Privacy-Preserving Reachability Query Services for Sparse Graphs

[1]Peipei Yi [1]Zhe Fan
[1]Hong Kong Baptist University
csppyi,zfan@comp.hkbu.edu.hk

[2]Shuxiang Yin
[2]Fudan University
sxyin@fudan.edu.cn

***Abstract*—This paper studies privacy-preserving query services for reachability queries under the paradigm of data outsourcing. Specifically, graph data have been outsourced to a third-party service provider ($\mathcal{SP}$), query clients submit their queries to the $\mathcal{SP}$, and the $\mathcal{SP}$ returns the query answers. However, $\mathcal{SP}$ may not always be trustworthy. Therefore, this paper considers protecting the structural information of the graph data and the query answers from the $\mathcal{SP}$. This paper proposes *simple yet optimized* privacy-preserving 2-hop labeling. In particular, this paper proposes that the encrypted intermediate results of encrypted query evaluation are indistinguishable. The proposed technique is secure under *chosen plaintext attack*. We perform an experimental study on the effectiveness of the proposed techniques on both real-world and synthetic datasets.**

I. Introduction

Graphs have been found in many emerging applications including bioinformatics analysis, communication networks, social networks, knowledge networks and semi-structured databases. Due to the massive volume of graph data from such a wide range of recent applications and the IT resources required to evaluate numerous queries at large scale, it is becoming economically appealing to outsource graph data to a third-party service provider ($\mathcal{SP}$) who provides query services.

Example. Suppose the police are at an early stage of identifying a terrorism suspect who is studying in a university. The police may issue numerous queries – "*who are reachable by the suspect?*" – on the communication network (*e.g.*, email or phone communications) owned by the university. While the university is required by law to cooperate with the police, it does not prefer its daily operations affected by the queries. Hence, when needed, it may outsource its network to an $\mathcal{SP}$ for processing such queries. At the meantime, the network should be protected from the $\mathcal{SP}$ as it contains sensitive information. Importantly, the police do not prefer to expose both their queries and answers to the $\mathcal{SP}$ which in turn expose their investigations. Therefore, it is imperative to propose a scheme for privacy-preserving query services.

In this paper, we consider privacy-preserving *reachability* query on *sparse graphs*, as the reachability query is one of the most popular and fundamental queries and real graphs are often sparse [1]. In our technical report [13], we realize that there have been a large body of indexes on reachabaility queries, *e.g.*, [2], [4], [5], [12], we undertake a 2-hop approach to address this problem as the main benefits of the 2-hop are threefold. First, the 2-hop labels are simple: each vertex is only associated with two sets of vertices. Second, the query evaluation of 2-hop labeling is simply an intersection between two sets. Such simple structure and algorithm make the analysis of privacy simple as well. Third, the large body of works for 2-hop labeling (*e.g.*, [2], [4], [5]) can be readily adopted.

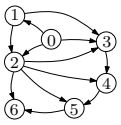
In 2-hop labeling, the intermediate query results of a pair of unreachable vertices and reachable vertices are an empty set and a non-empty set, respectively. To make the intermediate results of reachable/unreachable query vertices indistinguishable to provide privacy, [13] proposes privacy-preserving 2-hop (called pp-2-hop) and a heuristic to enlarge 2-hop labeling such that those result sizes always equal to a constant $I_{\texttt{max}}$. In particular, this involves modifying the results of unreachable query vertices from an empty set to a non-empty one. In sparse graphs, *most vertices of sparse graphs are not reachable from each other*. (From our experiment shown in Table II, the number of reachable pairs of vertices is only around 9% pair of the number of all possible pairs.) The value of $I_{\texttt{max}}$ that leads to the smallest privacy-preserving 2-hop is obviously 1. This paper therefore proposes 2-hop labeling where the sizes of its intermediate results during query evaluation of any pair of query vertices are always 1.

II. Problem Formulation and Overview

Data model. We consider *directed node-labeled graphs*. A graph is denoted as G, and $V(G)$ and $E(G)$ are the vertex and edge sets of G, respectively. Since the reachability information of vertices in a strongly connected component is identical, we assume directed acyclic graphs (DAG). A reachability query takes two vertices u and v as input, denoted as $\texttt{Reach}(u, v)$, and returns true *iff* v is reachable from u, *i.e.*, $u \rightsquigarrow v$.

