IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Encrypted Graph-Structured Data Privacy-Preserving Query

Mr.Haider Kareem Mohammed
Research Scholar, Department of Computer Science, Madurai Kamaraj University, Madurai - 625 021, Tamilnadu, India

Dr.B.Indrani
Assistant Professor, Department of Computer Science, Director of Distance Education Madurai Kamaraj University, Madurai - 625 021, Tamilnadu, India

Abstract

Datacenters play a critical role as key cloud infrastructure providers such as Amazon, Google, and Microsoft, in the increasingly popular cloud computing. Datacenters provide utility computing to software providers, who then deliver application services to end consumers via the Internet. The latter has long been known as "Software as a Service (SaaS)," while the former has recently been dubbed "Infrastructure as a Service (IaaS)," with the software service provider being known as a cloud service provider.

To take use of cloud infrastructure providers' computing and storage resources, data owners are increasingly outsourcing data to datacenters via cloud service providers, such as the online storage service provider, which are not fully trusted by data owners. The graph has been increasingly used to model complicated structures and schemaless data, such as the personal social network (the social graph), relational databases, XML documents, and chemical compounds studied by research labs as a general data structure to describe the relation between entities. The attributed relational graph (ARG) [20] can likewise be used to model images in a personal album. These sensitive data must be encrypted before being outsourced to the cloud to preserve users' privacy. Furthermore, some information is expected to be shared with trustworthy partners.

Key words: Privacy, Data, Coding, Frame, Document, Query

Introduction

The inner product of the query vector and the data vector could precisely count the number of query features in the data graph, which could be used to filter out negative data graphs that don't include the query graph. Direct outsourcing of the data vector or query vector, on the other hand, will compromise the index or query privacy. We propose a secure inner product computation mechanism, which is adapted from a secure k-nearest neighbor (kNN) technique [120], to meet the challenge of supporting graph semantics

without privacy breaches, and then show how we improved it to meet various privacy requirements under the known-background threat model. The following is a list of our contributions:

1. We investigate the topic of querying encrypted graph-structured data in cloud computing for the first time, and propose a set of stringent privacy requirements for such a safe cloud data usage system to become a reality.
2. For efficiency consideration, our suggested scheme uses the "filtering-and-verification" approach, and a full analysis of the proposed scheme's privacy and efficiency guarantees is provided.
3. The evaluation, which was carried out on the Amazon EC2 cloud infrastructure using the widely published AIDS antiviral screen dataset, revealed that our suggested technique had a minimal compute and communication overhead.

The rest of this chapter is laid out as follows: We introduce the system model, the threat model, and our design goals. Preliminaries are given in part and the framework and privacy standards in PPGQ are described in section followed by our suggested method in section The outcomes of the evaluation are presented in Section In Section we address related work on keyword searchable encryption and graph confinement queries, and in Section, we wrap up the chapter.

The System Model

Consider a cloud data storage service that involves four parties: the data owner, data user, storage service provider/cloud service provider, and datacenter/cloud infrastructure provider. The storage service provider deploys its storage service on top of the utility computing in the datacenter and delivers the service to end users (including data owners and data users) through the Internet to take advantage of the datacenter's utility computing services, such as computing and storage resources. Because neither the cloud service provider nor the cloud infrastructure provider can be fully trusted by data owners or users in our system model, they are considered as one entity, the cloud server.

The data owner has a graph-structured dataset G that will be encrypted and sent to the cloud server. Before data outsourcing, the data owner will create an encrypted searchable index I from G, and then both the index I and the encrypted graph dataset $\tilde{G}$ will be outsourced to the cloud server to provide query capability over $\tilde{G}$ for effective data usage. An authorized user obtains a corresponding trapdoor T_Q for each query graph Q using the search control mechanism, such as broadcast encryption [40], and sends it to the cloud server. The cloud server is responsible for performing a query over the encrypted index *I* and returning the encrypted candidate super graphs after obtaining T_Q from data users. Finally, data users employ the access control mechanism to decrypt the candidate super graphs and validate each one by checking subgraph isomorphism.

