Encryption Algorithm Using Graph Theory

Article *in* Journal of Scientific Research and Reports · January 2014

DOI: 10.9734/JSRR/2014/11804

1 author:

Wael Etaiwi
Princess Sumaya University for Technology
35 PUBLICATIONS **502** CITATIONS

Journal of Scientific Research & Reports
3(19): 2519-2527, 2014; Article no. JSRR.2014.19.004

SCIENCEDOMAIN *international*
www.sciencedomain.org

Encryption Algorithm Using Graph Theory

Wael Mahmoud Al Etaiwi[1*]

[1]*Information Technology Directorate, Jordan Customs Department, Amman, Jordan.*

Author's contribution

The sole author designed, analyzed and interprets and prepared the manuscript.

Original Research Article

Received 4th June 2014
Accepted 15th July 2014
Published 7th August 2014

ABSTRACT

In the recent years, with the increase of using Internet and other new telecommunication technologies, cryptography has become a key area to research and improve in order to transfer data securely between two or more entities, especially when the data transferred classified as a critical or important data. Even there are many encryption algorithms exist, the need of new non-standard encryption algorithms raise to prevent any traditional opportunity to sniff data. The proposed algorithm represents a new encryption algorithm to encrypt and decrypt data securely with the benefits of graph theory properties, the new symmetric encryption algorithm use the concepts of cycle graph, complete graph and minimum spanning tree to generate a complex cipher text using a shared key.

Keywords: Encryption; cryptography; graph theory.

1. INTRODUCTION

In this paper, we consider an undirected graph G(V,E), where V is the set of vertices and E is a set to edges that connect vertices each other. A walk from one vertex to another in which each vertex not appears more than once called a path. A cycle appears when the path (walk) starts from one vertex and return back to the same vertex, when the cycle consists of all vertices in the graph we called it a cycle graph. The graph called complete graph when there is an edge between any two vertexes in the graph.

Graphs represented in two main ways, adjacency-list and adjacency-matrix. The adjacency-list representation of graph G(V,E) consists of an array of V lists, one for each vertex in V.

**Corresponding author: E-mail: waelcis@yahoo.com;*

For each vertex v, the adjacency-list of v contains all vertices adjacent (there is an edge between them) to him. The adjacency-matrix representation of graph G(V,E) consists of a |V| * |V| matrix g_{ij} such that:

$$g_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$, for un-weighted graph.

Or

$$g_{ij} = \begin{cases} wij & \text{if } (i,j) \in E \\ NIL & \text{value otherwise} \end{cases}$$, for weighted graph.

When there is a connected sub-graph contains all vertices with the minimum weight of edges required, this sub-graph (tree) called a spanning tree. The problem of defining the minimum spanning tree called Minimum Spanning Tree (MST) problem, and there is many algorithms to solve this problem like: Kruskal algorithm and Prime algorithm [1].

Cryptography is the art of protect information by transforming it to unreadable format called Cipher text. The process of converting plain text to cipher text called encryption, and the process of converting cipher text on its original plain text called decryption. Cryptography algorithms classified mainly into two major types: Symmetric-key cryptography and public-key (Asymmetric) cryptography. In Symmetric-key cryptography, each sender and receiver shared the same key used to encrypt and decrypt data with disadvantage of key management required to keep the key secure. The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are examples of Symmetric-key cryptography methods. In public-key cryptography, each sender and receiver use two different keys to encrypt and decrypt data – public key and private key-, the public key can be freely distributed, while its paired private key must remain secret. In public-key cryptography, we overcome the key management distribution issue of Symmetric-key cryptography, but at the expense of performance speed.

