Structure-Preserving Subgraph Query Services (Extended Abstract)

Zhe Fan[†§], Byron Choi[†], Qian Chen[†], Jianliang Xu[†], Haibo Hu[†] and Sourav S Bhowmick[‡]
[†]Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong SAR, China.
Email: {zfan, bchoi, qchen, xujl, haibo}@comp.hkbu.edu.hk
[§]Research and Standard Department, Huawei Software Technologies CO. LTD.
Email: fanzhe@huawei.com
[‡]School of Computer Engineering, Nanyang Technological University, Singapore.
Email: assourav@ntu.edu.sg

I. Introduction

Subgraph query (via subgraph isomorphism) is a fundamental and powerful query in various real graph applications. It has actively been investigated for performance enhancements recently. However, due to the high complexity of subgraph query, hosting efficient subgraph query services has been a technically challenging task, because the *owners* of graph data may not always possess the IT expertise to offer such services and hence may *outsource* to *query service providers* ($\mathcal{SP}$). $\mathcal{SP}$s are often equipped with high performance computing utilities (*e.g.*, a cloud) that offer better scalability, elasticity and IT management. Unfortunately, as $\mathcal{SP}$s may not always be trusted, security (such as the confidentiality of messages exchanged) has been recognized as one of the critical attributes of *Quality of Services* (*QoS*) [4]. This influences the willingness of both data owners and query clients to use $\mathcal{SP}$'s services. Recently, there is a bloom on the research on query processing with privacy preservation[1], e.g., in the context of relational databases, spatial databases and graph databases. However, up to date, private subgraph query has not yet been studied.

Motivating example: Consider a pharmaceutical company with revenue that depends mostly on the invention of health care products. The company may have discovered new compounds for a new product. To save laboratory work, it may query the compounds from proprietary biological pathway networks to check whether it is possible for the ingredient compounds to form other compounds via certain chemical reactions (a structural pattern from the network). On the one hand, the company is reluctant to expose the queries (the ingredients) to the $\mathcal{SP}$, as it may apply for patents for the synthesis. On the other hand, the owner of the pathway networks may not only lack the expertise to host query services but may also be reluctant to release the networks to the public. The owner is willing to release it to paid users only. It is crucial to protect *both* the queries and the network from the $\mathcal{SP}$.

This paper studies that the client may prefer not to expose the structure of query graphs to the $\mathcal{SP}$, and the data owner may not want the $\mathcal{SP}$ to be able to infer the structure of their graph data. The problem is elaborated as follows.

System model. We follow the system model that has been well received in the literature of database outsourcing (shown in Fig. 1), and known to be suitable for many applications. It consists of three parties:

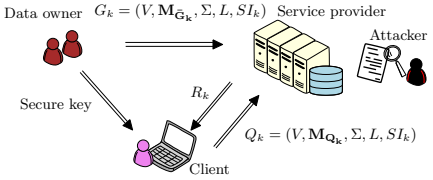


Fig. 1. Overview of the system model.

(1) *Data owner*: He/she owns a database of *voluminous data graphs of modest sizes*. He/she encrypts each graph G and then outsources the encrypted graph to the service provider, and delivers the secret keys to clients for encryption of the query graphs and decryption of the encrypted result;
(2) *Service provider* ($\mathcal{SP}$): The $\mathcal{SP}$ may be equipped with powerful computing utilities such as a cloud. The $\mathcal{SP}$ evaluates a client's encrypted query over the encrypted data, on behalf of the data owner, and returns the encrypted result to the client; and
(3) *Client*: A client encrypts the query graph Q using the secret keys, submits it to the $\mathcal{SP}$, and decrypts the returned encrypted result to obtain the final answer.

Attack model. We assume the dominating semi-honest adversary model from literature, where the attackers are *honest-but-curious* and the $\mathcal{SP}$ may also be the attacker. For presentation simplicity, we often *term the attackers as the* $\mathcal{SP}$. We assume that the attackers are the *eavesdroppers* and adopt the *chosen plaintext attack*. We assume that the $\mathcal{SP}$ and clients are not allowed to collude.

