International Journal of Computer Applications (0975 – 8887)
Volume 138 – No.4, March 2016

Encryption using Network and Matrices through Signed Graphs

Deepa Sinha
South Asian University,
Akbar Bhawan, Chanakyapuri,
New Delhi – 110 021.

Anshu Sethi
Center for Mathematical Sciences,
Banasthali University, Banasthali – 304 022

ABSTRACT

Security of a network is important to all organizations, personal computer users, and the military. With the invention of the Internet, major concern is about the security and the history of security allows a better understanding of the emergence of security technology. One of the ways to secure businesses from the Internet is through firewalls and encryption mechanisms. A *network* can be designed as a sigraph S where every sigraph will have its unique adjacency matrix associated with it. A *signed graph* (or *sigraph* in short) S is a graph G in which every edge x carries a value $s(x) \in \{-1, +1\}$ called its *sign* denoted specially as $S = (G, s)$. Given a sigraph S, $H = L(S)$ called the *line sigraph* of S is that sigraph in which edges of S are represented as vertices, two of these vertices are adjacent whenever the corresponding edges in S have a vertex in common and any such edge ef is defined to be *negative* whenever both e and f are negative edges in S. Here S is called *root sigraph* of H. In this paper first we give an algorithm to obtain a line sigraph [**1**] and line root sigraph from a given sigraph **[1]**, if it exists. This *algorithm* is an extension of an algorithm given by **Lehot [2]** in the realm of sigraphs. In the end we will propose a technique that will use adjacency matrix of S as a parameter to encrypt and forward the data in the form of adjacency matrix of $L(S)$ and will decrypt it by applying inverse matrix operations.

Keywords

Algorithm, sigraph, line sigraph, root sigraph, sign-compatible, network, network security, encryption, decryption.

1. INTRODUCTION

For notations in *signed graphs*, we refer to **Zasalavsky [3]** and West **[4]** and for *algorithms*, refer to **Golumbic [5]** and **Coreman [6]**. Throughout the text, we consider finite, undirected graph with no loops or multiple edges. By a (n, e) graph G we mean a graph having n vertices and e edges; n is called the *order* and e is called the *size* of G. Any graph G can be observed as a network where vertices are referred as nodes and edges as *links*. For any network communication must be secured because the communication signals are openly available as they propagate.

By a *signed graph* (or *sigraph*) S **[3, 7]** we mean a graph $G = (V, E)$ called the underlying graph of S and denoted by S^u, in which each edge x carries a value $s(x) \in \{+1, -1\}$ called its *sign*; an edge x is positive or negative according to whether $s(x) = +1$ or $s(x) = -1$. The set of positive edges of S is denoted by $E^+(S)$ and $E^-(S) = E(G) - E^+(S)$ is the set of negative edges of S. In general, a subgraph S' of a sigraph S is said to be all-positive (all-negative) if all the edges of S' are positive (negative). A sigraph is said to be *homogeneous* if it is either all-positive or all-negative and *heterogeneous* otherwise.

For a sigraph S, **Behzad and Chartrand [8]** defines its *line sigraph*, $L(S)$ as the sigraph in which the edges of S are represented as vertices, two of these vertices are defined adjacent whenever the corresponding edges in S have a vertex in common, any such edge ef is defined to be negative whenever both e and f are negative edges in S. A given sigraph S is a line sigraph if it is isomorphic to the line sigraph $L(T)$ of a sigraph T. Here T is called the line root of S.

An adjacency matrix for a network with ‘n’ vertices and no parallel edges is an n x n symmetric matrix such that

$$a_{ij} = \begin{cases} 1 & \text{if (i, j) is a solid line} \\ -1 & \text{if (i, j) is a dotted line} \\ 0 & \text{if (i = j)} \end{cases}$$

When network security is considered, one must ensure that the whole network is secured. Data Security is the major factor to secure data transmission over unreliable network. Data Security is a challenging issue of data communications today that touches many areas including secure communication channel, strong data encryption technique and trusted third party to maintain the database. The conventional methods of encryption can only maintain the data security. The unauthorized user could access the information for malicious purpose. Therefore, it is necessary to apply effective encryption/decryption methods to enhance data security. In this paper we are going to introduce a new method for encoding and decoding of data using network as sigraph and basic properties of matrices.

For the purpose of network security, adjacency matrix of S will be considered as basis of information which is to be encrypted to adjacency matrix of $L(S)$ to assure confidentiality, integrity and authentication of transmitted data.