System model. In the literature of database outsourcing, query services often involve three parties:

- *Data owner*: An owner owns the graph and precomputes indexes *offline once*. It outsources (the encrypted version of) them to an $\mathcal{SP}$, and sends query clients secret keys to encrypt queries and decrypt query results; and
- *Service provider* ($\mathcal{SP}$): An $\mathcal{SP}$ is often equipped with powerful computing utilities such as a cloud. The $\mathcal{SP}$ processes massive query requests on encrypted data and returns the encrypted results to clients; and
- *Client*: A client encrypts his/her query, sends it to the $\mathcal{SP}$ and decrypts the result from the $\mathcal{SP}$. We assume that the clients and $\mathcal{SP}$ do not collude.



v	$\texttt{Lin}(v)$	$\texttt{Lout}(v)$
0	0	0, 1, 2, 3, 5, 6
1	1	1, 2, 3, 5, 6
2	2	2, 3, 5, 6
3	3	3, 4
4	0, 1, 2, 4	4
5	4, 5	5, 6
6	4, 6	6

Fig. 1. A small communication network (LHS) and its 2-hop labeling (RHS)

Attack model and privacy target. The attackers may be the $\mathcal{SP}$ or another adversary hacking the $\mathcal{SP}$. For simplicity, we *term the attackers as the* $\mathcal{SP}$. We assume the $\mathcal{SP}$s are *honest-but-curious*. We assume that the $\mathcal{SP}$ adopts the *chosen plaintext attack* (CPA). Our *privacy target* is required to keep the following two pieces of information private.

- *Reachability of the query vertices* a.k.a the query result. In particular, given a reachability query $\mathsf{Reach}(u, v)$, the $\mathcal{SP}$ cannot infer whether $u \rightsquigarrow v$; and
- *Graph structure* a.k.a. the topology of the data graph, *e.g.*, the existence of an edge.

III. Background of Privacy-Preserving 2-Hop

2-hop labeling. We first briefly summarize 2-hop labeling, which forms the basis of our approach. In 2-hop labeling, each vertex $u \in V(G)$ is associated with two sets of vertices, denoted as $\texttt{Lout}(u)$ and $\texttt{Lin}(u)$, called 2-hop *labels*. Vertices in $\texttt{Lout}(u)$ (resp., $\texttt{Lin}(u)$) can be reachable from u (resp., can reach u), which are also called *center nodes*. Given two vertices u and v, $u \rightsquigarrow v$ *iff* $\texttt{Lout}(u) \cap \texttt{Lin}(v) \neq \emptyset$.

Example 3.1: Consider the simplified communication network shown in the LHS of Fig. 1. The vertex ID represents the user ID and the edge denotes a message from one user to another. A possible 2-hop labeling of the graph is shown in the RHS of Fig. 1. Consider two vertices 1 and 5. Vertex 1 can reach Vertex 5, *i.e.*, $1 \rightsquigarrow 5$. $\texttt{Lout}(1) \cap \texttt{Lin}(5) = \{5\}$. Vertex 0 is not reachable from Vertex 6 as $\texttt{Lout}(6) \cap \texttt{Lin}(0) = \emptyset$.

Privacy-preserving 2-hop labeling (pp-2-hop) [13]. The majority of existing works on 2-hop labeling focus on minimizing the sizes of 2-hop labels. In comparison, as motivated in Sec. I, pp-2-hop enlarges the 2-hop labels such that the intermediate results of any queries indistinguishable, to provide privacy preservation. Specifically, the intermediate result is the intersection of 2-hop labels: $\texttt{Lout}(u) \cap \texttt{Lin}(v)$, where $u, v \in V(G)$. In [13], we introduce *surrogate nodes* into 2-hop labels (Lins and Louts) such that that the intersection sizes between the 2-hop labels with surrogate nodes $\texttt{Lout}^{\texttt{s}}(u)$ and $\texttt{Lin}^{\texttt{s}}(v)$ (called surrogate labels) always equal to a constant $I_{\texttt{max}}$. Surrogate nodes are indicated by a flag, defined as follows.

Definition 3.1: Each *center node* of $\texttt{Lout}^{\texttt{s}}(u)$ or $\texttt{Lin}^{\texttt{s}}(v)$ is a binary tuple (w,f), where f = true if w is a real center, and a fake center, otherwise. ■

A heuristic algorithm is proposed to minimize the value of $I_{\texttt{max}}$, and construct the pp-2-hop labels to meet the requirement that $|\texttt{Lout}(u) \cap \texttt{Lin}(v)| = I_{\texttt{max}}$, for all u and v in $V(G)$.