Privacy target. We assume that the privacy target is to protect the *structures* of a query graph Q and a graph data G from the $\mathcal{SP}$ under the attack model defined above. The *structural information* of Q and G considered is the adjacency matrices of Q and G, respectively. More specifically, the probability that the $\mathcal{SP}$ correctly determines the values of the adjacency matrix of the graph is guaranteed to be lower than a threshold with reference to that of *random guess*.

The *problem statement* of this paper can be stated as follows: *Given the above system and attack model, we seek an efficient approach to facilitate the subgraph isomorphism query services with preserving the above defined privacy target.*

[1]In addition to privacy protection via legal means, this stream of research has aimed to offer technological solutions for such protection.

To our knowledge, such a problem has never been addressed before. The intrinsic difficulty of this research is that the $\mathcal{SP}$ cannot optimize query processing by directly using the structures of the graph, since such information cannot be exposed. However, most of the existing subgraph isomorphism algorithms (*e.g.*, VF2 [1], QuickSI [5] and Turbo$_{\text{iso}}$ [3]) for the query services must *traverse* the graph, which by definition leaks structural information. A naïve method is to transfer the entire database to the client for query processing. However, it is inefficient when the database is large.

II. Overview of Our Proposed Solutions

Our techniques for a *structure-preserving* subIso (denoted as SPsubIso) [2] are derived from the Ullmann's algorithm [6], a seminal algorithm for subgraph isomorphism. We revise the Ullmann's algorithm into *three steps* that form the foundation of our techniques. (1) Enum enumerates all *possible subgraph isomorphism mappings* $\mathbf{M}_i$s from query graph Q to data graph G; (2) Match verifies if the mapping $\mathbf{M}_i$ is *valid* or not; and (3) Refine reduces the search space of $\mathbf{M}_i$s by degree and neighborhood constraints. The benefits of adopting the Ullmann's algorithm are twofold: (1) the query evaluation between Q and G is mostly a series of matrix operations between their adjacency matrices $\mathbf{M_Q}$ and $\mathbf{M_G}$. It does not require traversals on structures; and (2) its evaluation requires simple structures. This makes the privacy analysis simpler.

We transform the above three steps into a series of matrix computations, denoted as TsubIso. TsubIso comprises three steps: (1) TEnum enumerates all $\mathbf{M}_i$s; (2) TMatch verifies the validity of $\mathbf{M}_i$ by *additions and multiplications* using $\mathbf{M_Q}$ and $\mathbf{M}_{\bar{\mathbf{G}}}$, where $\mathbf{M}_{\bar{\mathbf{G}}}$ is the complement of $\mathbf{M_G}$; and (3) TRefine reduces the search space of $\mathbf{M}_i$s by *inner product*s on our proposed *static indexes* $\mathsf{SI_Q}$ and $\mathsf{SI_G}$ of Q and G, where $\mathsf{SI_Q}$ ($\mathsf{SI_G}$) is an ensemble of *h-hop information* of each vertex of Q ($\mathsf{SI_G}$) represented by a *bit vector*.

SPsubIso is then proposed on top of TsubIso. The overview of our techniques is presented in Fig. 2. We first propose a new *private-key encryption scheme*, namely *cyclic group based encryption scheme* (CGBE). The data owner transforms, indexes, and encrypts a data graph G into $\bar{G}_k$ offline. The encrypted graph $\bar{G}_k$ is outsourced to the $\mathcal{SP}$. The client obtains a secret key from the data owner and encrypt his/her query Q as Q_k. He/she submits the encrypted query Q_k to the $\mathcal{SP}$.

① We develop SPRefine which exploits private inner products on the static indexes ($\mathsf{SI}_{\mathsf{Q}_k}$ and $\mathsf{SI}_{\mathsf{G}_k}$) to derive a refinement that reduces the number of possible mappings $\mathbf{M}$ between $\bar{G}_k$ and Q_k. The static indexes of the graphs were computed and encrypted offline, whereas those of the queries are computed once by the clients online. We analyze the effects of these optimizations on the probabilities that the $\mathcal{SP}$ may correctly determine graph structures. Therefore, the clients may tune the trade-off between refinement performances and privacy requirements.