A sigraph S is *sign-compatible* **[9]** if there exists a marking μ of its vertices such that the end vertices of every negative edge receive ‘-’ signs in μ and no positive edge in S has both of its ends assigned ‘-’ sign in μ. In other words, a sigraph is sign-compatible if and only if its vertices can be partitioned into two subsets V_1 and V_2 such that the all-negative subsigraph of S is precisely the subsigraph induced by exactly one of the subsets V_1 and V_2. Every line sigraph is sign-compatible. However, not every sign-compatible sigraph need be line sigraph.

Based on concept *sign-compatible sigraphs***[9]**, *characterization of line sigraphs* **[10]** and *algorithm to detect a line graph and output its root graph by **Lehot*** **[2]**, we will provide a computer-oriented characterization of line sigraph that will output the root sigraph S of sigraph H (sometimes called $L(S)$) whenever the latter is a line sigraph.

Since, for a network we use nodes in case of vertices and link for edges, so while giving algorithms for the characterization, nodes and links are used.

2. CHARACTERIZATION OF LINE SIGRAPHS

By a *positive*(*negative*) *section* of a subgraph *S'* of a sigraph *S* we mean a maximal edge-induced connected subsigraph in *S'* consisting of only the positive(negative) edges of *S*; in particular, a positive(negative) section in a heterogeneous cycle of *S* is essentially a maximal all-positive(all-negative) path in the cycle. Using the definition, following is a characterization of line sigraphs by **Acharya and Sinha** [see [10]].

Theorem 1. [10] The following statements are equivalent for any sigraph*S*:

(i) *S* is a line sigraph ;

(ii) S^u is a line graph and for any two vertices $u, v \in V(S)$ and for any $u - v$ path in *S* either $u = v$ and the cycle so created is not a triangle with exactly two negative edges or the u - v path contains no positive section of length one unless one of its ends is *u* or *v*;

(iii) *S* does not contain an induced subsigraph isomorphic to either of the two sigraphs, S_1 formed by taking the path $P_4 = (x, u, v, y)$ with both the edges (x, u) and (v, y) negative and the edge (u, v) positive and S_2 formed by taking S_1 and identifying the vertices *x* and *y* as shown in **Fig 1**, or to any sigraph on Beineke's nine forbidden subgraphs;

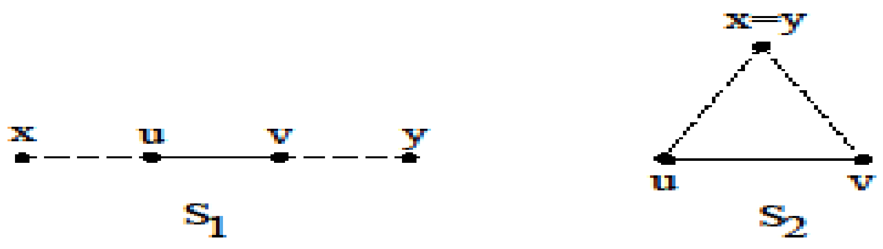


Fig 1: Condition to check sign-compatibility

(iv) S^u is a line graph and for any positive edge *uv* of *S* either there is no negative edge at *u* or there is no negative edge at *v*;

(v) S^u is a line graph and vertices of *S* can be assigned signs '+' or '-' such that both the ends of every negative edge receive '-' sign and the same is not true for any positive edge;

(vi) S^u is a line graph and the vertex set $V(S)$ of *S* can be partitioned into two subsets V_1 and V_2, one of them possibly empty, such that all the negative edges of *S* join vertices of just one of the subsets.

3. ALGORITHM TO CONVERT A SIGRAPH TO LINE SIGRAPH [1, 11]

3.1 Part – 1:

Here input matrix *vertex*[*i*][*j*] can be either upper triangular matrix or lower triangular matrix with entry 0 if there is no edge and 1 if edge exists. For each non-zero entry in *vertex*[*i*][*j*], we define *signver*[*i*][*j*] to be either upper triangular or lower triangular matrix with entries 0 and 1, 0 will represent negative edge and 1 represents positive edge.

Step1. Enter the number of vertices *n* of input sigraph *S* whose line sigraph *L*(*S*) is required.

Step2. Input the lower triangular matrix (say) *vertex*[*i*][*j*] and sign matrix *signver*[*i*][*j*] of sigraph *S*.

Step3. Instead of 1 assign distinct numbers at all those positions of matrix where there is a non-zero entry.

Step4. Now search for non-zero entries in given matrix.

