Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 28 (2025), No. 7, pp. 2811–2825
DOI : 10.47974/JDMSC-2201

Encryption and decryption scheme through signed Cayley graph

Obaidullah Wardak [§]
Sandeep Kumar [†]
Deepa Sinha *
Department of Mathematics
South Asian University
New Delhi 110068
Delhi
India

Abstract

The encryption and decryption of sensitive information have become increasingly important in today's digital age. In this paper, we explore the innovative application of signed Cayley graphs for achieving enhanced security in data transmission. In the realm of communication, where information exchange is pervasive, the graph itself serves as the message, with its vertices, edges, and edge signatures containing the concealed information. This paper presents the development of an encryption and decryption scheme that utilizes a signed Cayley graph, along with a private (symmetric) key, to securely encrypt and decrypt the edges of the graph.

Subject Classification (2020): *05C22, 05C75, 05C85.*

Keywords: *Signed cayley graph, Encryption, Decryption, Algorithm.*

1. Introduction

Communications, on several occasions, are to be transmitted secretly. This become all the more important in today's advanced time of online network. Time and again, new methods are developed in this regard with

[§] *ORCID-ID:* **0000-0002-2410-1960**
[†] *ORCID-ID:* **0009-0005-7107-3914**
* *ORCID-ID:* **0000-0002-5695-5282**

[§] *E-mail:* **obaid.wardak22@live.com**
[†] *E-mail:* **ssulakh.94@gmail.com**
* *E-mail:* **deepasinha2001@gmail.com (Corresponding Author)**

modern technology. The terminologies and studies on cryptography can be seen in [15, 18].

There has been growing interest in the last few years to link graph theory, algebraic structures [19] and cryptography [11, 22]. Graphs have been used as a tool for producing ciphertext from the plaintext, keys in the encryption and decryption, for generating digital structures [3, 6, 14, 20]. Here the structure of the graphs and their transformation is used in the cryptography process. On the other hand homomorphic encryption [2, 5] have been used to encrypt graph and with the advancement of cloud computing, can be found a dealt with without of decryption [7, 21], which also has the limitations and challenges discussed in [4, 9].

In this article, we are mainly concerned with the encryption of graphs, which are confidential and are not secure enough to be transmitted from the sender to the receiver through the network. The method used in the paper is different from the methods used earlier as here we developed an encryption function where cryptographic keys are difficult to attack and take exponential time of computation in the decryption if the knowledge of the keys is kept secret.

The Signed Cayley Graph method stands out as a more suitable encryption approach compared to other graph-based methods due to its enhanced security features, efficient key management, flexibility, scalability, resistance to attacks, and robust mathematical foundation. These attributes collectively position it as a promising solution for addressing the evolving challenges of data security in modern communication networks.

Our proposed method and Random Walk Graph Encryption scheme offer graph-based approaches to encryption and decryption, they differ in terms of security mechanisms, key management, flexibility, and resistance to attacks. The Signed Cayley Graph method provides enhanced security through edge signatures, streamlined key management, flexibility in graph structures, resilience against attacks, and a solid mathematical foundation, making it a more suitable choice for secure data encryption in communication networks. Further, we consider a signed graph for encryption and decryption with the assumption that information that has to be concealed lies on the edges of the signed graph [10, 17].

2. Preliminaries

2.1 *Signed Graph*

The following section defines key concepts that are relevant to the encryption and decryption scheme proposed in the paper using the signed Cayley graph. These concepts will provide a framework for understanding the algorithm's effectiveness and limitations in securing graph data.

An ordered pair $S = (S^u, \sigma)$, with S^u as a graph $G = (V, E)$ and σ as a function that assigns each edge in the edge set E either +1 or –1, $\sigma : E \rightarrow \{+1, -1\}$, is called *Signed graph* [1]. Here S^u is known as *underlying graph* of S and σ is known as *signature* of S. A network can be represented by signed graphs using matrices, so any network (or signed graph) can be converted into another network by using encryption and decryption techniques and vice-versa.