The Known Background Threat Model

In our approach, the cloud server is "honest-but-curious," which is consistent with the majority of related publications on searchable encryption [30, 129]. The cloud server, in particular, acts in a "honest" manner and correctly follows the protocol specification. However, inferring and analysing the data and the index in its storage and interactions during the protocol to gain more information is "curious." Because both the encrypted data $\tilde{G}$ and the searchable index I are outsourced and stored on the cloud server, the cloud server can readily acquire both. The cloud server is also expected to know some background information about the dataset, such as its subject and relevant statistical data, in addition to the encrypted data.

PPGQ: Privacy and the Framework

We define the framework for querying encrypted graph-structured data in cloud computing in this part, as well as several severe system-wide privacy requirements for such a safe cloud data consumption system.

The Framework

The query operates with the aid of an index that is outsourced to the cloud server, according to our proposed design. We don't show how the data is encrypted, outsourced, or accessed because this is a separate and distinct issue that has been researched elsewhere [129]. The PPGQ framework is depicted in the diagram below.

1. FSCon (G, σ) The frequent feature set F is generated using the graph dataset G and the minimum support σ as inputs.
2. KeyGen (ξ) Takes a secret ξ as input and returns a symmetric key K as output.
3. Build Index (G, K) takes two inputs: a graph dataset G and a symmetric key K, and returns a searchable index I.
4. TDGen(Q, K) takes a query graph Q and a symmetric key K as inputs and produces a trapdoor T_Q.
5. Query (T_Q, I) yields $G_F Q$, i.e. the candidate super graphs of query graph Q, with the trapdoor T_Q and the searchable index I as inputs.

The data owner runs the first three algorithms, FSCon, Build Index, and Build Index, as pre-processes. As part of the cloud data storage service, the query algorithm is run on the cloud server. The trapdoor generation algorithm TDGen may be run by either the data owner or the data user, depending on various search control techniques. Furthermore, while all users may be able to conduct searches on confidential documents, low-priority data users may be denied access to the document's contents, depending on the application circumstance. It should be noted that neither search control nor access control are covered in this dissertation.

Selecting Common Features

There are three types of characteristics that can be taken from the graph dataset to generate a feature-based index: subpath, subtree, and subgraph. According to the feature comparison in [133], either a subtree-based or a subgraph-based feature set is larger than a subpath-based feature set with the same minimum support, especially when the feature size is between 5 and 20. The size of a feature is measured by its number of vertices $|V(F_i)|$ in order to be consistent with the size of a graph,

Which is $|V(G)|$. The larger feature set on the cloud server will necessitate more index storage, as well as a higher computing cost throughout the query procedure. The pruning power of the subgraph-based index, on the other hand, outperforms the other two options, resulting in the lowest false positive ratio and the smallest candidate super graph set.

Privacy Requirements

Data privacy prevents the cloud server from snooping into outsourced data, as mentioned in the architecture, and can be properly safeguarded by existing access control mechanisms [129]. A characteristic privacy criterion in related efforts on privacy-preserving query, such as searchable encryption [40], is that the server should learn nothing but query results. We examine and establish a set of severe privacy criteria specifically for the PPGQ framework with this general privacy statement. While data privacy protections are requested by default in the literature, the following query privacy requirements are more complex and difficult to meet.

Index Privacy

In terms of index privacy, if the cloud server learns the major structure of a graph, or even the complete topology of a small graph, from the outsourced index, it may learn the major structure of a graph, or even the entire topology of a tiny graph. As a result, the searchable index should be built in such a way that the cloud server is unable to undertake such an association attack.

Feature Privacy

The most significant concern for data consumers is to disguise what they are requesting, i.e. the features indicated by the matching trapdoor, so that their query is not disclosed to outsiders like the cloud server. Although the query features can be protected by a cryptographic trapdoor, the cloud server may perform some statistical analysis on the search results to produce an estimate. Specifically, feature support (i.e., the number of data graphs containing the feature), which is a type of statistical information, is sufficient to reliably identify the feature.

This feature-specific information can be used to reverse-engineer the feature if the cloud server has some background knowledge about the dataset. The distribution of feature support in the AIDS antiviral screen dataset [5], as shown in Fig. 4.2, gives adequate information to identify the most common features in the dataset. This challenge is analogous to [131], where document frequency (the number of papers containing the phrase) is utilized as statistical data to reverse-engineer the term.