2. RELATED WORK

Yamuna M et al. [2] presented an encryption mechanism using Hamilton path properties (path that covers all vertices in the graph), they encrypt data twice, once using the Hamilton path, and the second using the complete graph to impose more secure method. Rick Kilma and Neil Sigmon showed how graph can be used in cryptanalysis of Vigenere cipher (cryptanalysis is analyzing encrypted information to get the plain text without knowing the encryption keys). Ustimenko VA [3] used symbolic computations technique to create a public key mode based on algebraic graphs that can be used for the implementation of secure and fast symmetric encryption algorithm. In [4], they present a method of using paths between a pair of graph vertices for designing polyalphabetic substitution ciphers, and also they modify the labels of vertices or edges (arcs) of the graph in order to influence the statistical properties and period lengths. Steve Lu et al. [5] use an arbitrary graph where every node and every edge are assigned an arbitrary image; Images on the vertices are “public” and images on the edges are “secret”, using this approach, pixel expansion and contrast are proportional to the number of images.

3. PROPOSED ALGORITHM

The first step in this algorithm is to represent data as vertices in a graph, each character represented by a vertex while all adjacent characters in the data will be represented as adjacent vertices in the graph, we keep adding vertices until we form a cycle graph. Every edge in the graph has its own weight represents the distance of these two characters in the encoding table–table used to encode all alphabets characters-.Then each vertex in the graph will joined with edges to make the graph a complete graph, while every new added edge has a sequence weight started from the last index in the encoding table. Adjacency-Matrix is constructed for the complete graph. After that Minimum Spanning Tree (MST) is computed from the complete graph and represented as adjacency-matrix that keeps data characters order in its diagonal. Adjacency-matrix of the complete graph multiplied to the adjacency-matrix of MST. The resultant matrix multiplied to the key matrix. The final matrix is the encryption data to be sent to the recipient.

Encryption Algorithm:

- Add a special character to indicate the starting character (Let A).
- Add vertex for each character in the plain text to the graph.
- Link vertices together by adding an edge between each sequential character in the plain text until we form a cycle graph.
- Weight each edge using the encoding table. Each edge's weight represents the distance between the connected two vertices from the encoding table.
- Adding more edges to form a complete graph M1, each new added edge has a sequential weight starting from the maximum weight in the encoding table.
- Then find the Minimum Spanning Tree M2.
- Then store the vertices order in the M2 matrix in the diagonal places.
- Then we multiply matrices M1 by M2 to get M3.
- After that we multiply M3 by a predefined Shared-Key K to form C.
- Then the cipher text contains Matrix C and Matrix M1 line-by-line in a linear format.

Decryption Algorithm:

- The receiver computes M3 by using the inverse form of the Shared-Key K-1.
- Then compute M2 by using the inverse form of M1.
- Then compute the original text by decoding M1 using the encoding table.

Fig. 1 below summarizes the encryption and decryption algorithms.

Example:

Suppose we want to encrypt the message WAEL to send it to the receiver.

The first step is to convert the message to a graph, by converting each character to a vertex as shown in Fig. 2.