Definition 3.2: The *privacy preserving* 2-hop (pp-2-hop) [13] is a 2-hop labeling where each encrypted vertex u_e, where $u_e = h_{s_2}(u)$ and h_{s_2} denotes a one-way collision-resistant with a salt s_2, is associated with two encrypted surrogate labels

v	$\texttt{Lin}^{\texttt{s}}(v)$	$\texttt{Lout}^{\texttt{s}}(v)$
0	**0**, 7, 8, 9	**0**, **1**, **2**, **3**, **5**, **6**, 7, 8
1	**1**, 7, 8, 10	**1**, **2**, **3**, **5**, **6**, 7, 8, 9
2	**2**, 7, 8, 11	**2**, **3**, **5**, **6**, 7, 8, 9, 10
3	**3**, 7, 8, 12	**3**, **4**, 7, 8, 9, 10, 11
4	**0**, **1**, **2**, **4**, 9, 10, 13	**4**, 7, 8, 9, 10, 11, 12
5	**4**, **5**, 7, 8, 14	**5**, **6**, 7, 8, 9, 10, 11, 12, 13
6	**4**, **6**, 7, 8	**6**, 7, 8, 9, 10, 11, 12, 13, 14

Fig. 2. The 2-hop labels of the network in Fig. 1 with surrogate nodes [13]

v	$\texttt{Lin}(v)$	$\texttt{Lout}(v)$
0	**3**, 7, 8, 10	**0**, **1**, **3**
1	**1**, 7, 8	**0**, **1**, 10
2	**1**, **4**, 7, 11	**0**, **4**, 8
3	**0**, 7	**0**, 8, 11
4	**0**, **5**, 9, 12	**2**, **5**, 7
5	**0**, **2**, 9	**2**, 7, 12
6	**0**, **2**, **6**	**6**, 7, 9

Fig. 3. The m-2-hop labels of the network in Fig. 1

$\texttt{Lout}^{\texttt{s}}(u)$ and $\texttt{Lin}^{\texttt{s}}(u)$, denoted as $\texttt{Lout}^{\texttt{e}}(u)$ and $\texttt{Lin}^{\texttt{e}}(u)$, and (w, f) in the surrogate labels is encrypted as $(h_{s_1}(w), \mathsf{Enc}(f))$, and $\mathsf{Enc}(\cdot)$ is the Elgamal encryption scheme. ■

Example 3.2: Fig. 2 shows a possible pp-2-hop labeling, after adding surrogate nodes. We do not encrypt the labels, for discussion's sake. The bold text indicates the real center nodes. $I_{\texttt{max}}$ is 3. $\texttt{Lin}^{\texttt{s}}(4) \cap \texttt{Lout}^{\texttt{s}}(0)$ is {**0**,**1**,**2**} and $0 \rightsquigarrow 4$. However, $\texttt{Lin}^{\texttt{s}}(4) \cap \texttt{Lout}(5)$ equals {9,10,13} and $5 \not\rightsquigarrow 4$. Recall from the 2-hop of Fig. 1 that $\texttt{Lin}(4) \cap \texttt{Lout}(5)$ is $\emptyset$. The sizes of 2-hop labels in pp-2-hop may notably increase for sparse graphs as most vertices are not reachable from each other.

IV. Minimum Unified Intersection 2-hop (m-2-hop)

In this section, we define m-2-hop and present its construction. We show that the construction can be solved by adopting algorithms for the minimum set cover problem.

Definition 4.1: The *minimum unified intersection* 2-hop (m-2-hop) is a 2-hop labeling of a graph G, denoted as Lin and Lout (defined as Def. 3.1), where

- $\forall u, v \in V(G)$, $\texttt{Lin}(v) \cap \texttt{Lout}(u) = \{(w, \texttt{true})\}$ if $u \rightsquigarrow v$, and $\{(w', \texttt{false})\}$ otherwise;
- $\sum_{u \in V(G)}(|\texttt{Lin}(u)| + |\texttt{Lout}(u)|)$ is minimized. ■

Def. 4.1 implies that the query processing of m-2-hop is simply an intersection and its result is always a singleton. Given a reachability query $\mathsf{Reach}(u, v)$, $\texttt{Lin}(v) \cap \texttt{Lout}(u) = (w, \texttt{true})$ indicates that $u \rightsquigarrow v$, and $(w', \texttt{false})$, otherwise.