The *negative (positive)* degree of a vertex u, denoted by $d^-(u)$, $(d^+(u))$, is the number of negative (positive) edges incident to a vertex u in signed graph S. The sum of both the negative and positive degrees of a vertex u is called *total* degree of that vertex u, and denoted as $d(v) = d^-(v) + d^+(v)$.

2.2 *Signed Cayley Graph*

Let Z_n be a ring of integers modulo n, for $n \in I^+$ and U_n is a unit set. Consider X_n is the unitary Cayley graph [16] then an ordered pair $S_n = (X_n, \sigma)$ is called *signed Cayley graph* if Z_n is vertex set and two vertices u_1 and u_2 are adjacent adjacent in X_n, if and only if, $x_1 - x_2$ is in U_n. For any edge u_1u_2 in S_n,

$$\sigma(u_1u_2) = \begin{cases} + & \text{if } u_1, u_2 \in U_n, \\ - & \text{otherwise} \end{cases}$$

2.3 *One-way Function*

A one-way function f is a mathematical function if and only if satisfies:

a. For an argument x in the given domain, the function $y = f(x)$ is easy to compute polynomially depending on the size of x. and
b. to obtain x from the inverse, $f^{-1}(y)$, the computation requires exponential time in the size of y [12, 13].

2.4 *Trapdoor One-way Function*

The trapdoor one-way function is a one-way function f with some additional conditions so that the inverse, $f^{-1}(y)$ becomes computationally polynomial type in the size of y.

2.5 *One-time key encryption*

As described in [8], a one-time key encryption in any cryptographic scheme is occurs when an entirely new key is used for every plaintext and such newly used key is not reused in encryption.

The cryptographic scheme used in this paper using a signed Cayley graph is also a trapdoor one-way function, not utilizing the key more than once in the encryption.

The scheme comprises of four graphs. The first graph suppose of order n is plaintext which has to be encrypted; the second is a graph on the same number of vertices which is used for implementing encryption, which in the proposed scheme is signed Cayley graph S_n; the third graph is obtained by superimposing the vertices of first and second graph; and the fourth graph is the graph holding the message as the ciphertext, the encrypted message which is to be transmitted.

In this way, it will be difficult to attack or will take the computational tasks to dig the plaintext.

3. Proposed Mechanism for Encryption and Decryption

The method used here is a novel approach that utilizes the structure of the message itself. In this approach, the message is represented as a signed graph $S = (V_1, E_1, \sigma_1)$, and information is stored in the edges of the graph. By doing so, the information is not only encrypted but also disguised within the structure of the message. This method offers a unique advantage over traditional encryption methods, where the encrypted message is separate from the structure of the original message. This proposed mechanism could potentially provide enhanced security and privacy for sensitive information and has promising implications for the field of cryptography.

3.1 *The signed graph S*

Let the plaintext, or original message be represented as a signed graph $S = (V_1, E_1, \sigma_1)$ where,

$$V_1 = \{0, 1, 2, \cdots, n_1 - 1\}$$

and set of m_1 edges,

$$E_1 = \{e_1, e_2, \cdots, e_{m_1}\}$$

Now, define signed Cayley graph $S_n = (V_1, E, \sigma)$ as given in [?].

3.2 *The signed graph* S'

In the context of encrypting a graph S and its associated message, the first stage involves defining a new signed graph S' based on the pre-existing signed graphs S and S_n. Specifically, we construct S' as follows: $S' = (V', E', \sigma')$, where $V' = V_1$ and E' is set of edges, defined as

$$E' = E + E_1$$

Where E is set of edges of the signed Cayley graph S_n and E_1 is set of edges of the signed graph S. The resulting graph S' is a multi-graph. This initial stage sets the foundation for the subsequent encryption steps.

3.3 *Generating Key for Encryption and Decryption*

In this scheme we are using private key or symmetric key for encryption and decryption, which uses the same key for both encryption and decryption. This key is kept secret by the owner and must be shared with the recipient of the encrypted message.

Here the key is a trap-door one way function $\phi : E' \to \mathbb{R}$, where

$$\phi(v_i v_j) = \begin{cases} \phi_1(v_i v_j) & \text{if } i \in U_n \text{ or } j \in U_n, \\ \phi_2(v_i v_j) & \text{otherwise} \end{cases}$$

It is important to note that in $\phi(v_i v_j)$, $v_i v_j$ refers to the weight of the edge $v_i v_j$.