(i) Now, for each non zero entry of the given matrix, say (*i*, *j*)th, search for non-zero entries in row *i* and column *i*. For each such non-zero entry, say (*i*, *k*)th entry in *i*th row, there corresponds an edge in *L*(*S*), in this case edges (*i*, ,*j*) and (*i*, *k*) in *S* would be two adjacent vertices of *L*(*S*).

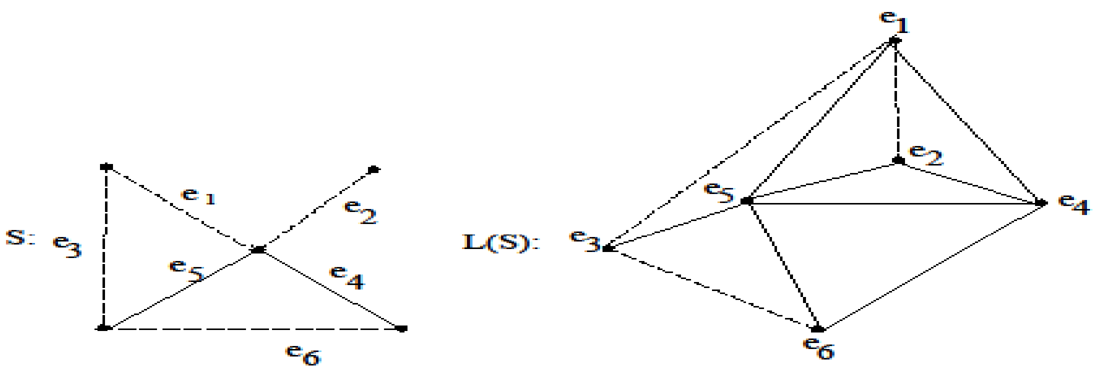


Fig 2: Example showing sigraph and its line sigraph

(ii) Now sign of this edge in $L(S)$ depends on sign of edges (i, j) and (i, k) in S. If both the entries in matrix signver are zero then corresponding edge in $L(S)$ would be negative otherwise it would be positive.

Step5. The sigraph so produced is required line sigraph $L(S)$ of given sigraph S as shown in **Fig 2**.

Complexity of computation involved in above algorithm

In Step3 we have to assign distinct numbers to all the non-zero entries in adjacency matrix. Since we have entered lower triangular matrix, thus we need to check $n(n - 1)/2$ entries.

Hence complexity for this step is $O(n^2)$.

Then in Step4 first we have to search for non-zero entries in lower triangular matrix and then corresponding to each such

entry, say (i, j)th entry, we have to search for non -zero entries in row i and column i. Also we have to check signs of these entries.

Thus complexity of this step = $O(n^2 \times n) = O(n^3)$.

Hence complexity of computation involved in above algorithm is $O(n^3)$, where n is number of vertices in S.

3.2 Part – 2

Here input matrix *vertex*[*i*][*j*] can be either upper triangular matrix or lower triangular matrix with entries 0, 1 and -1 for no edge, positive edge and negative edge respectively.

Step1. Enter the number of vertices n of input sigraph S whose line sigraph $L(S)$ is required.

Step2. Input lower triangular matrix (say) *vertex*[*i*][*j*] with entries 0, 1 and -1 for no edge, positive edge and negative edge respectively of sigraph S.

Step3. Now search for non-zero entries in matrix.

(i) Now, for each non-zero entry of the matrix, say (i, j)th entry, search for non zero entries in row i and column i. For each such non zero entry, say (i, k)th entry in ith row, there corresponds an edge in $L(S)$, in this case edges (i, j) and (i, k) in S would be two adjacent vertices of $L(S)$. Now sign of this edge in $L(S)$ depends on sign of edges (i, j) and (i, k) in S. If both the entries in matrix are -1 then corresponding edge in $L(S)$ would be negative otherwise it would be positive.

Step4. The sigraph so produced is required line sigraph $L(S)$ of given sigraph S as shown in **Fig 2**.

Complexity of computation involved in above algorithm

In Step3 we have to search for non-zero entries in lower triangular matrix of order n x n and then corresponding to each such entry, say (i, j)th entry, we have to search for non-zero entries in row i and column i. Also we have to check signs of these entries.

Hence complexity for this step is $O(n^2)$.

Hence complexity of computation involved in above algorithm is $O(n^2)$, where n is number of vertices in S.

3.3 Conclusion

Since we can implement the line sigraph from a given sigraph in two ways. First part takes input in the form of two matrices and with complexity $O(n^3)$ whereas second part takes only one matrix as input and with complexity $O(n^2)$. Thus we can say PART - 2 is optimal in space and time complexity and will be used as a parameter to encrypt and forward the data. Encryption is applied at this step.

