Application of Graph Complements and Labeling in Secure Communication

Naik Sinchana Ganapathi, Medini H.R., and Sabitha D'Souza*

***Abstract*—The internet has become a permanent fixture in people's lives and has expanded significantly over the last few decades. As it has grown, data protection has become a critical concern to ensure that information is accessed only by intended recipients and remains secure from unauthorized modifications. Encryption plays a crucial role in data protection by transforming information into an unreadable format using a specific algorithm, requiring proper decryption to access and utilize the data. Graph theory, a field within discrete mathematics, is widely applied to strengthen encryption techniques for securing information. Harmonious labeling is utilized to assign values to edges in a graph, representing encrypted plaintext to ensure secure communication. Furthermore, incorporating the complement of a graph method along with the partial complement enhances encryption by adding an extra layer of complexity.**

***Index Terms*—Encryption and decryption, Harmonious Labeling, Secure communication, Complement of a graph, Partial complement.**

I. Introduction

GRAPH THEORY is a rapidly evolving field with diverse applications in cryptography. Various encryption methods have applied graph theory concepts to enhance security. One such approach, proposed by Etaiwi, incorporates principles such as the minimum spanning tree, cycles and complete graphs [1]. Several techniques of graph theory such as graph labeling, complement of a graph and generalized complements have applications in the field of cryptography [2], [3], [4].

In this study, the proposed algorithm employs graph labeling and partial complement operations for plaintext encoding. Alexander Rosa first introduced the concept of graph labeling in 1967 [5], and extensive research on the subject was later conducted by Graham and Sloane in 1980. The comprehensive Gallian survey on graph labeling provides a vast collection of studies in this area [6]. Additionally, Deepa and Maheswari explored the relationship between various ciphers and graph labeling techniques for data encoding [7]. Harmonious labeling, a significant type of graph labeling introduced by Graham and Sloane in 1980, is applicable to a specific subset of graphs [8]. Their study established that a cycle (C_n) is harmonious if and only if it is of odd length and the wheel (W_n) is harmonious for any value of n. Vaidya et al. conducted additional research on odd harmonious labeling of certain graphs in 2012 [9]. The concept of partial complementation of a graph was introduced by Fomin et al. [10].

In encryption, harmonious graph labeling is performed to set edge values identical to different plaintext values from the encryption table. The process is executed on a general graph to find edge values for the input plaintext. Then the complement and partial complement of the graph are calculated to produce a complicated cipher graph. Decryption is a process of reversal, beginning with the partial complement of the input cipher graph, followed by the regular complementation. The original edge values (plaintext values) are retrieved using the same graph labeling method. The plaintext is then reconstructed from the assigned keys and the encryption table. The algorithm adopts a symmetric key cryptographic technique since the same keys are employed for encryption and decryption.

Manuscript received April 9, 2025; revised May 28, 2025.

Naik Sinchana Ganapathi is a postgraduate student in the Department of Mathematics, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka, India-576104. (email: sinchananaik18@gmail.com).

Medini H R is a research scholar in the Department of Mathematics, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka, India-576104. (e-mail: medinihr@gmail.com).

*Sabitha D'Souza is an Associate Professor in the Department of Mathematics, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka, India-576104. (corresponding author; phone: 9449727376; e-mail: sabitha.dsouza@manipal.edu).

II. Preliminaries

Definition 1. *Graph labeling is a function that maps a set of vertices or edges of a graph to a non-negative integer that fulfills specific properties/rules.*

Definition 2. *A graph G with q edges is said to be harmonious if there is a one-to-one (injective) function f that maps the set of vertices to* $\{0,1,2,\ldots,q\text{-}1\}$ *so that each edge gets the label of the sum of the two vertices labels incident with that edge in modulo q, which are all different.*

Definition 3. *The Partial complement of a graph G is the concept derived from the complement of a graph, where only a subset of edges (say S) is toggled (added or removed) instead of considering the entire edge set.*

Theorem 1. *Splitting graph of a bistar S'*(B_n, n) *is an odd harmonious graph [10].*

III. Research Approach

This study introduces a novel cryptographic technique that integrates graph complements with harmonious labeling methods. The approach involves constructing a general graph, defining an encryption table, and subsequently encrypting the plaintext using these techniques.