3.4 *Encryption of the Graph* S'

As previously mentioned, the information in the signed graph S is stored in its edges. Therefore, in this study, an encryption scheme is being applied to the edges of signed graph S'. Specifically, the encryption is being performed on the weights of each edge using the key ϕ, resulting in a new weight for each edge. Subsequently, a new graph S'' is constructed using the same set of vertices, but with newly encrypted edges. Here, $S'' = (V'', E'')$ is a multi graph with the same vertices set as of S', i.e.,

$V'' = V_1$ and E'' is the new edges set, which are the encrypted edges of S', i.e., $\phi(v_i v_j)$ for every edge $v_i v_j \in E'$.

3.5 *Decryption of the Graph* S''

i. The decryption of the graph S'' involves the application of a decryption process on its edges. The key ϕ is used for encryption and is shared with the recipient. It is also a trap-door one-way function. The recipient can determine the inverse of ϕ denoted as $\phi^{-1} : E'' \rightarrow E'$. This inverse function maps weight of every edge $v_i v_j \in E''$ to its weight of corresponding edge $v_i v_j \in E'$. Hence the graph S'.
ii. Following the construction of graph S', we proceed to remove all edges of the signed Cayley graph S_n from S'. This process yields the receiving graph denoted as S, which corresponds to the original message or plaintext.

3.6 *Algorithm for the above Proposed Mechanism*

Input: Signed graph S and signed Cayley graph S_n

1. Define V' as $\{0,1,2,...,n_1 - 1\}$
2. Define E' as $E + E_1$, i.e., add all the edges of graph S to the graph S_n
3. Set $S' = (V', E')$

Generating the key:

1. Choose two trap-door one-way functions ϕ_1 and ϕ_2
2. Create a key function ϕ that combines ϕ_1 and ϕ_2 based on the set U_n, where U_n is unit set,

$$\phi = (\phi_1, \phi_2, S_n)$$

3. Output the key function ϕ as the private key.

The key function ϕ is created by combining ϕ_1 and ϕ_2 based on the vertex set U_n, as described in the statement.

Encryption of S'

Input: Graph S', key ϕ

1. For each edge $v_i v_j \in E'$:
   1.1 If $i \in U_n$ or $j \in U_n$, set weight $w = \phi_1(v_i v_j)$.

1.2 Otherwise, set weight $w = \phi_2(v_i v_j)$

1.3 Encrypt w using the key ϕ to get the new weight w'.

1.4 Set the weight of the edge $v_i v_j \in E''$ as w'.

2. Set $S'' = (V', E'')$ and hence the encrypted graph.

Decryption of S''

Input: Encrypted graph S'', key ϕ

- Compute the inverse function of ϕ.
- For each edge $e = (v_i, v_j) \in S''$, decrypt the edge weight using ϕ^{-1} for edge $e \in S''$.
- Construct the graph S' with respect to the ϕ^{-1} values.
- Remove edges of the signed Cayley graph S_n from S'.
- The resulting graph is the decrypted message or plaintext S.

4. Illustration

Consider the following graph in **Figure** 1.

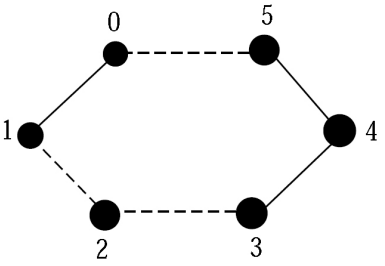

Figure 1

Given graph *S*

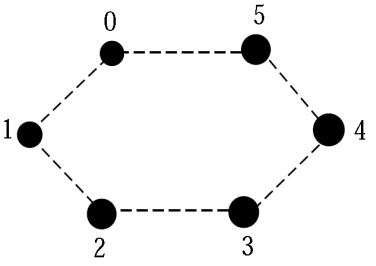

Figure 2

Signed Cayley graph S_n

Based on the given graph S, Signed Cayley graph S_6 is shown in **Figure** 2.