4. ALGORITHM TO DETECT A LINE SIGRAPH AND OUTPUT ITS ROOT SIGRAPH [1, 11]

The algorithm to detect a line sigraph and output its root sigraph is an extension of a paper named "*An Optimal Algorithm to Detect a Line Graph and Output its Root Graph*" by **Lehot [2]**.

To check whether a given sigraph is a root linesigraph or not we have to check two conditions:

i. The underlying graph is a line graph.

ii. Given sigraph is sign-compatible.

If both the conditions are satisfied we say that root linesigraph exists and will print the new root sigraph of S.

Newnode represents the node of the intermediate graph i.e. Fig - b and lookup represents if current index is mapped to which node. It shows the mapping between Fig - a and Fig - b. Since each node represents 2 points, maximum amount of numbers required is 2* Max where Max denotes maximum value of n.

Newgraph denotes the adjacency matrix of required root sigraph and corresponding to this matrix Fig - c is plot.

Find the first non-zero entry in adjacency matrix i.e. graph[*i*][*j*]. For the first edge (i, j) create a new node $(i, 1, 2) \rightarrow$ Node i named as pair (1, 2) i.e. in Fig - b. Now start traversing the graph from this node. Also populate the final graph with this edge i.e. (1, 2).

Now traverse through the graph and name other nodes. For each untraveled neighbor create a modified node for Fig - b. Find an $i1$ for new node. We want to find j as the first index $i1$, we look if j is assigned to any node, do we have any edge. For e.g. if $j = 2$ and current nodes are (a, (1, 2)), (b, (2, 3)) and (c, (1, 4)) we check if current node has edge from a to b. Once we have find j we want to find k as second index. If j and k are found we have found the naming of $i1$ and $i2$. Add $i1$ and $i2$ to newnode. Make curidx minimum of (curidx, j, k). Record that j and k have been assigned to current node (i).

If pairing of edges can be done as given by Lehot, then we say that graph is a line graph otherwise not a line graph and hence not a root linesigraph.

This pair would represent an edge in newgraph. If this node has negative edge anywhere, set this edge as negative else positive. Also add this node to the queue as we want to travel its neighbor.

Next step we will check whether a given sigraph is sign-compatible or not. If it is sign-compatible, root line sigraph exists and print the new modified graph otherwise root line sigraph does not exists.

Step wise procedure is shown in **Fig 3**.

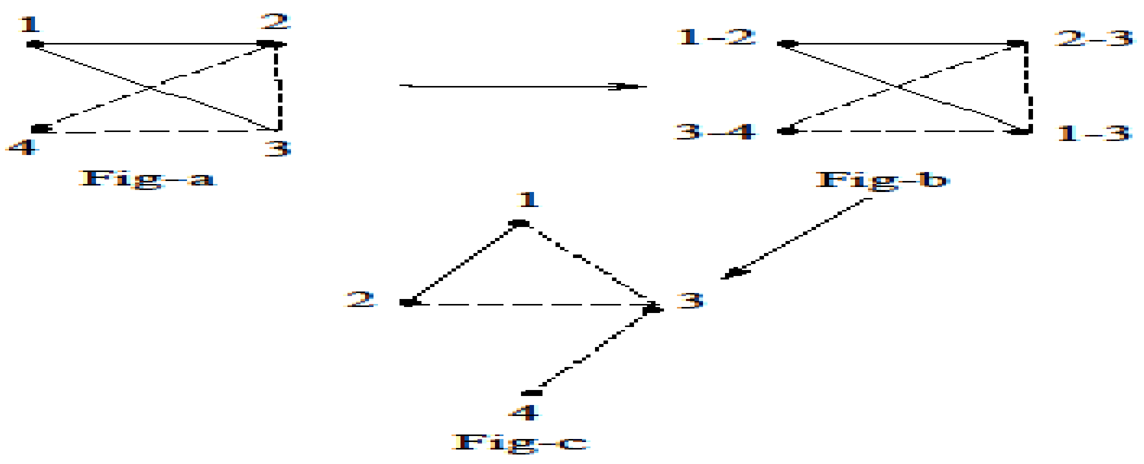


Fig 3: Stepwise procedure to find root line sigraph

Following algorithmis used as a decryption technique to recover the original information. It uses following functions:

MAX denotes maximum number of vertices

Structure of Node :idx; // Primary and i1, i2; // The pair

isTraversed is a function which represents whether node is travelled or not.