A. *Encryption table*

English alphabets, spaces between words, special characters up to '#', and numerals up to '2' are assigned odd values ranging from 1 to 67. The remaining special characters and numerals are assigned even values ranging from 0 to 68. Using Table 1, the plaintext is converted into its numeric representation. These values serve as the first level of encryption, where unique values are used as edge weights to construct a cipher graph for the given plaintext.

TABLE I
ENCRYPTION TABLE

A	B	C	D	E	F	G	H	I	J	K	L
1	3	5	7	9	11	13	15	17	19	21	23
M	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**
25	27	29	31	33	35	37	39	41	43	45	47
Y	**Z**	**space**	~	\	**!**	**1**	**@**	**2**	#	**3**	**$**
49	51	53	55	57	59	61	63	65	67	0	2
4	**%**	**5**	ˆ	**6**	**&**	**7**	*	**8**	(	**9**	)
4	6	8	10	12	14	16	18	20	22	24	26
0	_	**-**	**+**	**=**	**{**	**}**	[	]	|	\	**:**
28	30	32	34	36	38	40	42	44	46	48	50
;	**'**	**"**	**,**	<	>	**.**	**?**	**/**			
52	54	56	58	60	62	64	66	68			

B. Procedure to construct a General Graph

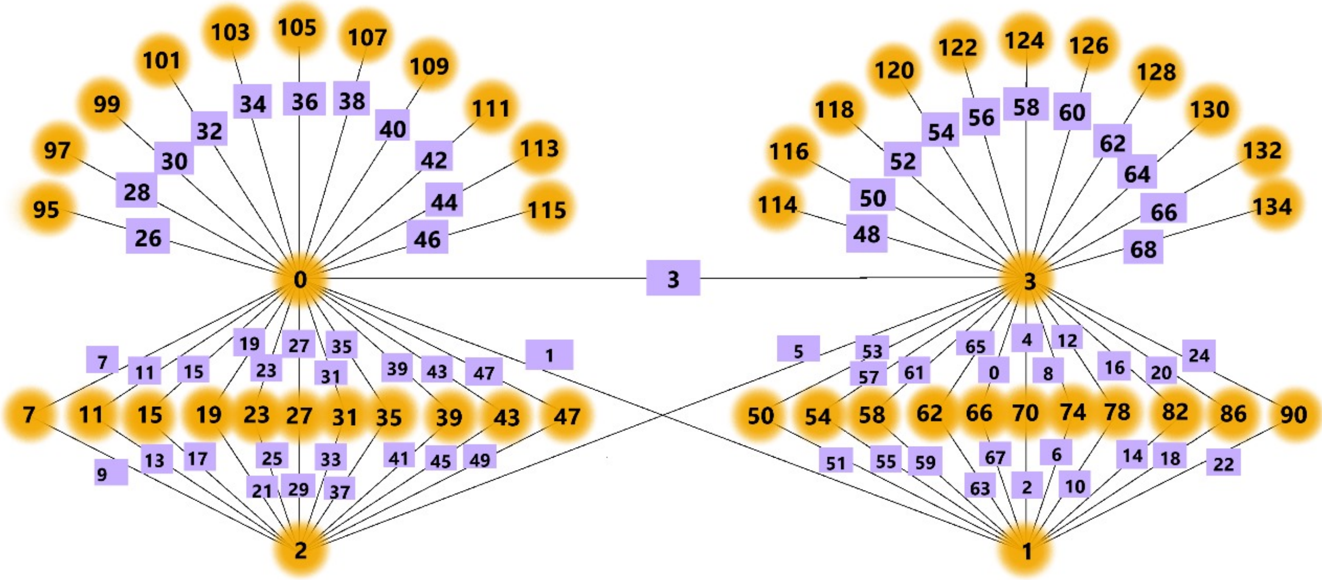


Fig. 1 General Graph

The vertices of the general graph are labeled according to Theorem 1 [7]. The edge values ranging from 0 to 68 are generated using harmonious labeling. The resulting graph is a general structure that can be applied to various types of plaintext. Table 1 illustrates the assignment of values to 69 characters. If additional characters are needed, the general graph can be expanded by incorporating extra vertices and edges.

