Global and Stochastic Analysis
Vol. 11 No. 1 (January, 2024)

Received: 15th August 2023 *Revised: 06th October 2023* *Accepted: 15th October 2023*

ENCODING AND DECODING OF MESSAGES USING GRAPH LABELING AND COMPLEMENT OF A GRAPH

MEDINI H R, SABITHA D'SOUZA*, DEVADAS NAYAK C, AND PRADEEP G BHAT

Abstract. Nowadays, the use of digital technology for communication is constantly growing. As a result, algorithms that provide more secure communication must be employed. Despite the fact that there are numerous encryption algorithms, app developers are constantly looking for a new or updated versions of the algorithms. In this proposed work, the vertex odd mean labeling method is used to produce edge values for a graph that serve as the encoded numbers for the characters of a plaintext. The complement of that graph is taken to generate a cipher graph. Communication will be safer since each text will have a distinct cipher graph. The text is encrypted and decrypted using the symmetric key technique. This algorithmic approach is useful for all texts with or without special characters.

1. Introduction

Graph theory is an emerging and broad field of research. Numerous graph theory topics have been used in the domain of cryptography. The encryption algorithm developed by W. Etaiwi made use of graph theory ideas such as the minimum spanning tree, the cycle, and complete graphs [1]. Late in the 1960s, graph labeling was first developed. Rosa in 1967 and Graham and Sloane in 1980 conducted multiple studies on graph labeling. Rosa identified 4 types of labelings namely, α, β, σ and ρ-labelings [5]. Later, Solomon Golomb changed the name of β-labeling to graceful. Mean labeling of graphs was first introduced by Somasundaram and Ponraj [6]. Later, it was expanded to super mean, k-super mean, even vertex mean, odd vertex mean, even vertex odd mean, and odd vertex even mean, and several others. The Gallian survey on graph labeling contains the majority of the works that deal with graph labeling [3]. Deepa and Maheswari used different ciphers and graph labeling methods for coding [2]. N.Revathi worked on the vertex odd mean and vertex even mean labelings of some of the graphs in 2015 [4].

The proposed work involves the use of graph labeling and complement of the graph for coding a plaintext. The suggested approach uses odd vertex mean labeling to encode plaintexts since it creates cipher graphs most effectively. In encryption, the method of graph labeling is used to get edge values which are the distinct plaintext values obtained from the encoding chart. The vertex odd

2000 *Mathematics Subject Classification.* 05C85; 05C78.
Key words and phrases. Encryption, Decryption, Coding of messages, Graph labeling, Complement of a graph, Secure communication.

mean labeling is applied for the general graph to obtain values of the edges for the corresponding plaintext. The complement of the graph is taken to generate a complex cipher graph. In the process of decryption, the complement of the cipher graph is taken. By using the same graph labeling method, it is possible to get the edge values which are the plaintext values. The plaintext can be decoded using the keys and the encoding table. The keys used are the same for both processes. Therefore, it is a symmetric key algorithm.

2. Proposed work

A new type of crypto-technique is described which makes use of both complement of graphs and vertex odd mean labeling methods. The procedure to construct a general graph and the encoding table are explained, followed by the encryption of the plaintext.

2.1. Encoding table. The following encoding table is used to convert each character in the plaintext into a number.

Table 1: Encoding chart

A	B	C	D	E	F	G	H	I	J	K	L	M	N
2	3	4	5	6	7	8	9	10	11	12	13	14	15
O	P	Q	R	S	T	U	V	W	X	Y	Z	space	
16	17	18	19	20	21	22	23	24	25	26	27	28	
˜	`	!	1	@	2	#	3	$	4	%	5	ˆ	6
29	30	31	32	33	34	35	36	37	38	39	40	41	42
&	7	*	8	(	9	)	0	_	-	+	=	{	[
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	70

English alphabets are given values ranging from 2 to 27, and the space between words is given the number 28. The numbers assigned to the special characters and numerals range from 29 to 70.

Using Table 1, a numeric representation of the plaintext can be obtained. These values are regarded as the first encrypted values, where distinct values are used as edge values to create a cipher graph for the specific plaintext.

2.2. Procedure to construct a general graph. The general graph consists of odd vertices labeled 1,3,5,...,95. The edge values from 2 to 70 are generated by using odd vertex mean labeling. The resulting graph is a general graph that can be used with different types of plaintexts. Table 1 shows that values are assigned to 69 characters. The general graph can be expanded by including extra vertices and edges if a few extra characters are required.