Lookup defines if the current index is mapped to which node. Now, since each node is given 2 numbers, the maximum amount of numbers required is 2*MAX.

newGraph represents required root sigraph.

Step1. Input a sigraph *H* with entries 0, 1 and -1 whose root sigraph*S* is required.

Step2. Find the first edge i.e repeat step for *i*= 0 to MAX and *j*= 0 to MAX and till queue is not empty.

(i) Check if (*graph*[*i*][*j*] ! = 0) if yes then create a new node (*i*, 1, 2). This node *i* is named as pair(1, 2). If no Go to Step3.

(ii) Again start traversing the graph from this node.

(iii) Initialize weight = 1.

(iv) Repeat for *l*=0 to MAX and weight=1 If true assign weight = -1 else weight= 1.

(v) Populate the final graph with this edge i.e (1 ,2)

newGraph[newNodes[*i*].*i*1][newNodes[*i*].*i*2] = wieght;

newGraph[newNodes[*i*].*i*2][newNodes[*i*].*i*1] = wieght;

Step3. Now traverse through the graph and name other nodes. Traverse till queue is not empty.

3.1 Set current node to queue.front

3.2 Repeat for *i* = 0 to MAX For each untravelled neighbor

3.2.1 create new node

3.2.2 set newnode.idx = current node i.ei

3.2.3 Choose *i*1 and *i*2 such that newGraph (adjacency matrix = lookup) and graph are consistent. First loop – Try with the indices tried until now

3.2.3.1 Look for *i*1 such that if (*i*1, l) is an edge, then (curNode, l) should be an edge

3.2.3.2 Look for *i*2 such that if (*i*2, l) is an edge, then (curNode, l) should be an edge

3.2.3.3 Set curIdx as the minimum of *i*1 and *i*2

3.2.3.4 Add edges to newGraph from *i*1 and *i*2 to *i*. This pair would represent an edge in new graph. If this node has negative edge anywhere, set this edge as negative else positive.

3.2.3.5 Add the new node to the queue.

3.2.3.6 Set the current node as travelled. Second loop - If we go inside this loop, it means we did not find any *i*1 and *i*2 <curIdx.

So we do an unbounded search. The steps are same, try with indices not use until now.

Step4. Check if (curIdx> DIM) then print “Graph is not line graph” and exit else Print “Graph is a line graph”.

Step5. Since sigraph is a line graph now we have to check whether it is signcompatible.

5.1 Set for *i* = 1 to *n* all counts to 0.

5.2 Set for *i* = 1 to *n* and *j* = 1 to *n* and Count -1 in each row and column.

5.3 Set for *i* = 1 to *n* and *j* = 1 to *n*

5.3.1 Search for first 1 and check if (row[*i*] ≥ 1 and (col[*j*] ≥ 1) If yes, Print “Sigraph is not sign-compatible” and exit else goto Step8.

Step6. Print sigraph is signed compatible and hence its root line sigraph exists.

Step7. Print the newGraph(which is the required root linesigraph).

Step8. Exit

Complexity of computation involved in above algorithm

In Step1 since we input a graph of order *n* x *n*, complexity of this step = $O(n^2)$.

In Step2 we have to find first non-zero entry (i.e. first edge) in adjacency matrix of order *n* x *n* and then corresponding to each such entry, say $(i, j)^{th}$ entry, we again have to traverse the graph to find its adjacent node and push the node in the queue.

Thus complexity of this step = $O(n^3)$.

In Step3 since we have to traverse the graph and name all other nodes till queue is not empty. Queue contains the edges that are adjacent to first edge.

This maximum number of edges is $e = n(n - 1)/2$.

Thus complexity of this step = $O(e / n^3)$.

In Step5 we detect whether the given sigraph is sign-compatible or not. We count number of adjacent negative edges in adjacency matrix of order n x n.

Maximum number of edges can be $O(n^2)$.

Thus complexity of this step = $O(n^2)$.

Total complexity = $O(n^2) + O(n^3) + O(e / n^3) + O(n^2) = O(n^3)$.

Hence complexity of computation involved in above algorithm is $O(n^3)$, where n is number of vertices in S and e is the number of edges.

5. CONCLUSION

Security of the data is the major concern in data transmission over unreliable network. In this method of data transmission in a network (say S), data is taken in the form of adjacency matrix with entries as 0, 1 and -1. Then algorithm defined in Section 3 is used as an encryption mechanism for transmitting the data. We get $L(S)$ as the data which is transmitted. Further the decryption of this transmitted data to original data will be implemented through algorithm defined in Section 4. There exists several techniques on encryption and decryption, but we have developed a new technique in which weak and strong relationships among nodes in a network can be defined. The following technique is proposed that will use adjacency matrix of S as a parameter to encrypt and forward the data in the form of $L(S)$ and will decrypt it by applying inverse matrix operations. Thus, we can say that, this is a new technique, which can be used in encryption and decryption of a network. For the purpose of security, we have used asymmetric key cryptography.

