International Journal of Information & Network Security (IJINS)
Vol.4, No.1, February 2015, pp. 14~24
ISSN: 2089-3299

Graph Structured Data Security using Trusted Third Party Query Process in Cloud Computing

Prakash G L*, Dr. Manish Prateek† and Dr. Inder Singh‡
* Research Scholar, Department of Computer Science and Engineering, UPES, Dehradun
†Associate Dean, Center for Information Technology, UPES, Dehradun,
‡ Assistant Professor, Center for Information Technology, UPES, Dehradun

Article Info

Article history:

Received Dec 12th, 2014
Revised Jan 20th, 2015
Accepted Feb 26th, 2015

Keyword:

Security
Graph structured data
Feature
Trapdoor

ABSTRACT

Cloud computing enables the users to outsource and access data economically from the distributed cloud server. Once the data leaves from data owner premises, there is no privacy guarantee from the untrusted parties in cloud storage system. Providing data privacy for the outsourced sensitive data is a challenging task in cloud computing. In this paper we have proposed a Trusted Third party Query Process(TTQP) method to provide data privacy for graph structured outsourced data. This method utilize the encrypted graph frequent features search index list to search the matched query graph features in graph data base. The proposed system has analyzed in terms of different size of data graphs, index storage, query feature size and query execution time. The performance analysis of our proposed system shows, this method is more secure than the existing privacy preserving encrypted Query Graph(PPQG) method.

Corresponding Author:

Prakash G L,
,Department of Computer Science and Engineering,
UPES, Dehradun
Email: glprakash78@gmail.com

1. INTRODUCTION

In the recent year, the data size is increases exponentially with time. To maintain and process these data in cloud system is a challenging task. Cloud computing is the new computing model which enables to outsource user data and process more economically.

In cloud computing, since the data owner and the cloud service providers are no longer in trusted domain, outsourc- ing unencrypted more sensitive information such as; emails, personal health records, and financial data, etc. will be in risk at centralized cloud server. To provide data privacy and access control of sensitive information, before outsourcing data owner has to encrypt the data using secure encryption method. To provide the security and utilize the outsourced data more effectively data encryption is one of the challenging task in cloud computing. Share the outsourced data to the large number of authorized users, who needs only specific data, one of the way is the ranked keyword based search[1].

The existing keyword search searchable encryption method [1] allows users to search data without decrypting the en- crypted file. This method restricted to only the text documents and without pre knowledge of encrypted cloud data requires more computational overheads for cloud server to retrieve the requested and matched data.In order to retrieve the graph structured sensitive data from the cloud server a feature such as tree and path based graph retrieval methods are designed. The graph structured data applications includes Bio-informatics, chemistry, social networking and UML diagrams etc.

In chemical applications, the graphs are represents different types of chemical compounds. Consider a graph data base(G) consists of three data graphs and query graph (Q), retrieve all the graphs $g_i \epsilon G$ such that $Q \epsilon g_i$ as shown in Figure 1 and the sample of graph feature-based index as shown in Figure 2.

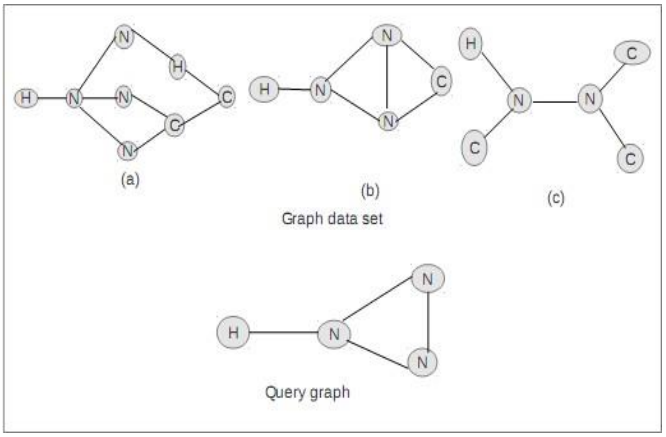


Fig. 1. Sample of Data Graph set and Query Graph

When the graph size is increases the number of graph features are also increases larger and larger. Due to this growth of graph features, cloud server consumes more computational and communication overheads to retrieve all the data graphs of the query features. To reduce these overheads during searching process, there are many graph filtering and verification meth- ods are introduced [2], [3].

