ≥ 39.

That is, if 2x ≥ 78, then draw an edge from the vertex '76' to 2x-76.

Thus, the generated graph is referred to as cipher graph 1. In order to have highly secure communication, the k(i)-complement of a graph is used. The resultant graph is referred to as cipher graph 2.

The following steps are followed while partitioning the vertex set of *G*.

a) Vertices 0 and 76 are considered as singleton sets in *P*.

b) Every partite is independent and of order two except for vertices 0 and 76. The independent partite of order two is chosen such that the two vertices have minimum vertex labels.

c) If the number of vertices between 0 and 76 is odd, then one of the maximum labeled vertices will be taken as a singleton set in *P*.

2.4 *Procedure for encryption and decryption*

Encryption: Every character of a given plaintext is converted into numerical value by using Table 1. The sequence of these numbers serves as FEV. A set of distinct values is selected from FEV. The cipher graph corresponding to the given plaintext is derived from the general graph, with the distinct numerals considered as edge values within the general graph. The cipher graph 1 is obtained by following Procedure 2.3. The extra independent edges added to cipher graph 1 with the help of k(i)-complement of a graph enhances the level of difficulty in decoding. Cipher graph 2 is obtained by following Procedure 2.3.

Key generation:

Key 1: The unique numerals of order n obtained from FEV are arranged in increasing order and assigned consecutive numbers starting from 1. This sequence of positions is taken as the second encrypted values (SEV) of the plaintext. Key 1 is generated by assigning the SEV to FEV.

Key 2: It is important to recognize upper and lowercase letters in plaintext. The values 1 and 0 are assigned for the uppercase and lowercase letters, respectively. The sequence of 1 and 0 provides key 2.

The vertex labeled cipher graph 2, its partition and the keys are sent to the receiver.

Decryption: The receiver needs to find $G^{p}_{k(i)}$ of the vertex labeled cipher graph as per the partition sent by the sender. This process will remove the additional edges, resulting in cipher graph 1. VEML is then used to determine the edge values of cipher graph 1.

The edge values are then arranged in increasing order and assigned sequential numbers. Key 1 is used to decipher the numbers based on the plaintext order, mapping them to characters using Table 1. Key 2 is employed to denote uppercase/lowercase alphabets, ultimately reconstructing the plaintext.

The conceptual framework of the proposed algorithm is depicted in Fig. 5.

2.5 *Flow chart*

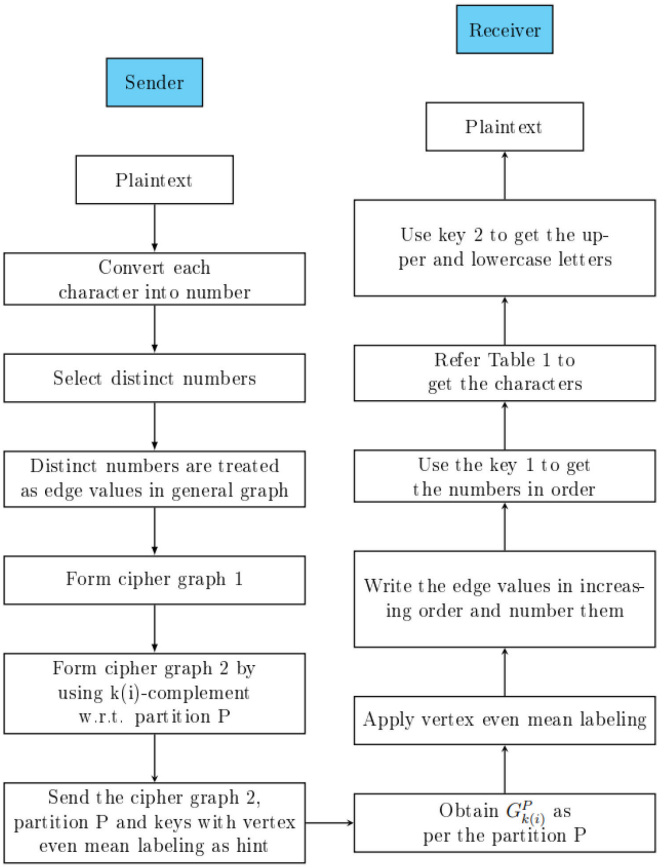


Figure 5

Conceptual framework

2.6 *Proposed algorithm*

Encryption algorithm:

Step 1: Convert each character in the plaintext to a numerical value by using Table 1.