5.1 System Model

Asymmetric cryptography is used in this model. Two different keys, a secret key d and a public key e are defined. The public key e is used for encryption i.e. for converting adjacency matrix of S to adjacency matrix of $L(S)$. Since we have a unique method for encryption, therefore, it can be published. Further, the secret private key d is used for decryption of adjacency matrix of $L(S)$ to adjacency matrix of S. Since we have many line root sigraphs of a given sigraph and we have to restrict our networks to obtain a unique line root sigraph, therefore, private key is used. Also, if labelling of vertices can be done by **Lehot [2]**, then only we obtain unique line root sigraph. The following model used as an application to above algorithms is shown in **Fig 4**.

Given a network (or sigraph) with negative and positive edges representing weak and strong relationships among nodes is first encoded in the form of adjacency matrix with entries 0, 1 and -1 representing no edge, positive edge and negative edge respectively. Adjacency matrix is the plain text. Encryption uses an algorithm called a cipher key (defined in Section3) to encrypt matrix into another matrix which can be decrypted only using a special key (defined in Section 4). Encrypted information is known as cipher text and the process of obtaining the original information(plaintext) from the cipher text is known as decryption.

Note:

For more security reasons, we can add one additional key matrix

K =

1 111

1 11 0

1 1 0 0

1 0 00

This key is multiplied to the adjacency matrix of $L(S)$ to obtain the encrypted data. This encrypted data is send to the receiver. Now again multiply this encrypted data with the inverse of matrix K. Here

K^{-1} =

0 00 1

0 0 1 −1

0 1 −1 0

1 −1 0 0

and apply decryption algorithm to obtain the original message.

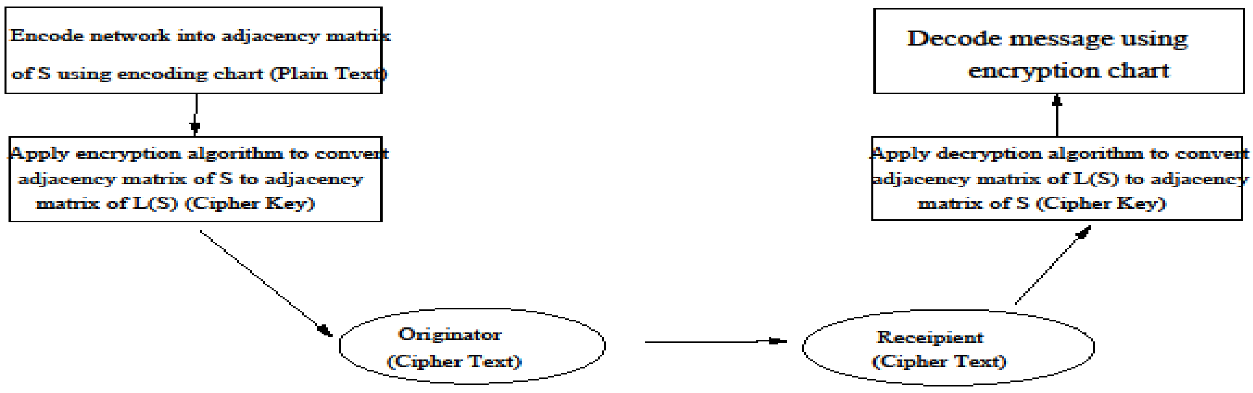
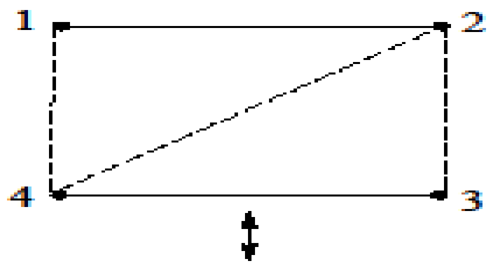

Encode network into adjacency matrix of S using encoding chart (Plain Text)
Apply encryption algorithm to convert adjacency matrix of S to adjacency matrix of L(S) (Cipher Key)
Originator (Cipher Text)
Receipient (Cipher Text)
Apply decryption algorithm to convert adjacency matrix of L(S) to adjacency matrix of S (Cipher Key)
Decode message using encryption chart