In filtering phase, using feature based index set of graph data set G, generate the set of candidate super graph set, which consists a query feature. In verification phase, generates the set of super graphs from the candidate super graph set which matches the exact query graph [2].

The rest of the paper is organized as fallows; In section 2, explain the background for data graph representation, In section 3, defines the problem and the design goals of our proposed security system. The detailed architecure, algorithms and performance analysis is explained in the section 4 and section 5, respectively and Section 6 explain the related work. Finaly, concluding our proposed system.

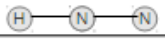
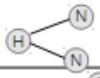
Features(F)	Graph features	Graph ID list
f1		{a,b,c}
f2		{a,b}
f3		{c}

Fig. 2. Data Graph Index list for fraph feature.

2. BACKGROUND

A. Data set Representation

A undirected and connected data graph is represented as $G =< V, E >$, where V is the set of graph vertices, E is the set of graph edges and $V(G)$ represent the total number of vertices in the graph G i.e. |size of the graph G. The graphs are represented in computer memory using adjacency matrix of $(n*n)$ matrix M with entries $Mij = 1$, if there is an edge from vertex i to vertex j in the graph, otherwise $Mij = 0$.

Consider a two graphs G and H, if they have a common adjacency matrices then they are isomorphic graphs [4]. If graph H is a sub graph of G then we call it as graph G contains graph H. Given a graphs $G = \{G1, G2, ...Gm\}$ and a sub graph Q, if Q contains k number of graphs in G, then k is called as frequency(minsup) of Q in G.

B. Frequent Feature Construction

Consider a data graph G of size is measured in terms of number of vertices $V(G)$ and the graph feature F_i such as sub trees and path of size $V(F_i)$. As the number of vertices increase in the data graph the number of graph features also increases with different size. To store all the features index and processing of these features cloud server consumes more storage and computation overheads. In order to reduce the feature set size, only different feature size will be consider with different frequency of the feature in the data graph to construct the index set. The various frequent feature sub structures construction algorithms such as gSpan[5], Breadth First Search and Depth First Search are defined [5].

3. PROBLEM FORMULATION

A. System Model

The cloud security system model [6] consists of data user, data owner, cloud service providers and trusted trapdoor query process components as shown in the Figure 3.

Data user: is an entity to represents individuals or organi- zations who access the data graphs stored in the cloud server via trusted query auditor.

Cloud Service Provider: is an entity that retrieves the encrypted data graphs of the user request by computing the inner product of trapdoors of the query and the frequent feature set of the data graphs.

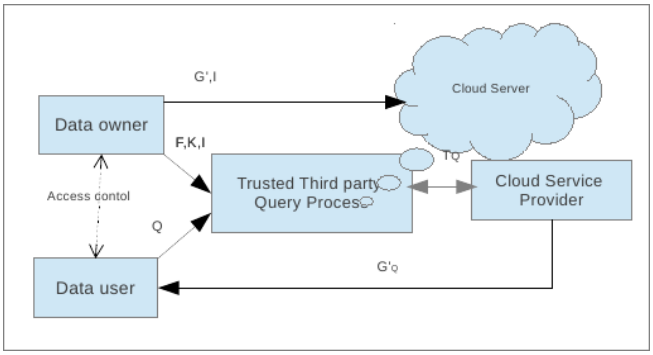


Fig. 3. Cloud System Model.

The data owner: is an entity to provide the data as a service to the user. In our proposed system model before outsourcing the encrypted data graphs to the cloud server, the data owner generate the symmetric key (Ks) frequent feature set (F) and data graph index list(I_{inv}), then outsource the encrypted data graphs and the index list to the cloud service provider. One copy of the index list, symmetric key and the feature set send to the trusted query auditor.