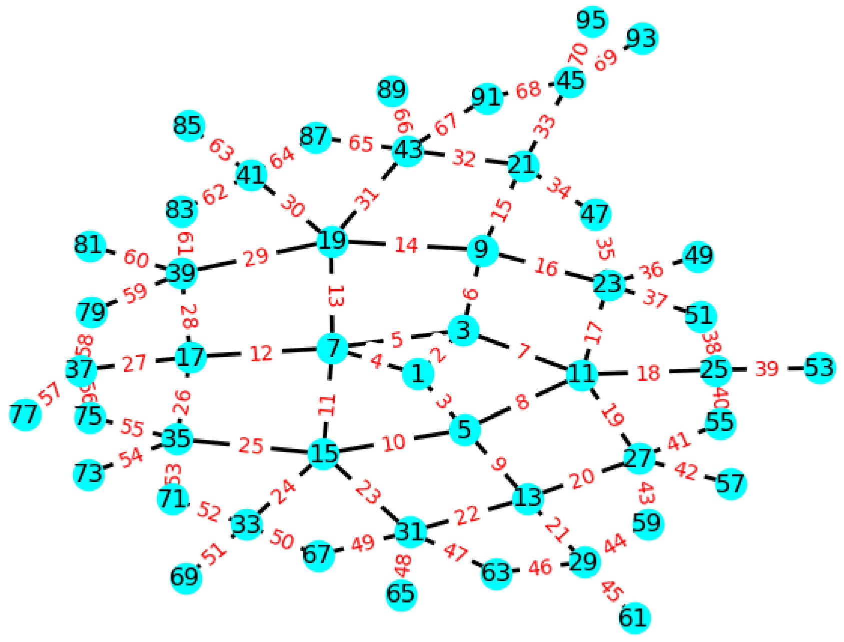


FIGURE 1. General graph

2.3. Procedure for encryption and decryption.

2.3.1. *Procedure for encryption:*

Using Table 1, each character in the plaintext is converted into a numerical value. These numerical values are employed as the first encrypted values. The distinct numbers are chosen from the first encrypted values and are regarded as edge values in the general graph. Only those edges are extracted from the general graph. By taking the complement of the extracted graph, the cipher graph is generated.

2.3.2. *Procedure for key generation:*

Key 1: The set of distinct numbers of order n of the first encrypted values should be organized in increasing order and their positions should be numbered from 1 to n. The positions are referred to as the second encrypted values. Key 1 is created by substituting the second encrypted values for the first encrypted values.

Key 2: In plaintext, it is essential to differentiate between capital and small letters. Uppercase and lowercase letters are given the numbers 1 and 0, respectively. This combination of 1 and 0 yields key 2.

The keys and the vertex-labeled cipher graph are transferred to the recipient.

2.3.3. *Procedure for decryption:*

The complement of the vertex labeled cipher graph sent by the sender must be obtained by the recipient. The edge values of the resulting graph are evaluated using the vertex odd mean labeling. The edge values are obtained, sorted in ascending order and numbered from 1 to n, where n is the total number of edges. Key 1 is used to decrypt Table 1's numbers in order of the plaintext, converting the numbers to characters as a result. Alphabets in upper and lowercase are written using Key 2. Then the plaintext will be produced.

2.4. Proposed algorithm.

2.4.1. *Encryption algorithm:*

Step 1: Use Table 1 to convert each character in plaintext into a number.

Step 2: Choose the distinct numbers, write them in increasing order, and then follow the key-generating instructions.

Step 3: As the distinct numbers indicate edge values in the general graph, extract the graph that only contains these edges.

Step 4: Take the complement of the extracted graph which is regarded as the cipher graph.

Step 5: Send the keys, along with the vertex-labeled cipher graph to the recipient.

2.4.2. *Decryption algorithm:*

Step 1: Obtain the complement of the graph sent by the sender.

Step 2: Apply the vertex odd mean labeling to compute the edge values.

Step 3: Arrange the edge values in increasing order and number their positions.

Step 4: Use key 1 to retrieve the edge values in plaintext order.

Step 5: Use Table 1 to map the characters to the values obtained during step 4.

Step 6: Use key 2 to represent the characters in uppercase or lowercase.

Step 7: Generate the plaintext.