Step 2: Select the distinct numbers (omit repeated numbers if any), express them in ascending order and follow Procedure 2.4 for key generation.

Step 3: As the distinct values correspond to the edge values in G, extract the graph that contains only those edges and name it as cipher graph 1.

Step 4: Follow Procedure 2.3 to obtain a partition P of the vertex set of cipher graph 1 to insert extra independent edges in cipher graph 2.

Step 5: Send the vertex labeled cipher graph 2, its partition P and keys to the receiver.

Decryption algorithm:

Step 1: Apply k(i)-complement of the graph with the help of the received partition set *P.*

Step 2: Calculate the edge values by applying VEML.

Step 3: Arrange the edges in increasing order of their values and assign them sequential numbers.

Step 4: Sort edge values using key 1.

Step 5: Use Table 1 to convert the numerical values obtained in Step 4 into their corresponding character representations.

Step 6: Use key 2 to transform letters into uppercase or lowercase.

Step 7: Plaintext will be generated.

3. Illustration

3.1 *Example 1*

The illustration of the algorithm when the plaintext consists of alphabets only is given below.

Steps involved in the encryption process:

Step 1: Consider the plaintext to be **Mathematics**.

Obtain the first encrypted values using Table 1.

Plaintext	M	a	t	h	e	m	a	t	i	c	s
First encrypted values	13	1	20	8	5	13	1	20	9	3	19

Step 2: Consider only the distinct encrypted values and arrange them in increasing order.

1	3	5	8	9	13	19	20

Generation of keys: Assign the numbers from 1 to 8 to the ordered 8 distinct values, referred to as SEV.

Distinct values	1	3	5	8	9	13	19	20
Second encrypted values	1	2	3	4	5	6	7	8

First, write the plaintext along with its first encrypted values. Generate Key 1 by mapping the SEV to FEV of the plaintext. Generate Key 2 by assigning the value 1 and 0 to uppercase and lowercase letters, respectively.

Plaintext	M	a	t	h	e	m	a	t	i	c	s
First encrypted values	13	1	20	8	5	13	1	20	9	3	19
Key 1	6	1	8	4	3	6	1	8	5	2	7
Key 2	1	0	0	0	0	0	0	0	0	0	0

Step 3: By referring to *G*, obtain cipher graph 1 (Fig. 6) for the chosen plaintext.

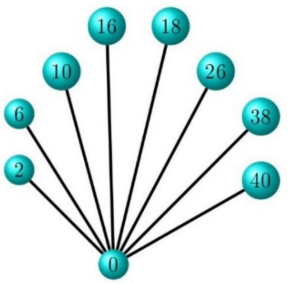


Figure 6

Cipher graph 1

Step 4: Follow Procedure 2.3 to add extra independent edges to cipher graph 1 to obtain cipher graph 2 cipher graph 2 (Fig. 7).

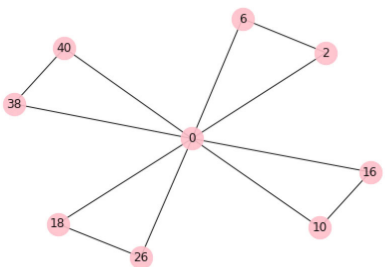


Figure 7

Cipher graph 2

Step 5: $P = [\{0\}, \{2, 6\}, \{10, 16\}, \{18, 26\}, \{38, 40\}]$
Send the cipher graph 2, partition P and keys to the receiver.

Steps involved in the decryption process:

Step 1: Obtain $G^{p}_{k(i)}$ using the partition P and apply VEML to get edge values to get edge values as shown in Fig. 8.