Trusted Third party Query Process(TTQP): is an entity, which is trusted by all other components of the system. For the data privacy in cloud server, which is capable to generate the trapdoor for every user query using index list and the symmetric key generated by the data owner. In our system the cloud service provider and the trusted query auditor are always online during the data retrieval process while the data owner can be an offline.

B. Design objectives

For effective utilization of out sourced and encrypted graph structured data, we are considered the following design objectives in the proposed cloud data security method.

- To prevent clod server to learn graph features to access data graph encrypt all the features.
- Design an efficient query graph feature encryption method than data graph encryption.
- Minimize the computation and storage overheads of the cloud server while reducing frequent feature set size and introducing third party as trusted query pro- cess.

C. Notation

The various notations are used in this paper are listed in Table 1.

4. PROPOSED TRUSTED THIRD PARTY QUERY PROCESS SYSTEM

The proposed cloud security system model as defined in Figure 4, Before out sourcing the encrypted data graphs G' the data owner generate the inverted list of data graph identifier

i.e. $ID(G)$ for every data graph frequent feature (F) using Ullman graph isomorphic algorithm [bbc] . Derive a symmetric.

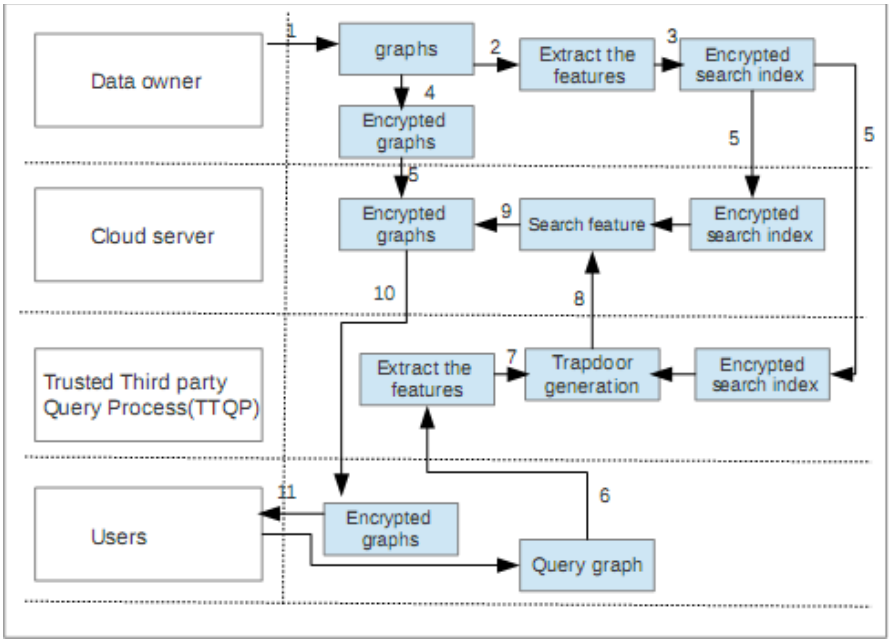


Fig. 4. Block Diagram of Proposed Data Security System

TABLE I. NOTATIONS

Notation	Meaning
G	Graph data set G={G1, . . . Gm}
G′	Encrypted graph data
F	Feature set F={F1, . . . Fn}
I	Data graph index set
Q	Query graph
TQ	Trapdoor for the query graph Q
g	Data graph vector
q	Query graph vector
K	Symmetric key
FQ	Query features

key $K = S, M1, M2, ro$ from secrete key (Ks), where S is the split vector contains a sequence of n binary values, $M1$ and $M2$ are the two invertible matrix of $(n + 1)$ size, and ro is the set of n random integer values.

A. Graph Search Index list generation

Consider a set of m data graph $G = \{G1, G2, ..Gm\}$ and a set of n frequent features of a given graph G is $\{F = f1, f2, ...fn\}$ to generate the index list for every data graph using the inverted list (I_{inv}) of data graph as fallows and the detailed algorithm for data graph index construction as explined in Table II.