2.5. Flow chart:

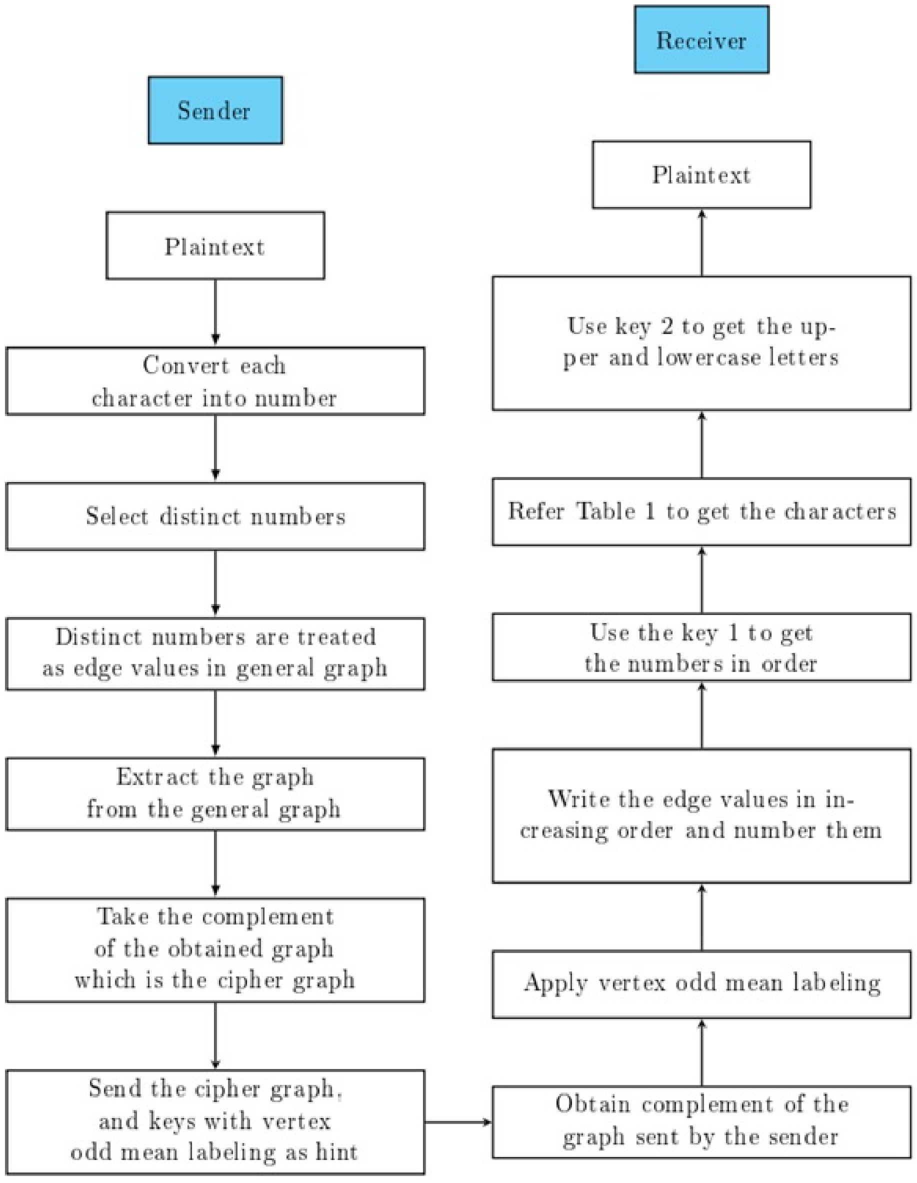

Sender
Plaintext
Convert each character into number
Select distinct numbers
Distinct numbers are treated as edge values in general graph
Extract the graph from the general graph
Take the complement of the obtained graph which is the cipher graph
Send the cipher graph, and keys with vertex odd mean labeling as hint
Receiver
Plaintext
Use key 2 to get the upper and lowercase letters
Refer Table 1 to get the characters
Use the key 1 to get the numbers in order
Write the edge values in increasing order and number them
Apply vertex odd mean labeling
Obtain complement of the graph sent by the sender

FIGURE 2. Conceptual model

2.6. Illustration. The procedure of converting a plaintext that solely contains alphabets is demonstrated. The same steps are followed when dealing with plaintext that comprises special characters.

Example 1
Encryption:

Step 1: Let the plaintext be **Mathematics**.

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

Plaintext	M	a	t	h	e	m	a	t	i	c	s
First encrypted values	14	2	21	9	6	14	2	21	10	4	20

Step 2: The distinct values from the first encrypted values are chosen and sorted in ascending order.