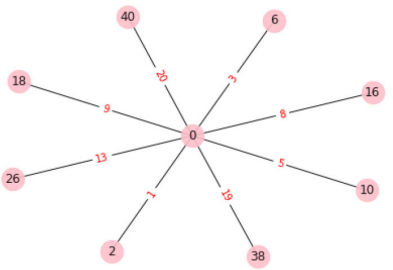


Figure 8
Decoded cipher graph

Step 2: Rearrange the edge values in ascending order and number them from 1 to 8.

Edge values in ascending order	1	3	5	8	9	13	19	20
Positions	1	2	3	4	5	6	7	8

Step 3: By using key 1 and the table in step 2, get the plaintext values as follows:

Key 1	6	1	8	4	3	6	1	8	5	2	7
Plaintext values	13	1	20	8	5	13	1	20	9	3	19

Step 4: Use Table 1 to get the characters of the plaintext.

13	1	20	8	5	13	1	20	9	3	19
M	A	T	H	E	M	A	T	I	C	S

Step 5: Use Key 2 to identify uppercase and lowercase characters.

Key 2	1	0	0	0	0	0	0	0	0	0	0
Plaintext	M	a	t	h	e	m	a	t	i	c	s

Step 6: The plaintext will be generated as **Mathematics**.

3.2 *Example 2*

In this example, the plaintext includes special characters.

Encryption:

Step 1: Consider the plaintext to be **P@$(@|'$ Tr!@ng|e**

Replace each character of the plaintext with numbers using Table 1.

Plaintext	P	@	$	(	@	|	'	$		T	r	!	@	n	g	|	e
First encrypted values	16	41	43	48	41	58	63	43	27	20	18	40	41	14	7	58	5

Step 2: Re-write distinct values in increasing order.

5	7	14	16	18	20	27	40	41	43	48	58	63

Follow the procedure for generating keys.

Distinct values	5	7	14	16	18	20	27	40	41	43	48	58	63
Second encrypted values	1	2	3	4	5	6	7	8	9	10	11	12	13

Write the SEV in the order of the plaintext by considering the FEV as follows.

Plaintext	P	@	$	(	@	|	'	$		T	r	!	@	n	g	|	e
First encrypted values	16	41	43	48	41	58	63	43	27	20	18	40	41	14	7	58	5
Numbers from second encrypted values	4	9	10	11	9	12	13	10	7	6	5	8	9	3	2	12	1

Obtain key 1 and key 2 as follows:

Key 1	4	9	10	11	9	12	13	10	7	6	5	8	9	3	2	12	1
Key 2	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Step 3: By referring to *G*, get the cipher graph 1 as shown as shown in Fig. 9.

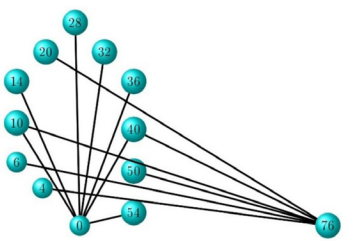


Figure 9

Cipher graph 1

Step 4: Add the extra independent edges by following Procedure 2.3 and obtain cipher graph 2 as shown in Fig. 10.

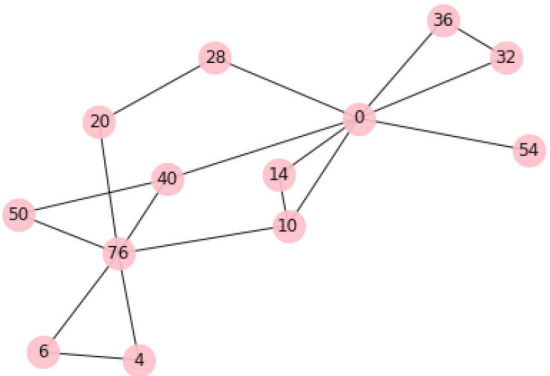


Figure 10

Cipher graph 2

Step 5: P = [{0}, {4, 6}, {10, 14}, {20, 28}, {32, 36}, {40, 50}, {54}, {76}]

Send the cipher graph 2 (Fig. 10), keys and partition P to the receiver. keys and partition P to the receiver.