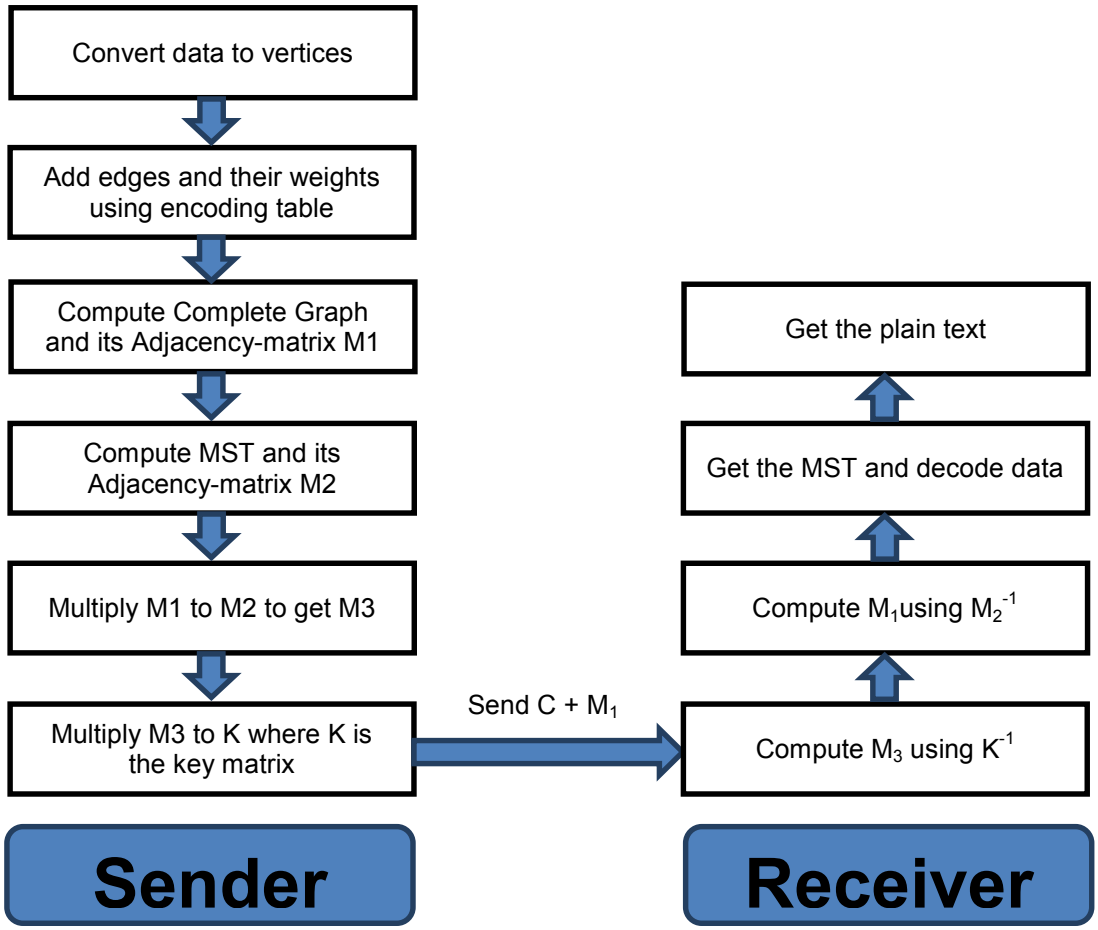

Fig. 1. Algorithm summary

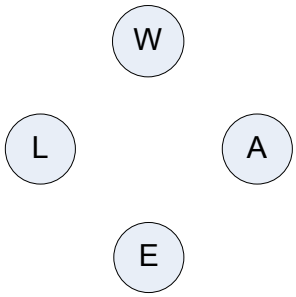

Fig. 2. Convert each character to vertex

Then, link each two sequential characters together to form a cycle graph.

Then, weight each edge using the bellow encoding table:

Table 1. Encoding table

A	B	C	D	E	…	L	…	W	X	Y	Z
1	2	3	4	5	…	12	…	23	24	25	26

Each edge's weight represents the distance between the connected two vertices from the encoding table, so the edge connect vertex W with vertex A has weight the distance between the two letters in the table as the follow:

Distance = code(A) – code(W)
= 1 – 23
= -22

And so on, then the graph will be as shown in the Fig. 3 below:

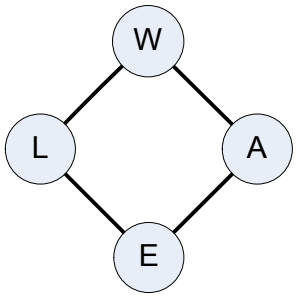

Fig. 3. Graph contains plain text characters.

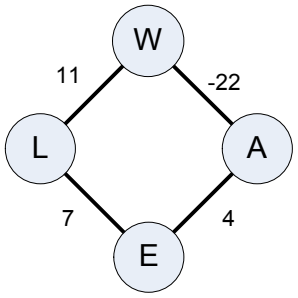

Fig. 4. Weighted graph contains plain text characters

After that, we keep adding edges to form a complete graph; each new added edge has a sequential weight starting from the maximum weight in the encoding table (26 +1 = 27). See Fig. 5.

Here we add a special character before the first character to point to the first character, let A. as shown in Fig. 6.