2	4	6	9	10	14	20	21

Key generation:
Since there are 8 distinct values, the numbers(their positions) 1 to 8 are assigned for the arranged distinct values and are regarded as second encrypted values.

Distinct values	2	4	6	9	10	14	20	21
Second encrypted values	1	2	3	4	5	6	7	8

The process of two keys generation is explained in the below table. The plaintext and its initial encrypted values are noted. By mapping the second encrypted values for the first encrypted values of the plaintext, the first key, denoted as key 1, is created. By allocating 1 for uppercase characters and 0 for lowercase letters, the second key, designated as key 2, is created.

Plaintext	M	a	t	h	e	m	a	t	i	c	s
First encrypted values	14	2	21	9	6	14	2	21	10	4	20
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: The following disconnected graph for the selected plaintext is obtained by referring to the general graph.

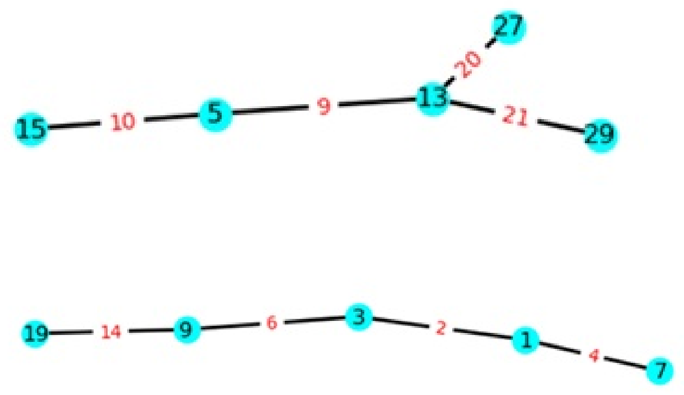


FIGURE 3. Extracted graph

Step 4: By taking the complement of the extracted graph, the following cipher graph is obtained.

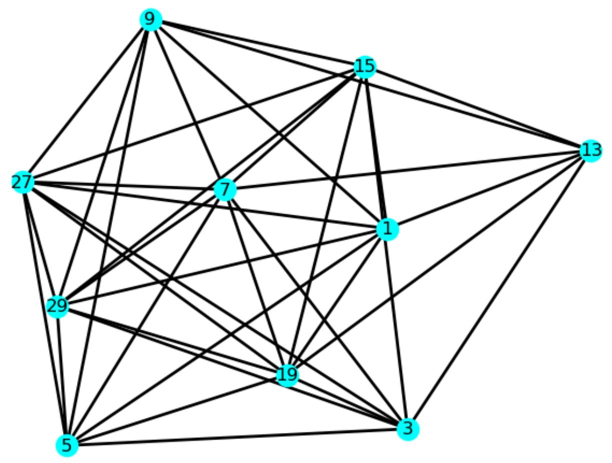


FIGURE 4. Cipher graph

Step 5:
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]
The keys along with the vertex-labeled cipher graph will be sent to the receiver.

Decryption:

The recipient must obtain the complement of the vertex labeled cipher graph which was sent by the sender. The vertex odd mean 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 from 1 to n, where n is the total number of edges. Key 1 is used to decode Table 1's numbers in order of the plaintext and to convert the numbers to characters. Key 2 is used to write the alphabet in both uppercase and lowercase. Then the plaintext will be generated as **Mathematics**.

Example 2

Encryption:

Step 1: Let the plaintext be **M@them@t!c$**
The first encrypted values for the given plaintext are obtained by referring Table 1.

Plaintext	M	@	t	h	e	m	@	t	!	c	$
First encrypted values	14	33	21	9	6	14	33	21	31	4	37

Step 2: The distinct values from the first encrypted values are chosen and sorted in ascending order.

4	6	9	14	21	31	33	37

Key generation:

Since there are 8 distinct values, the numbers(their positions) 1 to 8 are assigned for the arranged distinct values and are regarded as second encrypted values.

Distinct values	4	6	9	14	21	31	33	37
Second encrypted values	1	2	3	4	5	6	7	8

The process of generation of two keys is explained in the below table. The plaintext and its initial encrypted values are noted. By mapping the second encrypted values for the first encrypted values of the plaintext, the first key, denoted as key 1, is created. By allocating 1 for uppercase characters and 0 for lowercase letters, the second key, designated as key 2, is created.