In next step, we are creating a multi-graph denoted as S'. This graph will have the same number of vertices as the original graph S and will include all the edges from both S and S_n, as shown in **Figure** 3.

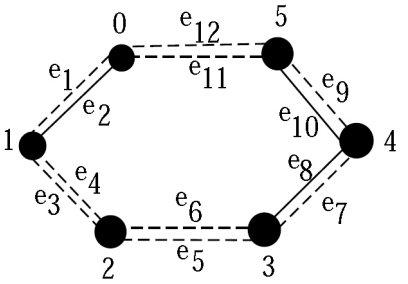

Figure 3
Multi-graph S'

Generating the key

The following function $\phi : E' \to \mathbb{R}$ has been shared with the receiver and will be used as the key for both encryption and decryption.

$$\phi(v_i v_j) = \phi(w_{ij}) = \begin{cases} 5w_{ij} + 6 & \text{if } i \in U_n \text{ or } j \in U_n, \\ 2w_{ij} - 1 & \text{otherwise.} \end{cases}$$

It is important to note that in $\phi(v_i v_j) = \phi(w_{ij})$, w_{ij} refers to the weight of the edge $v_i v_j$.

Encryption of the graph S'

To encrypt the graph S', determine the values of the function ϕ for all edges in S'.

$$\phi(e_1) = \phi(w_{e_1}) = 5(-1) + 6 = 1$$

$$\phi(e_2) = \phi(w_{e_2}) = 5(1) + 6 = 11$$

$$\phi(e_3) = \phi(w_{e_3}) = 5(-1) + 6 = 1$$

$$\phi(e_4) = \phi(w_{e_4}) = 5(-1) + 6 = 1$$

$$\phi(e_5) = \phi(w_{e_5}) = 2(-1) - 1 = -3$$

$$\phi(e_6) = \phi(w_{e_6}) = 2(-1) - 1 = -3$$

$$\phi(e_7) = \phi(w_{e_7}) = 2(-1) - 1 = -3$$

$$\phi(e_8) = \phi(w_{e_8}) = 2(1) - 1 = 1$$

$$\phi(e_9) = \phi(w_{e_9}) = 5(-1) + 6 = 1$$

$$\phi(e_10) = \phi(w_{e_10}) = 5(1) + 6 = 11$$

$$\phi(e_11) = \phi(w_{e_11}) = 5(-1) + 6 = 1$$

$$\phi(e_12) = \phi(w_{e_12}) = 5(-1) + 6 = 1$$

All the weights of edges are encrypted and we can construct the encrypted graph S'' with the new edges as shown in (**Figure** 4).

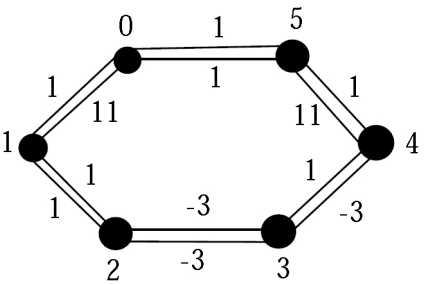

Figure 4

Encrypted graph S''

Decryption of the graph S''

In first step we are calculating ϕ^{-1} as ϕ is already shared with the receiver and $\phi^{-1} : E'' \to \mathbb{R}$ such that:

$$\phi^{-1}(v_i v_j) = \phi^{-1}(w_{ij}) = \begin{cases} \frac{1}{5}(w_{ij} - 6) & \text{if } i \in U_n \text{ or } J \in U_n, \\ \frac{1}{2}(w_{ij} + 1) & \text{otherwise.} \end{cases}$$

It is important to note that in $\phi^{-1}(v_i v_j) = \phi^{-1}(w_{ij})$, w_{ij} refers to the weight of the edge $v_i v_j$ in E''.