- Create a data vector gi of size n for each data graph and If the identifier of graph G_i i.e $ID(G_i)$ includes in the inverted list then $g_i[j]$ contains the value stored in the symmetric key i.e $K[4][j]$, otherwise $g_i[j]$ contain any random value.
- To hide the data graphs frequent features from the untrusted parties in system, extend the data graph vector gi from n dimension to $(n + 1)$ dimension and set it as 1.
- Split the data vectors g_i in to two data vectors g_i' and gi of same size based on the splitting vector S in the symmetric key K[1][j]. If the split vector S[j] is equal to 0, then set gi′ and gi″ as original value of gi,otherwise set it as sum of the gi′ equal to gi.
- Encrypt the data vectors g_i' and g_i'' by computing the inner product of g_i' and the invers of the invertible matrix $M1$ specified in the symmetric key $K[2]$ generate the index for the data vector g_i' is I_i' and the computing the inner product of g_i'' and the invers of the invertible matrix M2 specified in the symmetric key $K[3]$ generate the index for the data vector g_i'' is I_i''.
- Create the index list for the data graph g_i as $I_i = \{I_i', I_i''\}$

B. Trapdoor generation

Once the data is stored in the cloud server, to provide privacy for out sourced data from the untrusted parties in the cloud system, trusted trapdoor query process encrypt all the graph features using following procedure. The detailed algorithm for trapdoor generation is explained in the Table III.

- For every query Q from the user, generate the query feature set (F_Q) by checking the user query Q is included in the data graphs frequent feature set or not.
- Create a n dimensional query vector q and if F_j contains F_Q then it set as a positive random number r_j otherwise set it as 0.
- For the privacy of the query features, extend the query vector q from n dimension to $(n + 1)$ and set it as a random number t.
- Split the query vector q in to q' and q'' based on the split vector S in the symmetric key $K[1]$. If $S[j]$ is equal to 1 then set $q'[j]$ and $q''[j]$ as $q[j]$, otherwise set its sum is equal to the $q[j]$.
- Encrypt the query vectors q' and q'' by computing the inner product of q' and the invertible matrix $M1$ specified in the symmetric key $K[2]$ generate the trap- door for the query vector q' is T_{Q1} and by computing the inner product of q'' and the invertible matrix $M2$ specified in the symmetric key $K[3]$ generate the trapdoor for the query vector q'' is T_{Q2}.
- for the verification of data graph for the user query in the cloud server compute the trapdoor T_{Q3} $T_{Q3} = \sum K[4][j].q[j] + t$

- Create the trapdoor list $T_Q = \{T_{Q1}, T_{Q2}, T_{Q3}\}$

TABLE II. GRAPH FEATURE INDEX LIST GENERATION

Algorithm: Encrypt_Data_features(K, I_{inv} , I_i)

Input: Symmetric key (K) and Inverted list (I_{inv}) of data graph features

Output: Encrypted search index I_i

1. Initialize the matrix *X[i][j]* of size (*nxn*) with random value which is less than *K[4][j]*
2. Create a data vector g_i for each super graph
   If($ID(G) \in I_{inv}[j]$) then
   $g_i[j] = K[4][j]$
   Else
   $g_i[j] = X[i][j]$
   Endif
3. Extend every data vector g_i from n to *(n+1)* dimension and set to 1, $g_i[j] = 1$
4. Split the data vector g_i in to two vectors according to the splitting indicator *S* in the symmetric key *K*
   If *(K[1][j] == 0)* then
   $g_i'[j] = g_i''[j] = g_i[j]$
   else
   $g_i'[j] + g_i''[j] = g_i[j]$
   Endif
5. Encrypt the data vectors gi' and gi" using inverted matrix M1 and M2 in symmetric key and return the index I
   $I_i' = ((K[2])^{-1})^T . g_i'$
   $I_i'' = ((K[3])^{-1})^T . g_i''$
   $I_i = \{I_i', \quad I_i''\}$
6. Return the encrypted index list I_i, where *i= 1* to *m*

To retrieve all the encrypted super graphs for the query graph, cloud server computes the inner product of Query feature trapdoor vector with ever data graph feature index of cloud server, if this product matches with the TTQP inner product T_{Q3}, then cloud server returns all the encrypted super graphs which matches the query features to the user. The detailed encrypted data graph retrieval procedure is explained in the algorithm as shown in Table IV.