Plaintext	M	@	t	h	e	m	@	t	!	c	$
First encrypted values	14	33	21	9	6	14	33	21	31	4	37
Key 1	4	7	5	3	2	4	7	5	6	1	8
Key 2	1	0	0	0	0	0	0	0	0	0	0

Step 3 & **Step 4:** A disconnected graph is obtained by referring to the general graph and by taking the complement of that graph, the following cipher graph is obtained.

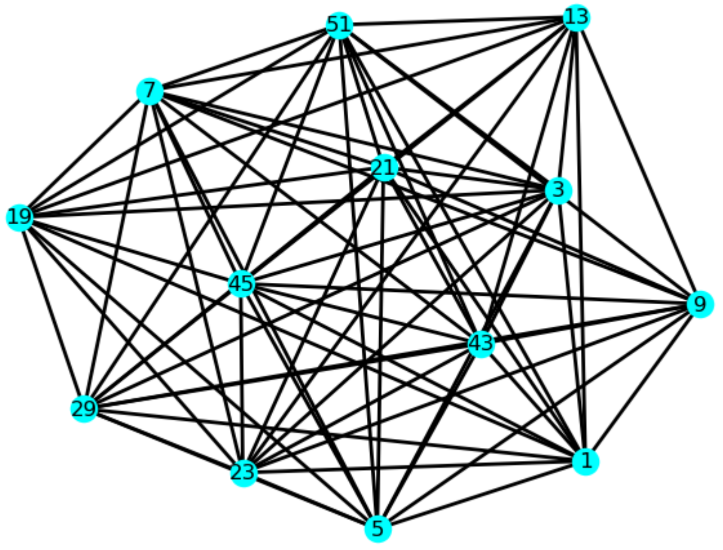


FIGURE 5. Cipher graph

Step 5:
Key 1: [4, 7, 5, 3, 2, 4, 7, 5, 6, 1, 8]
Key 2: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

The keys along with the cipher graph are sent to the receiver.

Decryption:

By applying the reverse process of encryption, the plaintext will be generated as **M@them@t!c$**.

3. Experimental examination

The Python Jupyter Notebook is used to implement the suggested algorithm. The program's execution times for various plaintexts when run on an i3 processor with 4GB of RAM are noted. It is observed that the running time of the program increases as the plaintext size increases in case of both plaintexts with or without special characters as shown below. However, more intricate cipher graphs will be created with an increase in the size of the plaintext.