② We propose SPEnum which optimizes the search for mappings by introducing a *protocol* that involves the client's participation, who informs the $\mathcal{SP}$ useless mappings. This provides $\mathcal{SP}$ some structural information and thus we quantify its effect on privacy.

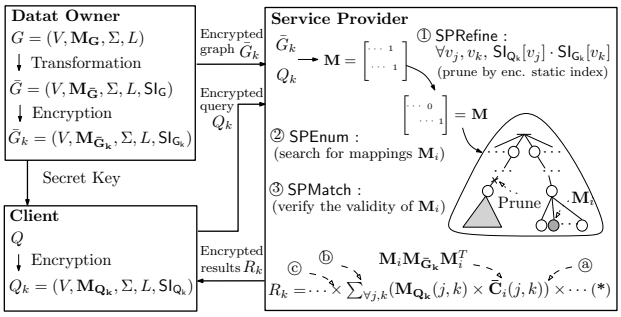
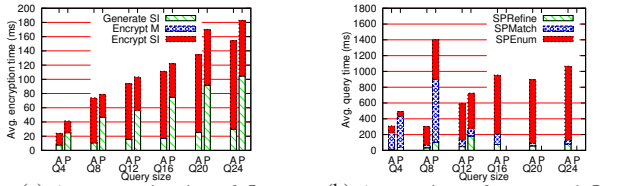


Fig. 2. Overview of our techniques.



(a) Avg. encryption time of Q (b) Avg. runtimes of non-pruned $Q$$n$s

Fig. 3. Performance on real datasets.

③ We propose SPMatch that involves additions and multiplications under CGBE to check the validity of each mapping $\mathbf{M}_i$, with negligible false positives. The computation results under CGBE can be *aggregated* into R_k to reduce communication overheads between the client and the $\mathcal{SP}$. The crucial formula for computing R_k is shown at the bottom of Fig. 2. We prove that CGBE is perfectly secure under *chosen plaintext attack* and the $\mathcal{SP}$ cannot learn any structures from SPMatch.

III. Highlights of Some Experimental Results

We report some results from `Aids` (A) and `PubChem` (P) and their query sets. First, the average encryption times are reported in Fig. 3(a). The encryption time of a query Q involves (1) the time for generating our index $\mathsf{SI_Q}$; (2) the time of encryption of $\mathbf{M_Q}$ by CGBE; and (3) the time of encryption of $\mathsf{SI_Q}$. We observe that the average encryption times are around 100ms and 150ms for `Aids` and `PubChem`, respectively. The encryption of $\mathbf{M_Q}$ by our proposed CGBE is efficient, which only costs several milliseconds on a commodity machine. Second, queries may be pruned by SPRefine at the $\mathcal{SP}$ and their average runtimes were very short. The average query times of non-pruned queries are shown in Fig. 3(b). For most queries, the query times are smaller than 1s. The query time of $Q8$ is the longest but it is still smaller than 1.4s. Third, the decryption time of R_k is tiny.

References

[1] L. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *PAMI, IEEE*, 26(10):1367–1372, 2004.

[2] Z. Fan, B. Choi, Q. Chen, J. Xu, H. Hu, and S. S. Bhowmick. Structure-preserving subgraph query services. *TKDE*, 27(8):2275–2290, 2015.

[3] W.-S. Han, J. Lee, and J.-H. Lee. Turbo$_{iso}$: towards ultrafast and robust subgraph isomorphism search in large graph databases. In *SIGMOD*, 2013.

[4] D. A. Menascé. QoS issues in web services. *IEEE Internet Computing*, 2002.

[5] H. Shang, Y. Zhang, X. Lin, and J. X. Yu. Taming verification hardness: an efficient algo. for testing subgraph isomorphism. In *PVLDB*, 2008.

[6] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23:31–42, 1976.