5. PERFORMANCE ANALYSIS

The performance analysis of the proposed graph structured data security system is analyzed in terms of index storage size, frequent feature set size, data and query vector encryption time and query process time as explained as fallows.

***Index storage size*:** for the analysis of index storage size of frequent feature set, we are considered four different data graph set $G = \{10, 20, 30, 40, 50, 60\}$ and each graph contains 10 distinct vertices. The maximum number of features in the query graph set to four in our analysis.

TABLE III. TRAPDOOR GENERATION FOR A QUERY

Algorithm : Generate_Trapdoor (Q, F_Q, F_G, K, T_Q)

Input: Query graph (Q), data graph frequent feature set (F_G), symmetric key K

Output : T_Q-trapdoor for a query feature.

1. Generate the query feature set (F_Q) from the query Q and frequent feature set (F_G).
2. Create a query vector q for user query graph Q
   If $(F_j \in F_Q)$ then
   $q[j] = r_j$ where r_j is any random number
   Else
   $q[j] = 0$
   Endif
3. Extend the query vector q from n to $(n+1)$ dimension and set to any random number t
   $q[j] = t$
4. Split the query a vector q in to two vectors according to the splitting indicator S in the symmetric key K[1]
   If $(K[1][j] == 1)$ then
   $q'[j] = q''[j] = q[j]$
   Else
   $q'[j] + q''[j] = q[j]$
   Endif
5. Encrypt the query vector q' and q' using invertible matrix $M1$ and $M2$ in symmetric key and return the trapdoor for the query graph Q as T_{Q1}, T_{Q2}.
   $$T'_{Q1} = (K[2])^T . q'$$
   $$T'_{Q2} = (K[3])^T . q''$$
6. compute the trapdoor $T_{Q3} = Sum(K[4][j].q[j]) + t$
7. Return the trapdoor $T_Q = \{T_{Q1}, T_{Q2}, T_{Q3}\}$

In Figure 5, the index storage size of data graph features depends on the number of features in the feature set and the frequency of each feature i.e. minimum support. While varying the minimum support size of the frequent feature set reduces gradually which reduces the index storage size. The index construction cost is depends on the inner product computation of the data graph vector and inverse of invertible matrix M1 in the symmetric key. As the number of data graphs increases in the data set, cost of index set construction is also increases exponentially.

The size of the feature set for different size of feature (number of vertices in the graph) with different minimum support as defined in the Figure 6. As the minimum support increases the number of features in the feature set decreases for different graph size.

Trapdoor generation cost for a query(Q): cost of trapdoor generation for a given query is the sum of the time required for checking the query feature in the frequent feature set of data graph , computation of inner product of query vector and invertible matrix M 2 in the symmetric key and inner product of query vector and computation of sum of the inner product of query vector and $K[4][j]$.

In figure 7, explains the total time required to generate the trapdoor for a different query feature size using fixed data graph size of 20. As the number of query features are increases trapdoor generation time is also increases linearly. Compared to the existing trapdoor generation method PPGQ, our method is optimal in terms of computation time, which reduces the computation overhead of the cloud server.