Example 4.1: Following Fig. 2, we show a possible m-2-hop of the graph (in Fig. 1) in Fig. 3. $I_{\texttt{max}}$ is 1. $\texttt{Lin}(4) \cap \texttt{Lout}(0)$ is {**0**} and $0 \rightsquigarrow 4$. $\texttt{Lin}(4) \cap \texttt{Lout}(5)$ equals {12} and $5 \not\rightsquigarrow 4$.

A. Analysis of MUI-2-Hop

To construct m-2-hop, two types of surrogate nodes are required by Def. 4.1. (1) *Real* nodes are to *cover* those $u \rightsquigarrow v$; and (2) *Fake* nodes are to *cover* those $u \not\rightsquigarrow v$. It is apparent that the problem of adding real nodes to cover all $u \rightsquigarrow v$ is *equivalent* to that of fake nodes and these problems can be considered independently. Then, the problem of constructing m-2-hop becomes *adding minimum real (resp. fake) nodes* to *cover* all $u \rightsquigarrow v$ (resp. $u \not\rightsquigarrow v$), *s.t.*, $|\texttt{Lin}(v) \cap \texttt{Lout}(u)| = 1$.

Proposition 4.1: The problem of constructing the m-2-hop labels of a graph is NP-hard. ■

The hardness is established by a simple reduction of the *minimum set cover problem* (MSC).

We then illustrate that the construction of m-2-hop is closely related to the MSC problem. Consider the reachable nodes of a graph G. We model the transitive closure $TC(G)$ as a bipartite graph $B = (V, L, R, E)$, where $L(B) = R(B) = V(G)$. Intuitively, $L(B)$ and $R(B)$ represent Lins and Louts, respectively. $\forall u \in R(B)$, $v \in L(B)$, $(u, v) \in E(B)$ if $u \rightsquigarrow v$.

We consider the universe of the MSC instance is $\mathcal{U} = E(B)$, which are all entries of $TC(G)$. A set of subsets of $\mathcal{U}$ is denoted as $\mathcal{S}$, where $\mathcal{S} = \{S | S = E(K)$, K is a subgraph of B, and $\forall u \in R(K), v \in L(K), (u, v) \in E(B)\}$. That is, K represents the entries of $TC(G)$, covered by $E(K)$. Further, we require to use *bicliques* (complete bipartite graphs) to cover $E(B)$, to ensure all edges $E(B)$ will be covered once. The weight attached to each $S \in \mathcal{S}$ is $\texttt{weight(S)} = |L(K)| + |R(K)| = |V(K)|$, as $E(K)$ is covered by adding a center node to each Lin of $L(K)$ and each Lout of $R(K)$.

The objective of the MSC problem is to find a collection $\mathcal{S}' \subset \mathcal{S}$ such that (1) $\mathcal{U}$ is completely covered by $\mathcal{S}'$, and correspondingly, each edge in $E(B)$ is covered. For minimum cover, an edge in $E(B)$ is covered once, which means $|\texttt{Lout}(u) \cap \texttt{Lin}(v)| = 1$ if $u \rightsquigarrow v$; and (2) $\sum_{S \in \mathcal{S}'} \texttt{weight(S)}$ is minimized, which implies $\sum_{u \in V(G)} (|\texttt{Lin}(u)| + |\texttt{Lout}(u)|)$ is minimized. It is obvious that finding such an $\mathcal{S}'$ is exactly finding the m-2-hop cover with the minimum size.

A MSC-based solution. To address this problem, we apply the classical greedy algorithm of MSC, whose approximation ratio is (1+1/2+$\cdots$+1/$|\mathcal{U}|$)OPT. Initially, $\mathcal{U}'$ is defined to represent the uncovered elements of $\mathcal{U}$, *i.e.*, $\mathcal{U}' = \mathcal{U} = E(B)$. A set S of $\mathcal{S}$ with the maximum value of $\frac{|S \cap \mathcal{U}'|}{w(S)}$ is iteratively chosen to cover $\mathcal{U}'$ and removed from $\mathcal{U}'$. It terminates until all elements in $\mathcal{U}'$ are covered. Finding such a set S is equivalent to finding the *maximum biclique* K from B due to the following.

(1) $\forall u \in R(K), v \in L(K), (u, v) \in E(B)$; and
(2) $\frac{|S \cap \mathcal{U}'|}{\texttt{weight(S)}} = \frac{|E(K) \cap \mathcal{U}'|}{|V(K)|} = \frac{|E(K)|}{|V(K)|}$ is the maximum.