Pattern of Access

The access pattern is a succession of query results, each of which is G_{FQ} , and includes the query graph's id list of candidate super graphs. The access pattern is thus denoted as (G_{FQ1} , G_{FQ1} , .) Although a few systems have been proposed in the literature to use the private information retrieval (PIR) technique [56] to hide access patterns (e.g., [23, 29]), our proposed schemes are not designed to protect access patterns due to efficiency considerations. This is due to the fact that any PIR-based technique must "touch" the entire dataset on the server, which is inefficient in a large-scale cloud system.

To this end, the cloud server cannot see the query result of any single feature F_j that is part of the access pattern. As explained above, such a query result $G_{\{F_j\}}$ will directly expose the feature's support, breaking the feature's secrecy. As a result, the single-feature query is not included in our proposed strategies.

PPGQ: Analysis and Proposed Scheme

The data graph Gi is chosen as a candidate super graph of the query graph Q to achieve the filtering goal in the graph query technique if and only if G_i contains all of the frequent features in$Q$. Let λ_i stand for the number of query features in the data graph G_i. The size of the query feature set F_Q should be equal to the size of each candidate supergraph Gi's corresponding λ_i I = $|F_Q|$. We suggest using the efficient inner product calculation for pruning negative data graphs G_j that do not contain the query graph, i.e., $\lambda_j < F_Q$, to find the candidate super graph set.

Every data graph G_i is formalized as a bit vector g_i, with each bit $g_{i[j]}$ determined by determining whether or not G_i contains the frequent feature $F_j \subseteq G_i$, $g_{i[j]}$ is set to 1 if F_j G_i; else, it is set to 0. The query graph Q is represented as a bit vector$q$, with each bit $q_{[j]}$ indicating the presence of the frequent feature F_j in the query feature set F_Q. The inner product of the data vector g_i and the query vector λ_i can then be computed, resulting in $g_i \cdot q$. The data vector g_i and the query vector q should not be exposed to the cloud server in order to maintain tight system-wide privacy.

In this part, we describe how to modify a secure inner product computation mechanism borrowed from the safe Euclidean distance computing technique to be privacy-preserving under the known-background threat scenario.

Issues of privacy in secure inner product computation

The safe Euclidean distance computation technique in the secure kNN scheme [120] cannot be directly used here since the inner product of the data vector and the query vector is preferred to pick candidate super graphs of the query graph. The final inner product result changes to r ($g_i \cdot q$) when the extended dimension associated to the Euclidean distance is removed, as illustrated in Section 3.4.1. Because the new result r ($g_i \cdot q$) may be used to determine the original inner product $g_i \cdot q$, it appears that an efficient and secure inner product computing scheme can be implemented.

The cloud server might investigate the relationship between final inner products in two queries whenever two query graphs had an inclusion relationship. Assume that T_Q and T_Q' are trapdoors for two query graphs QandQ', and that the inclusion relation between their corresponding query feature sets is F_Q

$\subset F_Q'$. The cloud server can extract an estimate of the differential feature's support and further identify this feature with the graph dataset's background knowledge, especially when the differential feature subset contains only one feature, i.e., $|F_Q''| = 1$ where $F_Q'' = F_Q' \backslash F_Q$.

The second query feature set F_Q' includes one additional feature as F_k than the first query feature set F_Q, as shown in Tab. 4.1. For each graph G_i, the cloud server evaluates the expression y_i'/y_i, which is equal to (λ_i'/λ_i) (r'/r), and returns a vast number of possible values. These values, on the other hand, can be divided into two groups. If the graph G_i does not have the feature F_k, i.e., $\lambda_i' = \lambda_i$, the expression evaluation y_i'/y_i is equal to r'/r; otherwise, it is greater than r'/r and may be easily spotted due to its peculiar ratio as $\frac{\lambda_{i+1}}{\lambda_i}$. As a result, the lowest values across the entire dataset imply that associated data graphs lack the feature F_k, while other graphs with higher values do.

The Privacy-Preserving Graph Query Scheme Proposed

When the final inner product y_i is a multiple of λ_i i.e. the number of query features contained in the data graph G_i, the statistical analysis attack demonstrated above works. In order to render the prior statistical analysis approach infeasible, we should break such scale relationships.