C. Procedure for encryption and decryption

1) Encryption procedure:: Table 1 maps each character in the plaintext to a numerical value, which serves as the first level of encryption. From these encrypted values, distinct numbers are selected and used as edge values in the general graph. Only the corresponding edges are extracted from the general graph. The cipher graph is then created by taking the complement of the extracted graph, followed by a partial complementation using a specific set of edges.

2) Procedure for key generation::
Key 1: Consider only adjacent vertices in both the extracted and complement graphs. Start with the smallest non-isolated vertex (if the smallest vertex is an isolated vertex, then switch to complement graph and start) in the extracted graph, select the minimum among its neighbors, and form an edge. Then, switch to the complement graph and repeat the process. Continue selecting vertices from extracted and complement graphs alternately until all vertices are covered. The total edges will be $\lfloor n/2 \rfloor$; if odd, the last vertex is ignored. This subset of edges **S** forms Key 1.

Key 2: The set of distinct numbers of order n of the first encrypted values should be sorted in increasing order and their positions should be numbered with prime numbers. These are referred to as the second encrypted values. Key 2 is created by substituting the second encrypted value for the first.

Key 3: In plaintext, it is essential to differentiate between capital and small letters. Uppercase and lowercase letters are given special characters, ? and #, respectively. This combination of ? and # yields key 2.

The keys, the vertex-labeled cipher graph, and the set of edges are transferred to the recipient.

3) Decryption Procedure:: The partial complement of the vertex-labeled cipher graph is taken using Key 1 sent by the sender. Then the obtained graph is again complemented, and the edge values of the resulting graph are evaluated using harmonious labeling. The edge values are obtained, sorted in ascending order, and numbered with prime numbers. Key 2 is then used to decrypt Table 1's numbers in order of the plaintext, converting the numbers to characters. Alphabets in upper and lowercase are written using key 3. Hence, the plaintext will be produced.

IV. PROPOSED ALGORITHM

A. *Encryption Algorithm:*

Step 1: Use Table 1 to convert each character in plaintext into a numerical value.
Step 2: Choose the distinct values and extract those edges from the general graph.
Step 3: Take the complement of this extracted graph.
Step 4: Follow the key generation steps to obtain keys 1, 2, and 3.
Step 5: Take the partial complement of the complement graph to get the cipher graph.
Step 6: Send the keys and the vertex-labeled cipher graph to the recipient with harmonious labeling as a hint.

B. *Decryption Algorithm:*

Step 1: Use Key 1 to obtain the partial complement of the graph sent by the sender.
Step 2: Take the complement of the obtained graph.
Step 3: Apply the harmonious labeling method to compute the edge values.
Step 4: Arrange the edge values in increasing order and assign the prime numbers.
Step 5: Use Key 2 to retrieve the edge values in plaintext order.
Step 6: Use the encryption table to map the characters to the values obtained during Step 5.
Step 7: Use Key 3 to represent the characters in uppercase or lowercase and obtain the plaintext.

V. ILLUSTRAIONS

A. *Illustration 1*

Let the plaintext be **Hello World**

Encryption:

The first encrypted values for the given plaintext are obtained by referring to Table 1.

Plaintext	H	e	l	l	o	
First encrypted values	15	9	23	23	29	53

W	o	r	l	d
45	29	35	23	7

From these first encrypted values, distinct values are taken and sorted in ascending order.

7	9	15	23	29	35	45	53

These edges are extracted from the general graph and the extracted graph is obtained as shown in Fig. 1.

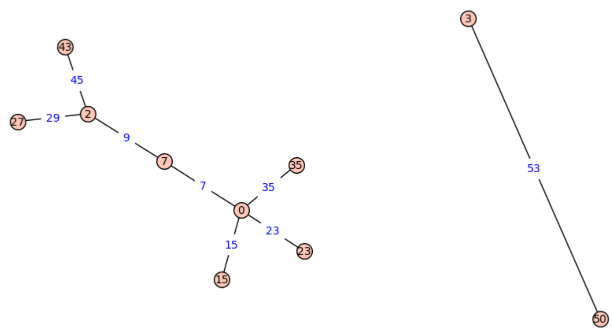


Fig. 2 Extracted Graph

Taking the complement of the extracted graph as shown in Fig. 3.