The complete plain graph in Fig. 5 represented as a matrix M_1

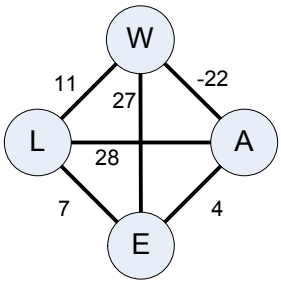

Fig. 5. Complete plain graph

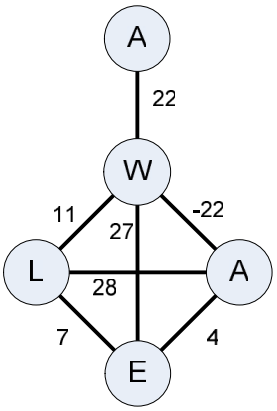

Fig. 6. Complete plain graph with a special character

$$M_1 = \begin{bmatrix} 0 & 22 & 0 & 0 & 0 \\ 22 & 0 & -22 & 27 & 11 \\ 0 & -22 & 0 & 4 & 28 \\ 0 & 27 & 4 & 0 & 7 \\ 0 & 11 & 28 & 7 & 0 \end{bmatrix}$$

Then we find the minimum spanning tree (MST) as Fig. 7 shows.

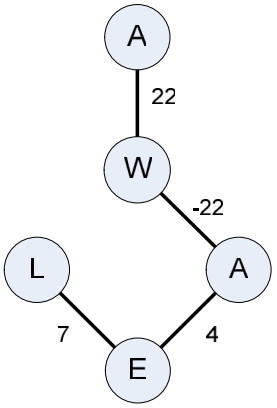

Fig. 7. Minimum spanning tree graph

$$M_2 = \begin{bmatrix} 0 & 22 & 0 & 0 & 0 \\ 22 & 0 & -22 & 0 & 0 \\ 0 & -22 & 0 & 4 & 0 \\ 0 & 0 & 4 & 0 & 7 \\ 0 & 0 & 0 & 7 & 0 \end{bmatrix}$$

Now, we store the characters order in the diagonal instead of 0's.

Character	**A**	**W**	**A**	**E**	**L**
order	0	1	2	3	4

$$\text{So, } M_2 \text{ modified to } \begin{bmatrix} 0 & 22 & 0 & 0 & 0 \\ 22 & 1 & -22 & 0 & 0 \\ 0 & -22 & 2 & 4 & 0 \\ 0 & 0 & 4 & 3 & 7 \\ 0 & 0 & 0 & 7 & 4 \end{bmatrix}$$

After that, we multiply matrix M_1 by M_2 to form M_3.

$$M_3 = M_1M_2 = \begin{bmatrix} 484 & 22 & -484 & 0 & 0 \\ 0 & 968 & 64 & 70 & 233 \\ -484 & -22 & 500 & 208 & 140 \\ 594 & -61 & -586 & 65 & 28 \\ 242 & -605 & -158 & 133 & 49 \end{bmatrix}$$

Now, we use the shared-key K to encrypt M_3.

$$\text{Let } K = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{ so cipher text } C = \begin{bmatrix} 836 & 302 & -664 & 476 & 450 \\ 352 & 280 & -180 & 476 & 450 \\ 352 & -688 & -244 & 406 & 217 \\ 836 & -666 & -744 & 198 & 77 \\ 242 & -605 & -158 & 133 & 49 \end{bmatrix}$$

Now the data to be send is C + M_1:

836 302 -664 476 450 352 280 -180 476 450 352 -688 -244 406 217 836 -666 -744 198 77 242 -605 -158 133 49

In the receiver side, we get M_3 by multiplying the cipher text received by the inverse form of the shared key K^{-1}.

$$M_3 = CK^{-1} = \begin{bmatrix} 836 & 302 & -664 & 476 & 450 \\ 352 & 280 & -180 & 476 & 450 \\ 352 & -688 & -244 & 406 & 217 \\ 836 & -666 & -744 & 198 & 77 \\ 242 & -605 & -158 & 133 & 49 \end{bmatrix} * \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 484 & 22 & -484 & 0 & 0 \\ 0 & 968 & 64 & 70 & 233 \\ -484 & -22 & 500 & 208 & 140 \\ 594 & -61 & -586 & 65 & 28 \\ 242 & -605 & -158 & 133 & 49 \end{bmatrix}$$