$|S \cap \mathcal{U}'| = |E(K) \cap \mathcal{U}'|$ as $S = E(K)$ by definition. $|E(K) \cap \mathcal{U}'| = |E(K)|$. This is to determine the densest subgraph. Since the MSC construction requires to use bicliques to cover $E(B)$, it determines the densest biclique. Note that finding the maximum biclique of a given bipartite graph is also NP-hard.

B. Heuristic MUI-2-Hop Construction

Next, we propose a greedy algorithm for finding maximal bicliques incorporated into the MCS-based construction of m-2-hop(presented in Algo. 1). The input is the data graph G. The outputs are the Lin and Lout labels. Algo. 1 first initializes Lins and Louts to empty sets and generates the transitive closures $TC(G)$ and its complement $TC(G)^-$ (Lines 1-2).[1] Then, bipartite graphs B and B^- are created from $TC(G)$ and $TC(G)^-$, respectively. AddSurNode is invoked on B (resp. B^-) for adding real (resp. fake) nodes in Lin and Lout with its flag set to true (resp. false) (Lines 3-4). Lastly, Lins and Louts are returned (Line 5).

[1]We remark that there are existing works on constructing 2-hop labels without generating the TC in advance, *e.g.*, [4]. For presentation simplicity, we discuss our algorithm with TC.

Algorithm 1 MUIS (G)

Input: A graph G.
Output: The m-2-hop label of G: Lin and Lout.
1: Initialize $\forall u \in V(G)$, $\texttt{Lin}(u) = \texttt{Lout}(u) = \emptyset$
2: Generate $TC(G)$ and $TC(G)^-$
3: Generate B of $TC(G)$, AddSurNode (B, Lin, Lout, true)
4: Generate B^- of $TC(G)^-$, AddSurNode (B^-,Lin,Lout,false)
5: **return** Lin and Lout

Procedure 1.1 AddSurNode (B, Lin, Lout, f)
6: **while** $E(B) \neq \emptyset$ /* edges not yet covered. */
7: $K \leftarrow$ GreedyFndMaxBiK (B)
8: $\forall (u, v) \in E(K)$, add a *new* surrogate node w, *i.e.*,
$\texttt{Lin}(v) \leftarrow \texttt{Lin}(v) \cup \{(w, f)\}$,
$\texttt{Lout}(u) \leftarrow \texttt{Lout}(u) \cup \{(w, f)\}$
9: $E(B) \leftarrow E(B)/E(K)$ /* remove edges in K from B*/

Procedure 1.2 GreedyFndMaxBiK (B)
10: Initialize an empty biclique K,
11: **while** $T \neq \emptyset$, where T=$\{u | \{u\} \cup V(K)$ forms a biclique in $B\}$
12: Let $u_{\max} = \arg\max_{u \in T}(Deg(u))$
13: $V(K) \leftarrow V(K) \cup \{u_{\max}\}$
14: **return** K

AddSurNode (Lines 6-9) is the classical greedy algorithm for MSC, as discussed, written in terms of bipartite graphs. For each iteration, if $E(B)$ is not yet completely covered, *i.e.*, $E(B) \neq \emptyset$ (Line 6), a *maximal biclique* K of B will be chosen by GreedyFndMaxBiK (Line 7). One new surrogate node w is created and added into $\texttt{Lin}(v)$ and $\texttt{Lout}(u)$ with the corresponding flag value, where $(u, v) \in E(K)$ (Line 8). Then, $E(K)$ is removed from $E(B)$ (Line 9).

GreedyFndMaxBiK (Lines 10-14) finds a maximal biclique K of B. The algorithm first initializes an empty K (Line 10). If it can find a set of nodes T, *s.t.*, each of the node $u \in T$ can form a biclique with K (Line 11), a node $u_{\max}$ with the largest degree is then selected from T (Line 12). $u_{\max}$ is added into $V(K)$ to enlarge the biclique K (Line 13). GreedyFndMaxBiK terminates until such T cannot be found and returns the biclique K (Line 14). Such a K is the maximal one. The benefit of GreedyFndMaxBiK is that its performance guarantee is immediate: Finding the *maximum biclique* K in B is equivalent to finding the *maximum independent set* M in B^-, where B^- is a complement bipartite graph of B. It is known that such a greedy algorithm for outputting the maximum independent set is a $1/(1 + \Delta)$-approximation.

V. Privacy-preserving 2-hop Query Processing

In this section, we define the encryption of m-2-hop and introduce its private query processing. We then give a proof on the privacy of m-2-hop.