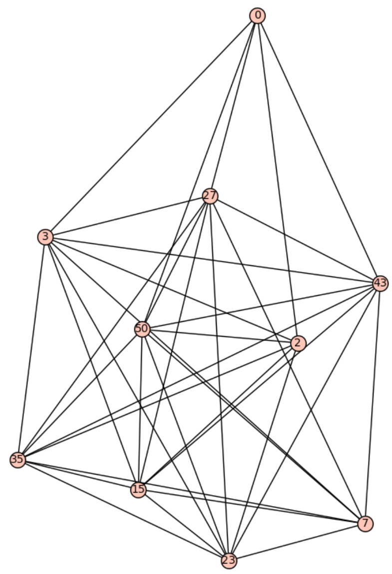


Fig. 3 Complement of the extracted graph

Key generation:

Consider the smallest non-isolated vertex in the extracted graph, which is '0' and the minimum among its neighbors is 7. Choose the edge **(0,7)**. Now, switch to the complement graph. Here, the next smallest vertex other than 0 and 7 is 2. And the minimum among its neighbors is 3. Choose the edge **(2,3)**. Again, switch to the extracted graph and repeat the process until all the vertices are covered.

Since the number of vertices is even (n = 10), the total number of edges in **S** is $\lfloor 10/2 \rfloor$ **= 5** and **S = {(0,7), (2,3), (15,23), (27,35), (43,50)}**. This set is considered as **Key 1** for the Decryption process.

The sorted distinct values are now numbered with prime numbers and are regarded as second encrypted values.

Distinct values	7	9	15	23	29	35	45	53
Second encrypted values	2	3	5	7	11	13	17	19

Key 2 is created by mapping the second encrypted values to the first encrypted values. By allocating '?' for uppercase characters and '#' for lowercase characters, **Key 3** is created.

Plaintext	H	e	l	l	o		W	o	r	l	d
First encrypted values	15	9	23	23	29	53	45	29	35	23	7
Key 2	5	3	7	7	11	19	17	11	13	7	2
Key 3	?	#	#	#	#	#	?	#	#	#	#

Take a partial complement of the resulting graph using S. The graph obtained is considered as the cipher graph.

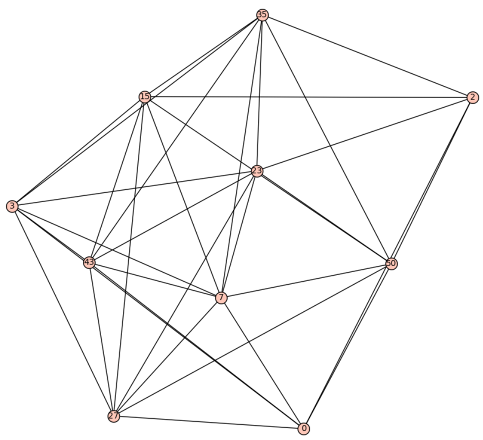


Fig. 4 Cipher Graph

Decryption:

The recipient must obtain the partial complement of the vertex-labeled cipher graph which was sent by the sender using Key 1. The graph obtained is then complemented and harmonious labeling is applied to get the edge values of the generated graph. The edge values are sorted in ascending order and their positions are numbered with prime numbers. Key 2 is used to decode Table 1's numbers in order of the plaintext and to convert numbers to characters. Key 3 is used to write the alphabet in both uppercase and lowercase. Then the plaintext will be generated as **Hello World**.

B. Illustration 2

Let the plaintext be **"Hell0, W0rld!"**

Encryption:

The first encrypted values for the given plaintext are obtained by referring to Table 1.

Plaintext	“	H	e	l	l	0	,		W	0	r	l	d	!	”
First encrypted values	56	15	9	23	23	28	58	53	45	28	35	23	7	59	56

From these first encrypted values, distinct values are taken and sorted in ascending order.

7	9	15	23	28	35	45	53	56	58	59

These edges are extracted from the general graph.