Fig 4: System Model

5.2 Encoding Chart

For a network with '*n*' number of vertices, we have *n* x *n* adjacency matrix encoded as shown in **Fig 5**.



$$\begin{matrix} 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & -1 \\ 0 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{matrix}$$

Fig 5: Example showing how network is encoded in the form of adjacency matrix

5.3 Encryption Algorithm

- •Input the network which is to be encrypted. Then encode this network into adjacency matrix of *S*.
- Populate edge list. Initially set number of edges to 0.
- Count number of non-zero entries in lower triangular matrix. Name it as *m*.
- This count will represent number of rows or vertices for the resultant matrix i.e. *L*(*S*).
- Apply algorithm defined in Section 3 to convert adjacency matrix of *S* to adjacency matrix of *L*(*S*).
- Multiply adjacency matrix of *L*(*S*) with *K* to get higher security to the data.
- The resultant matrix so produced is required line sigraph *L*(*S*) of given sigraph *S*.
- The resultant matrix is the encrypted data.
- Now send the resultant matrix to the receiver in a linear format (i.e., either column wise or row wise) with space between elements.

nn < Resultant matrix data >*mm*
where,
n = number of vertices or nodes in the network
m = number of edges in the network.

5.4 Decryption Algorithm

- Read the encrypted data and form the required matrices of order m x m.
- Multiply the obtained matrix with K−1.
- The matrix thus obtained is again the adjacency matrix of L(S).
- Apply "Algorithm to detect a line sigraph and output its root sigraph" defined in Section 4 to convert adjacency matrix of L(S) to adjacency matrix of S.
- Decode the resultant matrix back to the network using encoding chart.
- The network so produced is the original network.

Note:

The reading and writing of data can be done manually or by using file operation of any programming language (eg. C, C++....).

5.5 Example

In the example to be discussed here we have used C++ Program for all type of matrix operations.

Suppose we have a network. The network is represented by dotted and solid lines representing weak and strong relationships among the various nodes.

Consider a network in **Fig 6**:
We represent this network in the form matrix *T* of the adjacency matrix of *S*.
Let

$$T= \begin{matrix} 0 & 1 & -1 & 0 \\ 1 & 0 & -1 & 0 \\ -1 & -1 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{matrix}$$

be the adjacency matrix for the above network such that 1 represent solid line and -1 shows dashed line.
Populate edge list i.e. count number of non-zero entries in matrix *T*. Let this number be '*m*'. Then the resultant matrix after encrypting will be of order *m* x *m*. Here in our example it is 4. Thus the resultant matrix is of order 4 x 4 which is adjacency matrix of *L*(*S*).

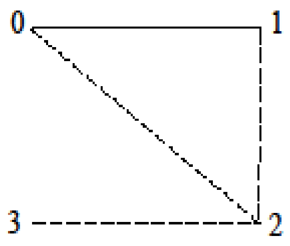


Fig 6: Network

Now, for each non zero entry of the matrix *T*, say (*i*, *j*)th entry, search for non zero entries in row *i* and column *i*. For each such non zero entry, say (*i*, *k*)th entry in *i*th row, there corresponds an edge in *L*(*S*), in this case edges (*i*,*j*) and (*i*,*k*) in *S* would be two adjacent vertices of *L*(*S*). Now sign of this edge in *L*(*S*) depends on sign of edges (*i* ,*j*) and (*i*, *k*) in *S*. If both the entries in matrix are -1 then corresponding edge in *L*(*S*) would be negative otherwise it would be positive.

Thus the resultant matrix is adjacency matrix of *L*(*S*):

$$T' = \begin{matrix} 0 & 1 & 1 & 0 \\ 1 & 0 & -1 & -1 \\ 1 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 \end{matrix}$$

The corresponding graph is shown in **Fig 7**:

If we decide to transmit the data row wise, the data to be send is (the data are separated by space)

4 4 0 000 1 0 00 -1 -1 0 000 -1 0 4 4

Suppose the received data is

4 4 0 1 1 0 1 0 -1 -1 1 -1 0 -1 0 -1 -1 0 55

From the data we have received we get the following matrix:

$$T' = \begin{matrix} 0 & 1 & 1 & 0 \\ 1 & 0 & -1 & -1 \\ 1 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 \end{matrix}$$

isomorphic to some sigraph.