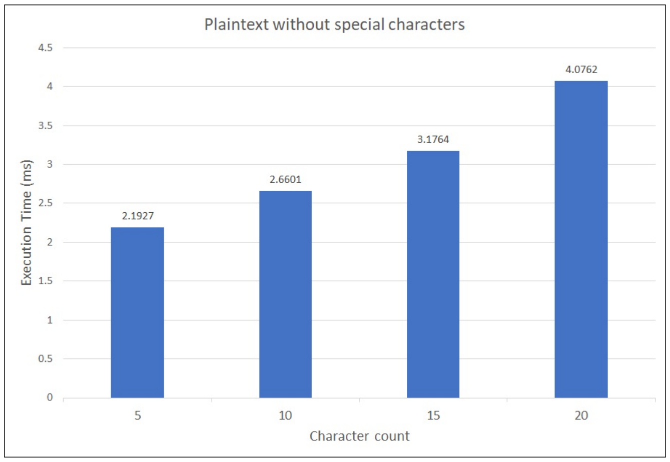


FIGURE 6. Execution time of plaintexts without special characters

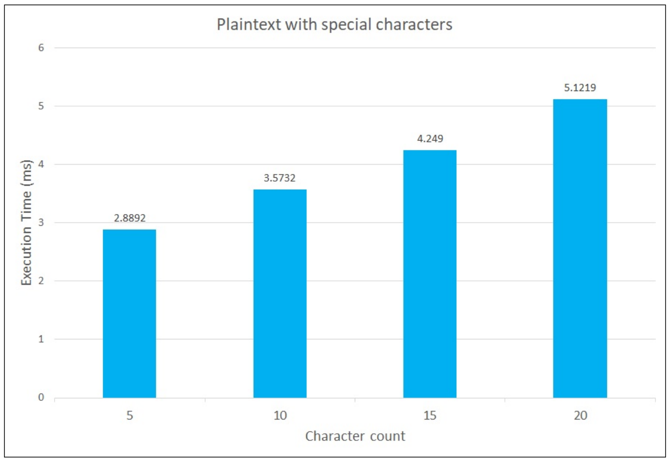
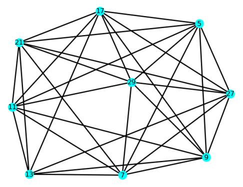
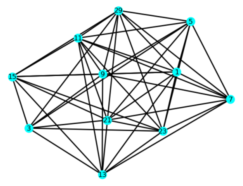
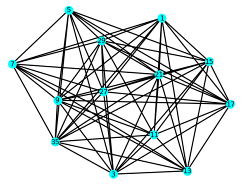
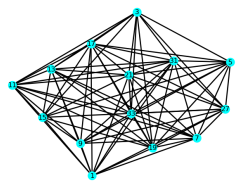
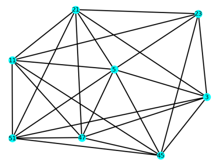
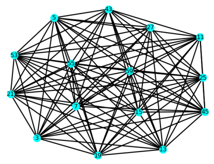
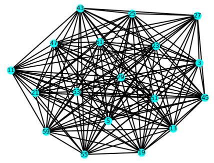
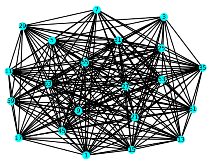


FIGURE 7. Execution time of plaintexts with special characters

Table 2: Cipher graphs and execution time of the random plaintexts.

Sl. No.	**Plaintext size without special characters**	**Cipher graphs**	**Running time (ms)**
1	5		2.1927
2	10		2.6601
3	15		3.1764
4	20		4.0762

Sl. No.	Plaintext size with special characters	Cipher graphs	Running time (ms)
1	5		2.8892
2	10		3.5732
3	15		4.2490
4	20		5.1219

3.1. Time complexity.
The time complexity of the algorithm involves multiple terms such as O(n), O(NlogN) and O(n*m). However, the terms O(n) and O(NlogN) become less significant as the input size increases and the O(n*m) term dominates the algorithm's performance. Therefore, the overall time complexity of the algorithm is O(n*m), where n is the length of the input and m is the unique elements in an array.

3.2. Space complexity.
The space complexity of the code is O(n+m), where n is the length of the input word and m is the number of unique elements in the array.

4. Conclusion

To ensure secure communication, end-to-end encryption is now widely employed in mobile applications. There are many advantages of the proposed algorithm, including the fact that it is nearly impossible to decode and takes less time to run, which makes it more secure and private and can be used as a method of communication when messages are being sent. It can also be used in digital currency exchanges to secure user data, including account numbers, amount transactions, and digital signatures.

The suggested work applies graph labeling and complement of a graph to create a new type of data-sharing mechanism. To create cipher graphs for the provided plaintext and key generation, many techniques are covered. Utilizing the complement technique, the proposed algorithm's complexity is increased, enabling greater security. The suggested approach is valid for all plaintexts, even those containing special characters. In this paper, the usage of vertex odd mean labeling and graph complement is explored. Additionally, it is also possible to use other approaches of graph labeling that are better suited for coding.

References

1. Al Etaiwi, Wael Mahmoud: Encryption algorithm using graph theory, *Journal of Scientific Research and Reports* **3(19)** (2014) 2519–2527.
2. Deepa, B. and Maheswari, V.: Ciphering and Deciphering Messages by Graph Labeling Techniques Through Multilevel Cryptosystem, *International Journal of Recent Technology and Engineering (IJRTE)* **8(4S5)** (2019) 33–36.
3. Gallian, Joseph A.: A dynamic survey of graph labeling, *Electronic Journal of combinatorics 1(DynamicSurveys)* (2018) DS6–Dec.
4. Revathi, N.: Vertex odd mean and even mean labeling of some graphs, *IOSR journal of mathematics* **11(2)** (2015) 70–74.
5. Rosa, Alexander: On certain valuations of the vertices of a graph, in: *Theory of Graphs* (1966) 349–355, Internat. Symposium, Rome.
6. Somasundaram, S. and Ponraj, R.: Mean labelings of graphs, *National academy Science letters* (2003) 210–213.

Medini H R: Department of Mathematics, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka-576104, India.
Email address: medinihr@gmail.com

Sabitha D'Souza (*Corresponding author): Department of Mathematics, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka-576104, India.
Email address: sabitha.dsouza@manipal.edu

Devadas Nayak C: Department of Mathematics, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka-576104, India.
Email address: devadas.nc@manipal.edu

Pradeep G Bhat: Department of Mathematics, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka-576104, India.
Email address: pg.bhat@manipal.edu