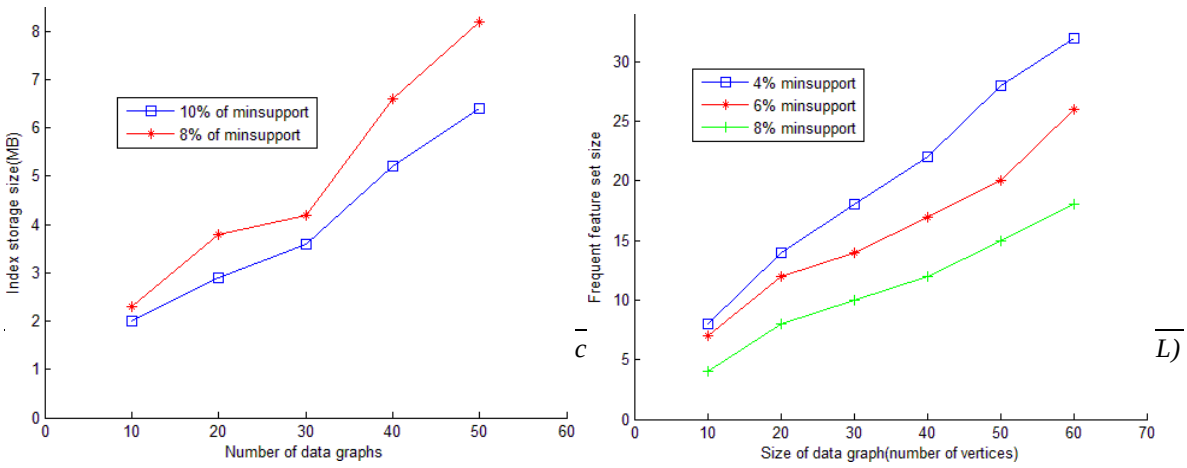
TABLE IV. QUERY PROCESSING IN CLOUD SERVER

Algorithm: Query processing(T_Q, I_{csp}, G_{FQ} **)**

Input: T_Q is a trapdoor for the query graph Q, I_{csp} is the index list for the data graph

Output: G_{FQ} is a list of candidate super graphs which contains a query features F_Q

1. Initialize the candidate super graph list $G_{FQ} = \{\}$
2. For each data graph index list I_{csp}
   Compute the sum of the inner products of index $I_i'.T_{Q1}$ and index $I_i''.T_{Q2}$ i.e T_{Q_csp}
3. If $(T_{Q\_csp} == T_Q)$ then set $G_{FQ} = G_{FQ}\ U\ \{ID(Gi)\}$
4. Return the list of candidate super graphs G_{FQ}



$\overline{c}$ $\overline{L)}$

Fig. 5. Index Storage Size over Data Graph

Fig. 6. Frequent Feature set over Different Data graph Size

Query processing time in cloud server: Query processing cost in cloud server is the time required for the inner product computation of cloud server data graph index (I_{csp}) and trapdoor of the query T_{Q1} and T_{Q2}. If this product is matches with the T_{Q3} sent from the query process [7-9], then cloud server send all the query feature matched encrypted super graphs to the user. In Figure 8, explain the fixed query size of 4 vertices execution time on different data base size(number of graphs). The query execution time is linear to the different data graphs size and our query processing is better than the existing PPGQ method as data base size increases.

In Figure 9, explain the analysis of time required to execute variable query sizes on fixed data base size is set to 40 graphs. Query execution time in cloud server is better than the trapdoor generation of same query which improves the computation overhead of cloud server than PPGP. Graph feature Encryption time

In Figure 10 explain the time required encrypt the data graph vector g and the query vector q of different frequent feature set of data graph. For variable size of frequent feature set, query vector encryption time is better than data vector g encryption time because data vector encryption compute the inner product of inverse of invertible matrix M 1 and data vector g.