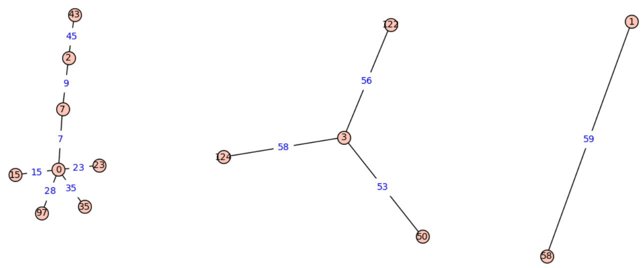


Fig. 5 Extracted graph

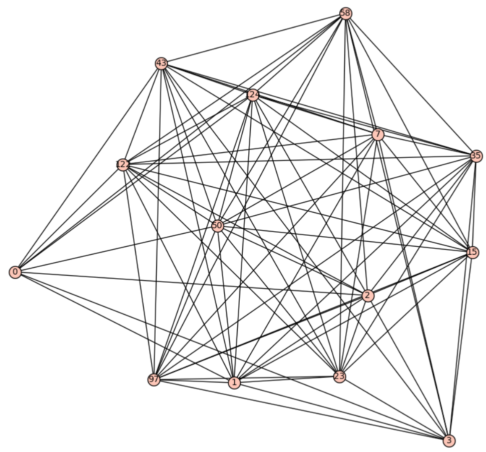


Fig. 6 Complement of the extracted graph

Key generation:

Consider the smallest non-isolated vertex in the extracted graph, which is '0' and the minimum among its neighbors is 7. Choose the edge **(0,7)**. Now, switch to the complement graph. Here, the next smallest vertex other than 0 and 7 is 1, and the minimum among its neighbors is 2. Choose the edge **(1,2)**. Again, switch to the extracted graph and repeat the process until all the vertices are covered.

Since the number of vertices is 14, the total number of edges in S is 7 and **S = {(0,7), (1,2), (3,50), (15,23), (35,43), (58,97),(122,124)}**. This set is considered as **Key 1** for the Decryption process. The sorted distinct values are now numbered with prime numbers and are regarded as second encrypted values.

Distinct values	7	9	15	23	28	35	45	53	56	58	59
Second encrypted values	2	3	5	7	11	13	17	19	23	29	31

Key 2 is created by mapping the second encrypted values to the first encrypted values. By allocating '?' for uppercase characters and '#' for lowercase characters, **Key 3** is created.

Plaintext	”	H	e	l	l	0	,		W	0	r	l	d	!	”
First encrypted values	56	15	9	23	23	28	58	53	45	28	35	23	7	59	56
Key 2	23	5	3	7	7	11	29	19	17	11	13	7	2	37	23
Key 3	#	?	#	#	#	#	#	#	?	#	#	#	#	#	#

The partial complement of the resulting graph is obtained using **S**. Hence, the graph obtained is the Cipher graph.

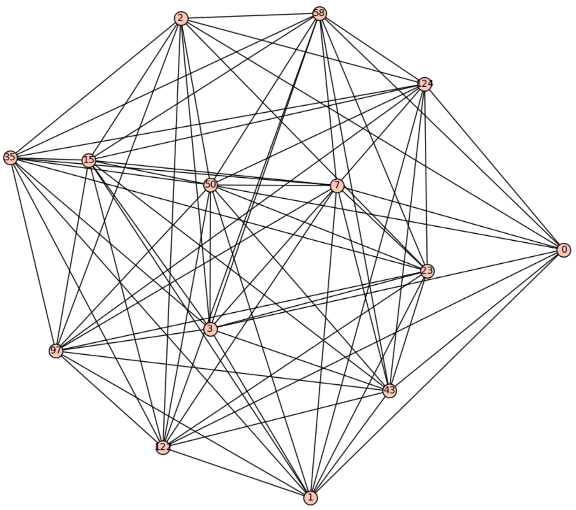


Fig. 7 **Cipher Graph**

Decryption:

By reversing the encryption process, the plaintext is retrieved as
”Hell0, W0rld!”.

C. Illustration 3

Let the plaintext be **Numbers never lie**

Encryption:

The first encrypted values for the given plaintext are obtained by referring to Table 1.

Plaintext	N	u	m	b	e	r	s
First encrypted values	27	41	25	3	9	35	37

	n	e	v	e	r		l	i	e
53	27	9	43	9	35		23	17	9

The sorted distinct values are given below.