Decryption:

Step 1: Find $G^{p}_{k(i)}$ and apply vertex even mean labeling for the vertex labeled graph as shown in Fig. 11.

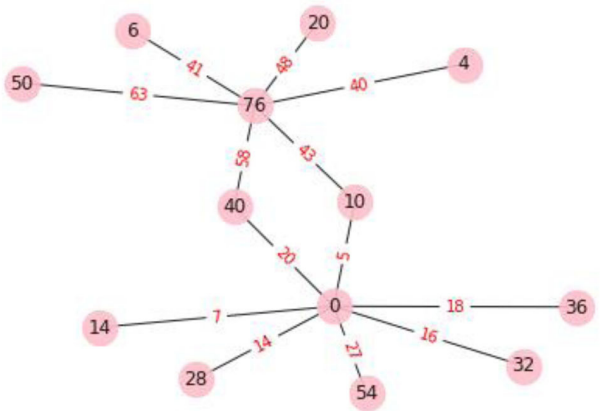


Figure 11

Decoded cipher graph

Step 2: Express the edge values in ascending order and number them.

5	7	14	16	18	20	27	40	41	43	48	58	63
1	2	3	4	5	6	7	8	9	10	11	12	13

Step 3: By using key 1 and from the previous table, get the plaintext values as follows:

Key 1	4	9	10	11	9	12	13	10	7	6	5	8	9	3	2	12	1
Plaintext values	16	41	43	48	41	58	63	43	27	20	18	40	41	14	7	58	5

Step 4: Get the characters of the plaintext using Table 1.

16	41	43	48	41	58	63	43	27	20	18	40	41	14	7	58	5
P	@	$	(	@	|	’	$		T	R	!	@	N	G	|	E

Step 5: Identify uppercase and lowercase letters using key 2.

Key 2	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Plaintext	P	@	$	(	@	|	’	$		T	r	!	@	n	g	|	e

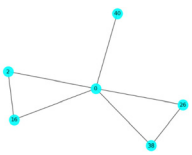
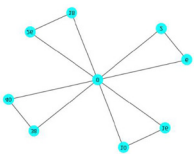
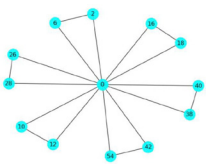
Step 6: The plaintext will be generated as **P@$(@|’$ Tr!@ng|e**.

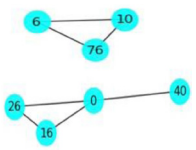
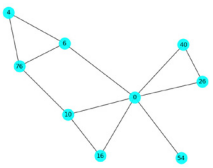
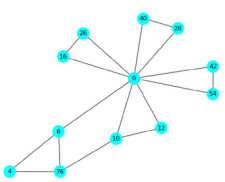
It can be seen from the examples that the cipher graph created for the plaintext **P@$(@|’$ Tr!@ng|e** (with special characters) is a bit more complex than the cipher graph of the plaintext **Mathematics** (without special characters). The examples also show how each word is transformed into different graph structures, reflecting artistic patterns.

4. Experimental analysis

The proposed algorithm is implemented using Python Jupyter Notebook. Program running time for encryption and decryption of different plaintexts when executed in an i3 processor with 4GB RAM are shown in Table 2. The first table shows plaintext without using special characters in which running time increases with the size of the plaintext and the second table shows plaintext with special characters where running time increases as the size of the plaintext increases. However, cipher graphs with complex structures will be generated.

Table 2
Comparison of plaintext running time

Sl. no.	Plaintext without special characters	Plaintext size	Cipher graph	Program running time (ms)
1	Maths	5		4.2183
2	Mathematics	11		5.4040
3	Mathematics Is Fun	18		10.6349

Sl. no.	Plaintext with special characters	Plaintext size	Cipher graph	Program running time (ms)
1	M@th$	5		14.0251
2	M@them@t!c$	11		21.5920
3	M@them@t!c$!$ Fun!	19		34.5888

4.1 *Time complexity and space complexity*

The time complexity and space complexity of the code can be approximated as $O(V + E + n' + P + m \log m + p)$ and $O(n' + m + E + V + P)$, respectively, where n' is the length of an array list, P is the number of elements in the partition list, m is the number of unique edge values in the list, p is the length of the array that rearranges the list of edge values, V is the number of vertices and E is the number of edges in the graph.