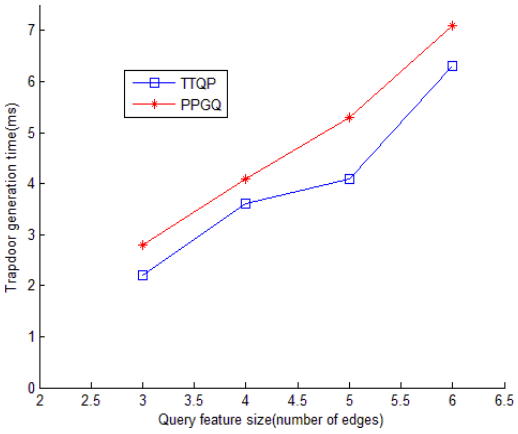


Fig. 7. Trapdoor Generation Time

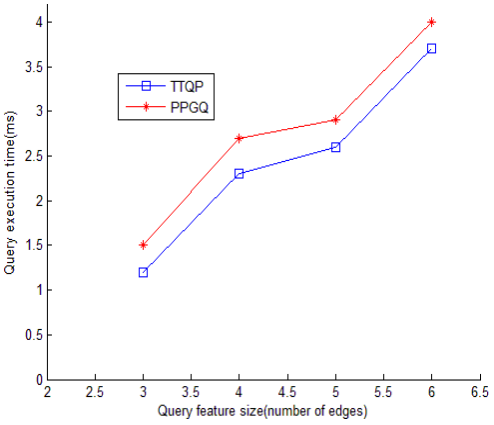


Fig. 9. Query Execution Time over Query Feature Size

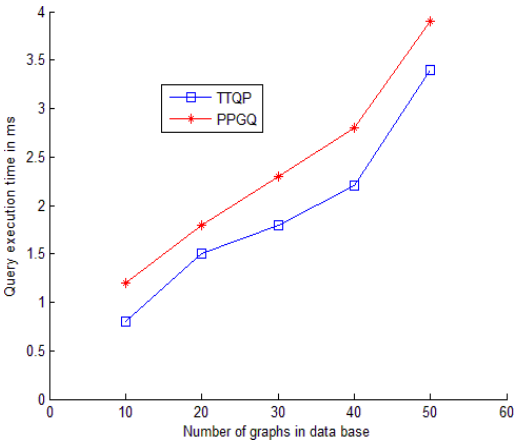


Fig. 8. Query execution Time over Different Data graph

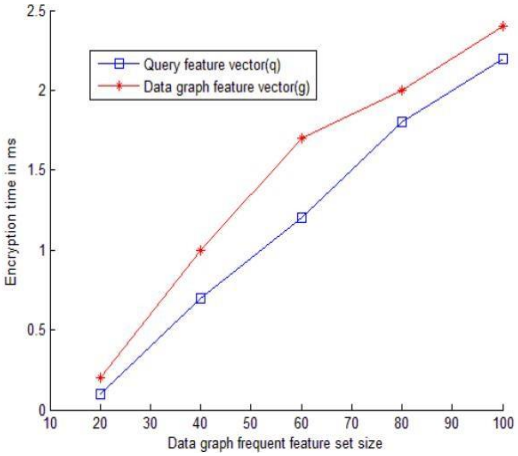


Fig. 10. Graph Feature Encryption Time

6. RELATED WORK

In cloud computing providing confidentiality for the out- sourced personal data is an important research issue. Wenjun Lu, et al. [10] defined as problem of content based search of image data for preserving data confidentiality. They are proposed a solutions such as homomorphic encryption and feature randomization based search for image data privacy. The homomorphic based and feature randomization based search technique is efficient for searching a encrypted data in a list, but it has more computation and communication overheads but feature randomization based search is a deterministic randomization.

The information retrieval using cryptographic method de- fined Song et al. [11] based on text documents and they pro- posed using boolean search operation to identify the matched document. Swaminathn et al. [12] they proposed a search protocol using rank ordered keyword search over the encrypted document. To handle large volume of semistructured XML data a tree search protocol [13]has proposed using storage, search and retrieval phases. In this method encryption time is linear to the input document size and which is depends on the structure of the document and search.

Jin Li et al. [14] defines the problem of fuzzy key- word search over the encrypted cloud data. To construct the fuzzy keyword set they proposed a wild-card and gram-based methods which helps to retrieve the user keyword matched documents from the cloud server. This solution has limitation of privacy of keywords and storage overhead in cloud server.