3	9	17	23	25	27	35	37	41	43	53

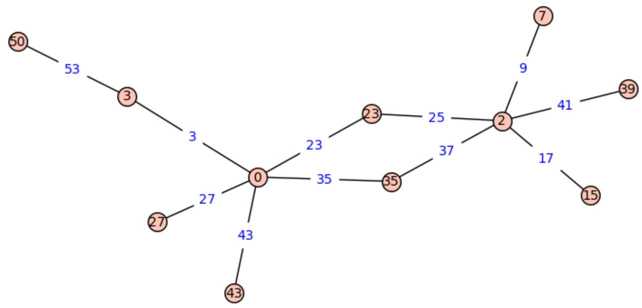


Fig. 8 Extracted graph

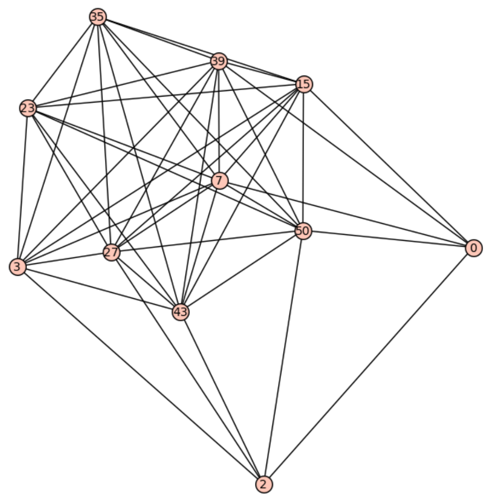


Fig. 9 Complement of the extracted graph

Key generation:

Key 1: {(0,3),(2,27),(7,15),(23,35),(39,43)}
Key 2: [13,29,11,2,3,17,23,37,13,3,31,3,17,37,7,5,3]
Key 3: [?,#,#,#,#,#,#,#,#,#,#,#,#,#,#,#,#]

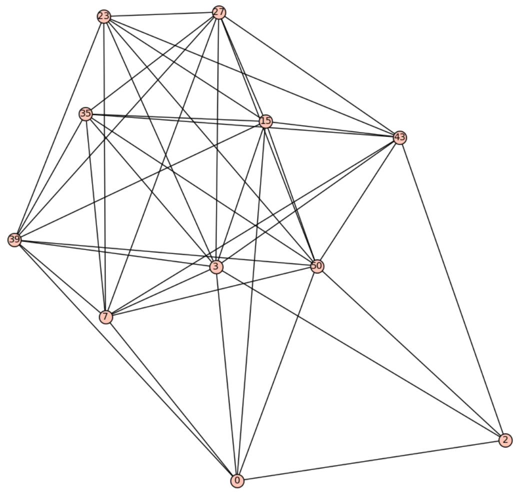


Fig 10. Cipher Graph

Decryption:

By reversing the encryption process, the plaintext is retrieved as
Numbers never lie.

D. Illustration 3

Let the plaintext be **Numbers never lie**

Encryption:

The first encrypted values for the given plaintext are obtained by referring to Table 1.

Plaintext	N	u	m	b	e	r	s
First encrypted values	27	41	25	3	9	35	37

	n	e	v	e	r		l	i	e
53	27	9	43	9	35		23	17	9

The sorted distinct values are given below.

3	9	17	23	25	27	35	37	41	43	53

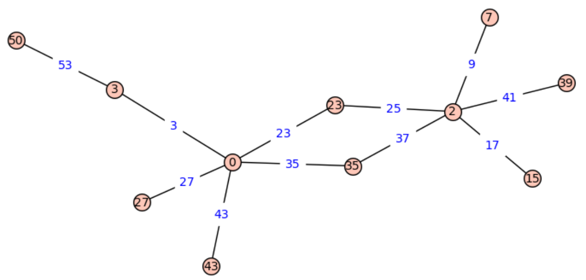


Fig. 8 Extracted graph

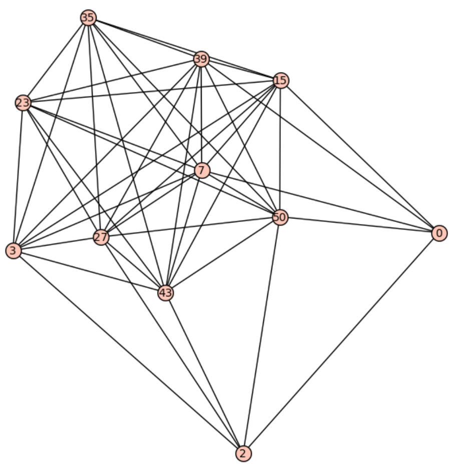


Fig. 9 Complement of the extracted graph

Key generation:

Key 1: {(0,3),(2,27),(7,15),(23,35),(39,43)}
Key 2: [13,29,11,2,3,17,23,37,13,3,31,3,17,37,7,5,3]
Key 3: [?,#,#,#,#,#,#,#,#,#,#,#,#,#,#,#,#,#]

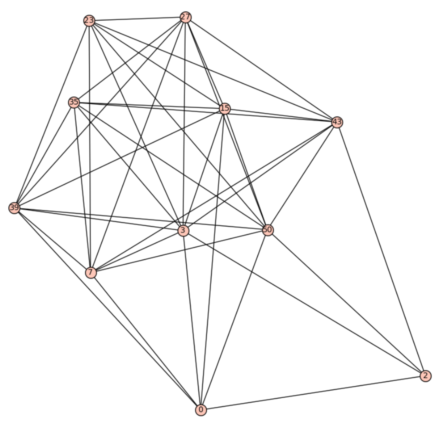


Fig 10. Cipher Graph

Decryption:

By reversing the encryption process, the plaintext is retrieved as
Numbers never lie.

E. Illustration 4

Let the plaintext be **N<mber$_ne^er_1!e**

Encryption:

The first encrypted values for the given plaintext are obtained by referring to Table 1.

Plaintext	N	<	m	b	e	r	$
First encrypted values	27	60	25	3	9	35	2

_	n	e	ˆ	e	r	_	1	!	e
30	27	9	43	9	35	30	61	59	9

The sorted distinct values are given below.

2	3	9	10	25	27	30	35	59	60	61

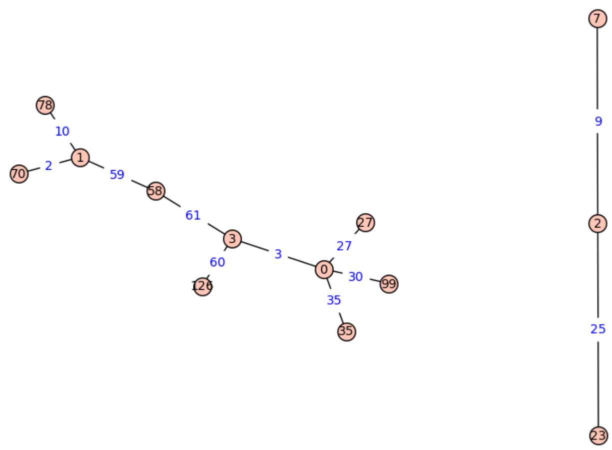


Fig. 11 Extracted graph

Key generation:

Key 1: {(0,3),(1,2),(7,23),(27,35),(58,70),(78,99)}
Key 2: [13,31,11,3,5,23,2,17,13,5,7,5,23,17,37,29,5]
Key 3: [?,#,#,#,#,#,#,#,#,#,#,#,#,#,#,#,#,#]

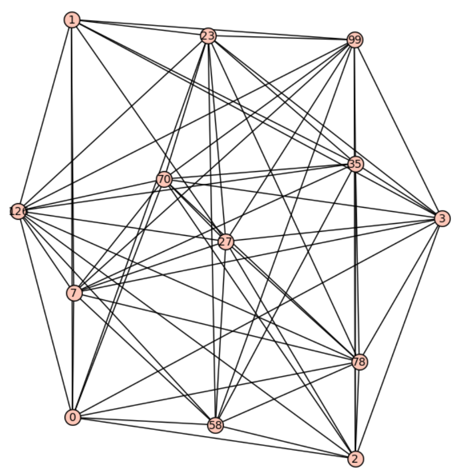


Fig. 12 Cipher Graph

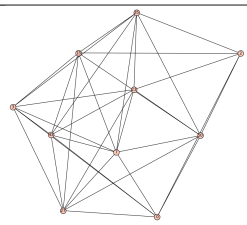
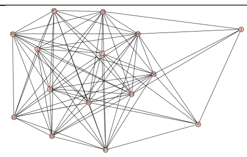
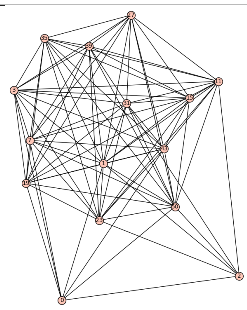
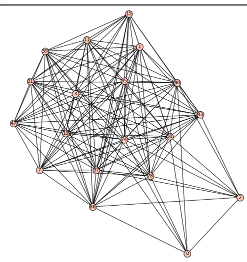
Decryption:

By reversing the encryption process, the plaintext is retrieved as
N<mber$_ne^er_1!e.

VI. Empirical Analysis

Table 2 displays the cipher graphs generated for different sizes of plaintext, emphasizing the effect of graph-based encryption. As the plaintext size increases, the resulting graph structures become more complex. The incorporation of special characters in plaintext also adds complexity to cipher graphs, influencing connectivity and encryption structure. In addition, the results emphasize how graph labeling and complement operations influence the encryption-decryption process.

Table 2. Some Cipher graphs without special characters

Sl. No.	Plaintext size without special characters	Cipher graphs
1	10	
2	20	
3	30	
4	40	

VII. Conclusion

This study introduces a novel encryption and decryption approach utilizing the concepts of graph theory, focusing on the use of graph complementation and graph labeling techniques. By leveraging these graph-based methods, the proposed approach enhances data security and encryption complexity. By representing plaintext characters as numbers and including them in graph structures, this algorithm provides security. The suggested approach is confidential through the employment of various keys and graph transformation, thus resistant to usual cryptographic attacks.

Additionally, the research builds upon existing studies in graph labeling, such as harmonious labeling and partial complementation, further expanding its potential applications in cryptography. This method not only illustrates the real-world applicability of graph theory to cryptography but also suggests new directions for future research, including the optimization of key generation algorithms and further application of this method to big data. Through ongoing developments, this encryption model can be modified for use in secure communications across a variety of fields, including cybersecurity and secure messaging networks.

References

[1] Wael Mahmoud Al Etaiwi, "Encryption Algorithm Using Graph Theory", *Journal of Scientific Research and Reports*, vol. 3, no.19, pp. 2519-2527, 2014.

[2] Medini H.R., Sabitha Dsouza, Devadas Nayak C, Pradeep G Bhat, "Encoding and Decoding of Messages Using Graph Labeling and Complement of a Graph". *Global and Stochastic Analysis*, vol. 11, pp. 1-13, 2024.

[3] Medini H.R., Sabitha Dsouza, Devadas Nayak C, Pradeep G Bhat, "Multifaceted Coding of Messages Using the Concepts of Graph Theory", *IAENG International Journal of Computer Science*, vol. 51, pp. 143-153, 2024.

[4] Medini H.R., Sabitha Dsouza, Devadas Nayak C, Pradeep G Bhat, "Exploring Secure Communication Through Artistic Graph Creation", *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 28, pp. 143–159, 2025.

[5] Alexandar Rosa, "On Certain Valuations of the Vertices of a Graph", *Theory of Graphs (Internat. Symposium, Rome)*, pp. 349-355,1966.

[6] Joseph A Gallian, "A Dynamic Survey of Graph Labeling", *Electron J Combin DS6*, vol. 19,2018

[7] Deepa Balaraman, Maheswari Veerasamy, Balaji V. "Creating Ciphertext and Decipher using Graph Labeling Techniques", *International Journal of Engineering and Advanced Technology*, vol. 9 pp. 206-211, 2019.

[8] Ronald L Graham, Neil James Alexander Sloane, "On Additive Bases and Harmonious Graphs", *SIAM Journal on Algebraic Discrete Methods*, vol.1(4), pp. 382-404, 1980.

[9] Samir Vaidya, Nirav Shah, "Odd Harmonious Labeling of Some Graphs", *International Journal of Mathematical Combinatorics*, vol. 3, pp. 105-112, 2012.

[10] Fedor V Fomin, Petr Golovach, Torstein Strømme, Thilikos, Dimitrios, "Partial complementation of graphs", *arXiv preprint arXiv:1804.10920*, 2018.