5. Conclusion and future work

The intersection of mathematics, computer science and art leads to fascinating discoveries and innovative applications in various domains. The proposed work provides a new top-secret data sharing algorithm by applying graph labeling and a generalized complement of a graph. The complexity of the proposed algorithm is enhanced by using the generalized complement method which helps achieve greater security. Without the knowledge of k(i)-complement of a graph, it will be difficult for hackers to decode it, thereby giving secure communication. Thus, a novel method of communication is created that uses various aspects of graph theory, enabling the transformation of each word into distinct graphical representations. The proposed algorithm can be applied to all plaintexts including special characters. This work emphasizes the use of generalized complements of a graph and VEML in cryptography.

Nowadays, many mobile applications use end-to-end encryption for secure communication. The proposed algorithm is more secure and private and it can be used for message communication since the algorithm is complex and has less running time. It also has applications in digital money transactions to protect the users' data, such as account numbers, transaction amounts and digital signatures.

The integration of graph theory principles into digital communication has opened up new avenues for creative and secure message transmission. By converting words into unique graphical structures, the potential for artistic and innovative communication strategies is demonstrated. The utilization of graph theory in encoding and decoding messages has not only enhanced security but has also introduced a visually appealing dimension to the process.

References

[1] A. Rosa, "On certain valuations of the vertices of a graph," *Theory of Graphs (Internat. Symposium, Rome)*, pp. 349-355 (1966).

[2] A. Sangwan, B. Keswani and A. Sharma, "Optimization of secure communication channel over the volunteer cloud and cloud computing," *Journal of Discrete Mathematical Sciences and Cryptography,* vol. 24, no. 8, pp. 2257-2266 (2021).

[3] E. Sampathkumar and L. Pushpalatha, "Complement of a graph: A generalization," *Graphs and Combinatorics*, vol. 14, no. 4, pp. 377-392 (1998).

[4] E. Sampathkumar, L. Pushpalatha, C. V. Venkatachalam and P. Bhat, "Generalized complements of a graph," *Indian Journal of Pure and Applied Mathematics,* vol. 29, no. 6, pp. 625-639 (1998).

[5] J. A. Gallian, "A dynamic survey of graph labeling," *Electronic Journal of Combinatorics,* vol. 1 (DynamicSurveys), pp. DS6 (2018).

[6] H. R. Medini, S. D'Souza, D. Nayak C and P. G. Bhat, "Encoding and decoding of messages using graph labeling and complement of a graph," *Global and Stochastic Analysis*, vol. 11, no. 1, pp. 1-13 (2024).

[7] H. R. Medini, S. D'Souza, D. Nayak C and P. G. Bhat, "Multifaceted coding of messages using the concepts of graph theory," *IAENG International Journal of Computer Science*, vol. 51, no. 2, pp. 143-153 (2024).

[8] N. Revathi, "Vertex odd mean and even mean labeling of some graphs," *IOSR Journal of Mathematics*, vol. 11, no. 2, pp. 70-74 (2015).

[9] P. L. K. Priyadarsini, "A survey on some applications of graph theory in cryptography," *Journal of Discrete Mathematical Sciences and Cryptography,* vol. 18, no. 3, pp. 209-217 (2015).

[10] S. Somasundaram and R. Ponraj, "Mean labelings of graphs," *National Academy Science Letters*, vol. 26, no. 7-8, pp. 210-213 (2003).

[11] V. Kapoor and R. Gupta, "Hybrid symmetric cryptography approach for secure communication in web application," *Journal of Discrete Mathematical Sciences and Cryptography,* vol. 24, no. 5, pp. 1179-1187 (2021).

Received January, 2024