For query processing E Mykletun and G Tsudik [15], uses the secure co-processor to perform the secure computation and which is slower computational capacity than usual processor. In this device install the encryption and decryption key and deploy the application logic for data privacy.

7. CONCLUSIONS

In this paper, we considered a graph structured data security problem in remotely stored encrypted data in cloud server. To solve this problem we proposed an efficient Trusted party Query Process method using encrypted graph features search index. Using frequent graph feature isomorphism optimizes the storage overheads during query graph searching process. While introducing Trusted Trapdoor Query process between data owner, user and cloud server reduces the computation overhead of cloud server. Through security analysis, we show that, our proposed solution is better graph data security than the existing methods. We also identified to extend this method to solve the nested query features in a larger data set.

REFERENCES

[1] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou, *Privacy- Preserving Multi-keyword Ranked Search over Encrypted Cloud Data*,The 30th IEEE Conference on Computer Communications (INFO-COM'11), Shanghai, China, April 10-15, 2011.

[2] Haichuan Shang Ying Zhang Xuemin Lin, Jeffrey Xu Yu, *Taming Verification Hardness: An Efficient Algorithm for Testing Subgraph Isomorphism*, PVLDB '08, August 23-28, pp 364-375, New Zealand, 2008.

[3] Ning Cao, Zhenyu Yang, Cong Wang, Kui Ren, and Wenjing Lou, *Privacy-preserving Query over Encrypted Graph-Structured Data in Cloud Computing*,IEEE ICDCS2011, Minnesapolis, MN, Jun. 20-24, 2011

[4] Tero Harju, *Lecture Notes on Graph Theory*, Department of Mathemat- ics University of Turku, Finland, 2011.

[5] Xifeng Yany, Philip S. Yuz, Jiawei Hany, *Graph Indexing: A Frequent Structure-based Approach*, SIGMOD-2004, June 13-18, Paris, France, 2004.

[6] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou,*Attribute Based Data Sharing with Attribute Revocation*, The 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS'10), Beijing, China, April 13-16, 2010.

[7] J R Winkler, *Securing the Cloud: Cloud Computing Security Techniques and Tactics*, Elsvier Inc., USA, 2011.

[8] Cong Wang, Bingsheng Zhang, Kui Ren, and Janet M. Roveda, *Privacy- Assured Outsourcing of Image Reconstruction Service in Cloud*, in IEEE Transaction on Emerging Topics in Computing, vol. (1):1, pp 166-177, June 2013,

[9] Mi Wen, Rongxing Lu, Kuan Zhang, Jinsheng Lei, Xiaohui, and Xuemin Shen, *PaRQ: A Privacy-Preserving Range Query Scheme Over Encrypted Metering Data for Smart Grid*, IEEE Transactions on Emerging Topics in Computing, vol (1):1, pp 178-191, June 2013.

[10] Wenjun Lu, Avinash L Varna and Min Wu, *Confidentiality-Preserving Image Search: A Comparative Study Between Homomorphic Encryption and Distance Preserving Randomization,* In IEEE Transactions on Content Mining, Vol. 2, March 2014.

[11] D Song, D Wagner and A Perrig, *Practical Search Techniques for Searches in Encrypted Data*, In IEEE Proceeding of Symp. Res. Sec. Privacy, pp. 44-55, 2000.

[12] A Swaminathan, Y Mao, G M Su, H Gou and A L Varna, *Confiden- tiality Preserving Rank Ordered Search*, in Proceeding ACM Workshop Storage Security Survivability, pp 7-12, 2007.

[13] R Brinkman, L Feng, J Doumen, P H Hartel and W Jonker, *Efficient Tree Search in Encrypted Data*, Infirmation System Security, June, 2004.

[14] Jin Li, at el., *Enabling Efficient Fuzzy Keyword Search over Encrypted Data in Cloud Computing*, 2011.

[15] E Mykletun and G Tsudik, *Incorporating a secure coprocessor in the Database as a Service Model*, In International Conference on IWIA, 2005.