Now, determine the values of ϕ^{-1} for all the edges of S''.

$$\phi^{-1}(e_1) = \phi^{-1}(w_{e_1}) = \frac{1}{5}(1 - 6) = \frac{1}{5}(-5) = -1$$

$$\phi^{-1}(e_2) = \phi^{-1}(w_{e_2}) = \frac{1}{5}(11-6) = \frac{1}{5}(5) = 1$$

$$\phi^{-1}(e_3) = \phi^{-1}(w_{e_3}) = \frac{1}{5}(1-6) = \frac{1}{5}(-5) = -1$$

$$\phi^{-1}(e_4) = \phi - 1(w_{e_4}) = \frac{1}{5}(1-6) = \frac{1}{5}(-5) = -1$$

$$\phi^{-1}(e_5) = \phi^{-1}(w_{e_5}) = \frac{1}{2}(-3+1) = \frac{1}{2}(-2) = -1$$

$$\phi^{-1}(e_6) = \phi^{-1}(w_{e_6}) = \frac{1}{2}(-3+1) = \frac{1}{2}(-2) = -1$$

$$\phi^{-1}(e_7) = \phi^{-1}(w_{e_7}) = \frac{1}{2}(-3+1) = \frac{1}{2}(-2) = -1$$

$$\phi^{-1}(e_8) = \phi^{-1}(w_{e_8}) = \frac{1}{2}(1+1) = \frac{1}{2}(2) = 1$$

$$\phi^{-1}(e_9) = \phi^{-1}(w_{e_9}) = \frac{1}{5}(1-6) = \frac{1}{5}(-5) = -1$$

$$\phi^{-1}(e_10) = \phi^{-1}(w_{e_10}) = \frac{1}{5}(11-6) = \frac{1}{5}(5) = 1$$

$$\phi^{-1}(e_11) = \phi^{-1}(w_{e_11}) = \frac{1}{5}(1-6) = \frac{1}{5}(-5) = -1$$

$$\phi^{-1}(e_12) = \phi^{-1}(w_{e_12}) = \frac{1}{5}(1-6) = \frac{1}{5}(-5) = -1$$

In next step, remove all the edges of S_n from the graph S', the resulting graph is the original graph S.

5. Complexity of the Algorithm

The complexity at the first stage in the above algorithm depends on the size of the input graphs S and S_n.

Step 1 requires a constant amount of space and time, so its time & spacecomplexity is $\mathcal{O}(1)$.

Step 2 involves adding all the edges of S to S_n, which takes time & spaceproportional to the number of edges in S, denoted by $|E|$. Thus, the time & spacecomplexity of Step 2 is $\mathcal{O}(|E|)$.

Step 3 involves creating a new graph with n_1 vertices and $|E'|$ edges, where $|E'| = |E| + |E_1|$ is the number of edges in E' after Step 2. The time & spacecomplexity of Step 3 is therefore $\mathcal{O}(n_1 + |E'|) = \mathcal{O}(n_1 + |E| + |E_1|)$.

Therefore, the overall time & spacecomplexity of this stage in the algorithm is $\mathcal{O}(n_1 + |E| + |E_1|)$. The time & spacecomplexity of generating the key in this algorithm depends on the time complexity of the two trap-door one-way functions ϕ_1 and ϕ_2, as well as the size of the unit set U_n.

Step 1 involves choosing two trap-door one-way functions ϕ_1 and ϕ_2. The time & spacecomplexity of this step depends on the specific functions chosen and is not specified in the algorithm.

Step 2 involves creating a key function ϕ that combines ϕ_1 and ϕ_2 based on the set U_n. Since U_n is a unit set, the size of U_n is constant, and the time & spacecomplexity of combining ϕ_1 and ϕ_2 based on U_n is also constant. Therefore, the time & spacecomplexity of Step 2 is $\mathcal{O}(1)$.

Step 3 involves outputting the key function ϕ as the private key, which takes constant space and time. Therefore, the time & spacecomplexity of Step 3 is also $\mathcal{O}(1)$.

Therefore, the overall time & spacecomplexity of generating the key in this algorithm is dependent on the time & spacecomplexity of the two trap-door one-way functions and can be expressed as $\mathcal{O}(f(n))$, where $f(n)$ depends on the time & spacecomplexity of the two functions and the specific implementation of the algorithm. The time & spacecomplexity of encryption of S' in this algorithm depends on the size of the vertex set and edge set of S', as well as the time & spacecomplexity of the trap-door one-way functions ϕ_1 and ϕ_2.