Then calculate M_2 by multiplying M_3 by M_1^{-1}:

$$M_2 = M_3 M_1^{-1} = \begin{bmatrix} 0 & 22 & 0 & 0 & 0 \\ 22 & 1 & -22 & 0 & 0 \\ 0 & -22 & 2 & 4 & 0 \\ 0 & 0 & 4 & 3 & 7 \\ 0 & 0 & 0 & 7 & 4 \end{bmatrix}$$

So, M_2 represents the following final graph (Fig. 8) (regardless the diagonal) that we use to retrieve the original message:

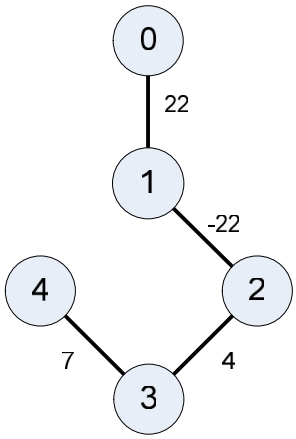

Fig. 8. Final graph

We suppose that the node 0 is A, So by using encoding table, node 1 =code(A) + 22 = 23 that represent character W, and node2 = code(W) – 22 = 1 that represent character A, and so on until we got the original text (WAEL).

4. IMPLEMENTATION AND EXPERIMENTAL RESULTS

The environment of the Microsoft.Net was selected to develop and test the proposed algorithm. More specifically, the algorithm was implemented on an Intel® Core™ i7 with speed 2.10 GHz processor and 6 GB RAM, running on the Microsoft Windows 7 Enterprise Service Pack 1 64-bit operating system.

Experiments on selected plain text which represent different length plain texts were contacted in order to check the algorithm's accuracy. The table below (Table 2) represents some testing experiments using a random encryption key. The following observations were extracted:

- The public key length can be in any different size, since the algorithm can apply any key length by expanding it to the required length by duplicating it.
- Cipher text size increase when the plain text become larger, that mean that the algorithm is more efficient when the plain text message is small.
- The required time to encrypt text increase too because of the matrix multiplication process.

Table 2. Testing experimental results

Plain text size	Cipher text size	Time / Milliseconds
4	118	15
6	269	18
12	1016	41
19	2767	105
28	6512	199
56	30318	947

5. CONCLUSION AND FUTURE WORK

In this paper, we present a new cryptography algorithm that can be implemented using any programming language like: C++, JAVA or Microsoft.Net. This algorithm used to encrypt data to be transmitted using an encoding table and graph theory properties which is complete graph and minimum spanning tree. This symmetric cryptography algorithm uses the concept of shared-key that must be predefined and shared between sender and receiver. The public-key cryptography could be used for more complexity and security by using two keys, one for encrypt and another for decrypt.

Many improvements can be done in the future related to decrease matrices required to encrypt and decrypt messages, and to decrease the cipher text size such as: divide large message into small blocks of messages and encrypt each block separately and concatenate cipher texts to form the whole message cipher text. On the other hand, distribute version may be used to reduce encryption time and optimize the overall algorithm performance.

COMPETING INTERESTS

Author has declared that no competing interests exist.

REFERENCES

1. Corman TH, Leiserson CE, Rivest RL, Stein C. Introduction to algorithms 2nd edition, McGraw-Hill.
2. Yamuna M, Meenal Gogia, Ashish Sikka, Md. Jazib Hayat Khan. Encryption using graph theory and linear algebra. International Journal of Computer Application. ISSN:2250-1797; 2012.
3. Ustimenko VA. On graph-based cryptography and symbolic computations, Serdica. Journal of Computing. 2007;131-156.
4. Paszkiewicz A, et al. Proposals of graph based ciphers, theory and implementations. Research Gate; 2001.
5. Steve Lu, Rafail Ostrovsky. Daniel Manchala. Visual Cryptography on Graphs, CiteSeerx, COCOON. 2008;225-234.

Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sciencedomain.org/review-history.php?iid=622&id=22&aid=5656