Now applying the decryption algorithm for *L*(*S*) to *S* as shown in example we get the matrix as

$$\begin{matrix} 0 & 1 & -1 & 0 \\ 1 & 0 & -1 & 0 \\ -1 & -1 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{matrix}$$

Which is equal to the matrix *T*

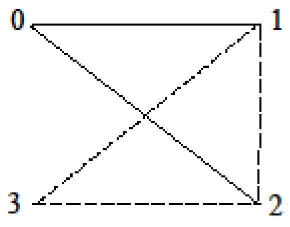


Fig 7: Encrypted Network

Thus the resulting sigraph after complete encryption and decryption is shown in **Fig 8**:

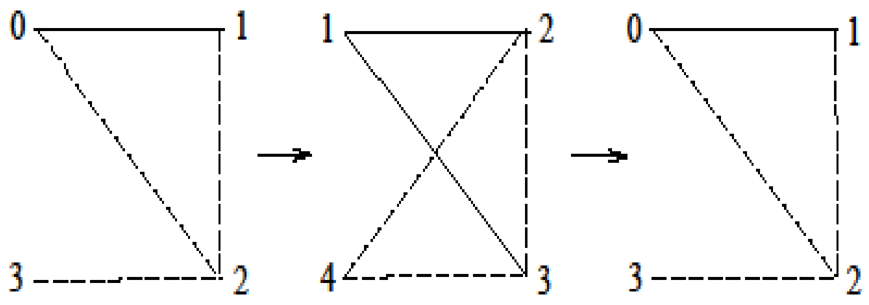


Fig 8: Encrypted and Decrypted network

Note:

As defined above, one can easily obtained a more secured network by multiplying *T*' by *K*. The matrix so produced is the encrypted data which is to be transmit row-wise.

If we decide to transmit the data row wise, the data to be send is (the data are separated by space)

4 4 0 1 -1 0 1 0 -1 0 -1 -1 0 -1 0 0 -1 0 4 4

Suppose the received data is

4 4 2 2 1 0 -1 0 1 1 -1 0 0 1 -2 -2 -1 0 5 5

Multiply this received data with K^{-1}, we get,

T'' =

0 1 1 0

1 0 -1 -1

1 -1 0 -1

0 -1 -1 0

isomorphic to some sigraph. This matrix is same as the matrix of *L*(*S*).Follow the same decryption mechanism as above and we get back the same original network. Thus, network is made more secured due to additional secret key.

6. ACKNOWLEDGMENTS

The authors express gratitude to Mr. Dhananjay Kulkarni who help in writing algorithms and finding complexity and to the referees who made extensive and constructively critical comments on the first version of the paper.

7. REFERENCES

[1] Sinha, D. andSethi. A 2015, An Algorithm to detect S-Consistency in Line Sigraph, Journal of Combinatorics, Information & System Sciences: Vol 40, No. 1-4 Comb. (Jan-Dec 2015).

[2] Lehot, P.G.H. 1974. An optimal algorithm to detect a line graph and output its root graph, Journal of the Association for Computing Machinery, 21 (4), (1974), 569-575.

[3] Zasalavsky, T. 1982. Signed graphs, Discrete Appl.Math, 4 (1) (1982), 47-74.

[4] West, D.B. 1996. Introduction to Graph Theory, Prentice-Hall of India Pvt. Ltd., 1996.

[5] Golumbic, M.C. 2004. Algorithmic Graph Theory and Perfect Graphs, Fourth Edition.

[6] Cormen, T., Leiserson C., Rivest, R., Stein, C. 2011. Introduction to algorithm, Third Edition, PHI Learning Private Limited, 2011.

[7] Chartrand, G. T. 1977. Graphs as Mathematical Models, Prindle, Weber and Schmidt, Inc., Boston, Massachusetts, 1977.

[8] Behzad, M. and Chartrand, G. T. 1969. Line coloring of signed graphs, Elem. Math., 24(3) (1969), 49-52.

[9] Sinha, D. and Sethi. A 2015, An Optimal Algorithm toDetect Sign Compatibility of a given Sigraph, National Academy of Science Letters, DOI 10.1007/s40009-014-0317-5, 2015.

[10] Acharya, M. and Sinha, D. 2005. Characterizations of Line sigraphs, Nat. Acad. Sci. –Letters., 28 (1 - 2) (2005), 31-34. [Also, see Extended Abstract in: Electronic Notes in Discrete Mathematics, 15 (2003).

[11] Sinha, D. and Sethi. A 2015, An Algorithmic Characterization of sigraphs whose common edgesigraphs and second iterated line sigraphs are switching equivalent, Journal of Discrete Mathematical Sciences &Cryptography DOI: 10.1080/09720529.2015,1013679,Vol. 18(2015), No. 5, pp. 581-603.