Step 1 involves iterating through all the edges in S' and performing four sub-steps for each edge. Each sub-step involves computing the weight of the edge and performing some operations on it. The time & spacecomplexity of each sub-step depends on the time & spacecomplexity of the trap-door one-way functions ϕ_1 and ϕ_2 and the specific operations being performed, which is not specified in the algorithm. Therefore, the time & spacecomplexity of Step 1 can be expressed as $\mathcal{O}(m \times g(n))$, where m is the number of edges in S', and $g(n)$ is the time complexity of the sub-steps.

Step 2 involves setting the new edge weights and creating a new graph S''. Since this step involves iterating through all the edges in S' and setting their weights, the time & spacecomplexity of Step 2 is also $\mathcal{O}(m)$.

Therefore, the overall time & spacecomplexity of encrypting S' in this algorithm can be expressed as $\mathcal{O}(m \times g(n))$, where m is the number of edges in S' and $g(n)$ is the time & spacecomplexity of the sub-steps, which depends on the specific operations being performed and the time & spacecomplexity of the trap-door one-way functions. The time & spacecomplexity of the decryption algorithm depends on the complexity of computing the inverse function of ϕ, which in turn depends on the complexity of the trap-door one-way functions ϕ_1 and ϕ_2. Assuming that the inverse function of ϕ can be computed in polynomial time, then the time & spacecomplexity of the decryption algorithm can be expressed as:

$$\mathcal{O}(|E''| \times t_{\phi^{-1}})$$

where $|E''|$ is the number of edges in the encrypted graph S'', and $t_{\phi^{-1}}$ is the time & spacecomplexity of computing the inverse function of ϕ.

Note that decrypting each edge requires computing the inverse function of ϕ, so the total time & spacecomplexity of the decryption algorithm is proportional to the number of edges in the encrypted graph.

6. Conclusion

Signed Cayley graphs offer enhanced security due to their inherent structure and the incorporation of edge signatures. Unlike traditional graph encryption methods that may rely solely on vertex or edge permutations, the inclusion of signed edges provides an additional layer of authentication and integrity verification. This helps mitigate risks associated with unauthorized access or tampering of encrypted data.

Our proposed method can also be implemented on datasets. To apply the proposed method to datasets, we initially partition the datasets in a prescribed manner, let's say we have r partitions. We then arrange the partition values in an upper triangular matrix according to a prescribed scheme in [22].

To ensure that we construct an upper triangular square matrix from these partitions, r should satisfy $r = \frac{n^2-1}{2}$, where n is the order of the upper triangular matrix. Thus, to ensure n is a positive integer, we add zero partitions to the end of the dataset if needed.

Let A be a symmetric matrix with the same upper triangular entries as the constructed matrix, and with diagonal entries set to 0. We then construct a graph S using this matrix as its adjacency matrix and each

entry as the weight of that edge. We proceed to encrypt graph S to S'' using the proposed encryption scheme.

During decryption, we reconstruct graph S from S'' using the proposed decryption scheme. We then convert graph S to its adjacency matrix A, and extract only the upper triangular elements, arranging them in the prescribed manner as described in [22]. This new arrangement constitutes the required dataset.

By following this methodology, we ensure the correctness of the encryption and decryption processes, maintaining the integrity of the dataset.

References

[1] D. Stinson and M. Paterson, Cryptography: Theory and Practice, ser. Textbooks in Mathematics, vol. 2. Chapman and Hall/CRC Press (2018). [Online]. Available: https://books.google.co.in/books?id=nHxqDwAAQBAJ.

[2] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal,* vol. 28, no. 4, pp. 656–715, 1949. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1949.tb00928.x.

[3] V. Taghvayi-Yazdelli and M. Ghorbani, "Characterization of bipartite cayley graphs in terms of boolean functions," *Journal of Information and Optimization Sciences,* vol. 39, no. 7, pp. 1567–1582 (2018). [Online]. Available: https://doi.org/10.1080/02522667.2017.1384181.

[4] P. 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). [Online]. Available: https://doi.org/10.1080/09720529.2013.878819.

[5] O. Wardak, D. Sinha, and A. Sethi, "Encryption and decryption of signed graph matrices through RSA algorithm," *Indian Journal of Pure and Applied Mathematics* (2023). [Online]. Available: https://doi.org/10.1007/s13 226–023–00 452–9.