The data vector and the query vector will be converted from a bit structure to a more sophisticated form in our proposed architecture. If the frequent feature F_j appears in the data graph g_i, the corresponding element $g_{i[j]}$ in the data vector gi is set to $\rho_{[j]}$ instead of 1 where is an n-dimensional vector; otherwise, $g_{i[j]}$ is set to $X_{[i][j]}$ where X is a $n \times n$ matrix and $X_{[i][j]}$ is a random number smaller than $\rho_{[j]}$.

Effectiveness

Assume Q is made up of l query features, with $\ell = |F_Q|$. All the l features in F_Q that are taken from Q are included in any supergraph G_i of the query graph Q. As a result, all of the ℓ corresponding elements in the data vector are equal to the corresponding elements in the, i.e., $g_{i[j_k]} = \rho_{[j_k]}$, where $1 \leq k \leq \ell$ Furthermore, each relevant member in the query vector labelled $q_{i[j_k]}$ is set as r_{j_k}, while all other elements are set to 0.

For any supergraph Gi, the final inner product $g_i \cdot q + t$ is equivalent to $\rho_{[j_k]} r_{j_k} + t$, which is also the outcome of $\rho \cdot q + t$. The latter, $\rho \cdot q + t$, is included in the trapdoor and acts as a selection indicator for potential supergraphs. As every precise supergraph in $G_{\{Q\}}$ produces the same inner product as $\rho \cdot q + t$ with the query vector, our technique does not introduce any false negatives into the return $G_F Q$. However, data graphs that do not contain the query graph Q but contain all of the properties in F_Q may introduce fals positive supergraphs into $G_F Q$.

Efficiency

The query response is well displayed to the data user since the cloud server can efficiently compute the final inner product for every data graph using two multiplications of (n + 1)-dimensional vectors. During the inquiry, the entire inner product computation is O. (mn). Although some expensive operations, such as

graph sequentialization, are involved in FSCon and BuildIndex, they are unavoidable for generating a graph index. And, more crucially, they are only used once during the entire plan.

Privacy

Our approach may construct two completely different trapdoors for the same query graph Q using the unpredictability generated by the splitting procedure and the random values r_j and t. The trapdoor generation's nondeterministic characteristic can ensure trapdoor unlinkability.

As previously stated, the statistical analysis attack works in the safe inner product computation technique because the final inner product y_i has a scale connection with λ_i. And because y_i is a multiple of the initial inner product $g_i \cdot q$, which is equal to λ_i this scale relationship exists. To break the equivalence relationship between $g_i \cdot q$ and λ_i our suggested technique introduces randomization in both gi and q. As a result, the value of $g_i \cdot q$ is not entirely dependent on λ_i. It is nevertheless feasible that $g_i \cdot q \geq g_j \cdot q$ in the case where data graph G_i has fewer query features than data graph G_j.

In addition, the extended dimension t is used to break the direct scale link between y_i and $g_i \cdot q$, removing the indirect scale relationship between y_i and λ_i. As a result, the cloud server is unable to calculate the special ratio $\frac{\lambda_{i+1}}{\lambda_i}$, which is needed to detect the inclusion relationship between two query feature sets, as explained in section 4.5.1. The cloud server cannot compute the support of a single feature without disclosing such an inclusion connection. In other words, statistical analysis cannot compromise feature privacy, and the suggested approach satisfies all of the predicted privacy requirements in section

Conclusion

We look at the problem of privacy-preserving queries in cloud computing over encrypted graph-structured data. The notion of "filtering-and-verification" is applied to our work. To give feature-related information about each encrypted data graph, we pre-build a feature-based index, and then use the efficient inner product as the pruning tool to carry out the filtering method. We propose a safe inner product computing technique and then refine it to meet various privacy requirements under the known-background threat model to address the difficulty of supporting graph query without privacy breaches.

We also intend to look into additional security and privacy issues related to the untrusted cloud server concept. In actuality, cloud servers may occasionally deviate from the established background model. This can happen for a variety of reasons, including software faults, internal/external attacks, or because the cloud server intends to do so in order to save money when serving a huge number of search requests.

References

Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Ensuring data storage security in cloud computing. In *Proc. of IWQoS*, 2009.

Dawn Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proc. of S&P*, 2000.

Elaine Shi. Evaluating predicates over encrypted data. In *CMU-CS-08-166, PhD thesis*, 2008.

Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. In *Proc. of ACNS*, pages 31–45, 2004.