[6] W. Al Etaiwi, "Encryption algorithm using graph theory," *Journal of Scientific Research and Reports*, vol. 3, pp. 2519–2527 (2014).

[7] V. A. Ustimenko, "Graphs with special arcs and cryptography," *Acta Applicandae Mathematica,* vol. 74, pp. 117–153 (2002).

[8] M. Ghorbani and F. N. Larki, "On the spectrum of finite cayley graphs," *Journal of Discrete Mathematical Sciences and Cryptography,*

vol. 21, no. 1, pp. 83–112 (2018). [Online]. Available: https://doi.org/10.1080/09720529.2018.1449797.

[9] A. Sethi, D. Sinha, and O. Wardak, "Algorithmic approach to find s-consistency in common-edge signed graph," *MethodsX*, vol. 9, p. Article ID 101783 (2022). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215016122001637.

[10] S. G. Akl and I. Assem, "Fully homomorphic encryption: a general framework and implementations," *International Journal of Parallel, Emergent and Distributed Systems,* vol. 35, no. 5, pp. 493–498 (2020). [Online]. Available: https://doi.org/10.1080/17445760.2018.1553041.

[11] C. Gentry, "Computing arbitrary functions of encrypted data," *Communications of the Association for Computing Machinery* (ACM), vol. 53, no. 3, pp. 97–105 (Mar. 2010). [Online]. Available: https://doi.org/10.1145/1666420.1666444.

[12] A. Kasten, A. Scherp, F. Armknecht, and M. Krause, "Towards search on encrypted graph data," Proceedings of the 2013th International Conference on Society, Privacy and the Semantic Web - Policy and Technology, vol. 1121, pp. 46–57 (2013).

[13] Q. Wang, K. Ren, M. Du, Q. Li, and A. Mohaisen, "Secgdb: Graph encryption for exact shortest distance queries with efficient updates," in Financial Cryptography and Data Security, A. Kiayias, Ed. Cham: Springer International Publishing, pp. 79–97 (2017).

[14] Z. Cao and L. Liu, "On the weakness of fully homomorphic encryption," ArXiv, vol. abs/1511.05341 (2015). [Online]. Available: http://arxiv.org/abs/1511.05341.

[15] K. E. Makkaoui, A. Ezzati, and A. B. Hssane, "Challenges of using homomorphic encryption to secure cloud computing," Proceedings of International Conference on Cloud Technologies and Applications (CloudTech), pp. 1–7 (2015).

[16] D. Sinha and A. Sethi, "Encryption using network and matrices through signed graphs," *International Journal of Computer Applications*, vol. 138, no. 4, pp. 6–13 (Mar. 2016).

[17] B. Ni, Q. Rabiha, S. U. Rehman, and F. Ghulam, "Some graph-based encryption schemes," *Journal of Mathematics,* vol. 2021, p. Article ID 6614172 (Feb. 2021). [Online]. Available: https://ideas.repec.org/a/hin/jjmath/6614172.html.

[18] M. Acharya and D. Sinha, "Common-edge sigraphs," *AKCE International Journal of Graphs and Combinatorics,* vol. 3, no. 2, pp. 115–130 (2006). [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/09728600.2006.12088809.

[19] D. Sinha and P. Garg, "On the unitary Cayley signed graphs," *Electronic Journal of Combinatorics*, vol. 18, no. 1, p. Article ID P229 (2011).

[20] G. A. Selim, "How to encrypt a graph," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 35, no. 6, pp. 668–681 (2020). [Online]. Available: https://doi.org/10.1080/17445760.2018.1550771.

[21] S. G. Akl, "The graph is the message: design and analysis of an unconventional cryptographic function," in From Parallel to Emergent Computing, CRC Press, pp. 425–442 (2019).

[22] J. Katz and Y. Lindell, Introduction to Modern Cryptography, ser. Chapman & Hall/CRC Cryptography and Network Security Series. CRC Press (2021). [Online]. Available: https://books.google.co.in/books?id=RsoOEAAAQBAJ.

Received July, 2023