Definition 5.1: A *privacy-preserving* m-2-hop (ppm-2-hop) a graph G is an encrypted m-2-hop of G, denoted as $\texttt{Lin}^{\texttt{e}}$ and $\texttt{Lout}^{\texttt{e}}$. A center (w, f) in m-2-hop labels is encrypted as $(h_{s_1}(w), \texttt{Enc}(f))$, where h_{s_1} is a one-way hash function with a salt s_1 and Enc is a randomized encryption scheme. ■

Private query processing. The query processing of ppm-2-hop consists of three main steps: (1) The client hashes the query $\texttt{Reach}(u, v)$ as $\texttt{Reach}^e(u_e, v_e)$, where $u_e = h_{s_2}(u)$ and $v_e = h_{s_2}(v)$, and issues it to the $\mathcal{SP}$; (2) The $\mathcal{SP}$ performs $\texttt{Lin}^{\texttt{e}}(v_e) \cap \texttt{Lout}^{\texttt{e}}(u_e)$ and retrieves *one* encrypted binary tuple (w_e, f_e). It then returns such f_e to the client; and (3) The client decrypts query answer from f_e using the secret key.

Privacy Analysis. The privacy offered by ppm-2-hop under chosen plaintext attack (CPA) can then be established.

Proposition 5.1: The reachability of the query vertices $\mathsf{Reach}^e(u_e, v_e)$ is preserved under CPA. ■

Proof: For each $(w_e,\ f_e) \in \mathtt{Lin}^e(u)$ (or $\mathtt{Lout}^e(u)$), the $\mathcal{SP}$ cannot determine if w_e is *real* or not since (1) different salts are used to hash u and w, *e.g.*, by exploiting SHA; and (2) f_e is encrypted by Enc. Enc can be readily implemented by AES, which is secure under CPA. Therefore, the $\mathcal{SP}$ cannot break both the vertices and the flags without the secret keys under CPA.

By the above argument, given a query $\mathsf{Reach}^e(u_e, v_e)$, $\mathcal{SP}$ cannot break $\mathtt{Lin}^e(v_e)$ and $\mathtt{Lout}^e(u_e)$. Since $\mathtt{Lin}^e(v_e) \cap \mathtt{Lout}^e(u_e)|$ of is *always* 1 and the result $(w_e,\ f_e)$ is encrypted, the $\mathcal{SP}$ does not gain any knowledge during query processing. Thus, the reachability of the query nodes is preserved. ■

Proposition 5.2: The edges of a graph are preserved under CPA. ■

Proof: This is established by a proof by contradiction. Suppose the $\mathcal{SP}$ can determine the existence of one edge (u, v), the $\mathcal{SP}$ breaks the reachability of at least one query $\mathsf{Reach}^e(u_e, v_e)$. This is a contradiction since the $\mathcal{SP}$ cannot break ppm-2-hop under CPA (by Prop. 5.1). ■

VI. Related Work

There have been related works on security of graph queries. One stream of works addresses the authenticity of subgraph data or query answers [7], [10], where the clients verify the data/answers returned by an $\mathcal{SP}$ are not tampered with.

Regarding the confidentiality of graph queries or data, He et al. [9] analyze the reachability of vertices and Gao et al. [8] propose neighborhood-privacy protected shortest distance. However, their privacy targets are different from our work. Mouratidis et al. [11] determine the shortest path of the query nodes with no information leakage by using PIR [6]. However, the high computational cost of PIR is still a concern. Cao et al. [3] consider subgraph queries but not reachability queries. Our previous work [13] also studies reachability queries but a different privacy target.

VII. Experimental Evaluation

In this section, we present an experimental evaluation to verify the performance of m-2-hop.

Experimental setup. We conducted all our experiments on a machine with a 3.40GHz CPU and 16GB memory. We benchmarked our implementation with 7 real-world datasets [2] and 3 synthetic datasets [13]. Some statistics of the datasets are reported in Table I. We recall that for each graph G, we construct the m-2-hop labels after reducing it to a DAG.

Effectiveness of m-2-hop. Table II reports the effectiveness of m-2-hop. It shows that the number of trues in $TC(G)$ ($|TC(G)|$) were around 9% of the $|V(G)|^2$. We also note that the sizes of m-2-hop of all real dataset were significantly smaller than $|V(G)|^2$ (at most 4%). The construction times of m-2-hop were at most 53min. We remark the construction can be further optimized by adopting existing work (*e.g.*, [5]).

[2]Real-1: ERDOS was from [13]. Real-2 (Cit-HepPh), Real-3 (p2p-Gnutella31), Real-4 (soc-Epinions1), Real-5 (soc-sign-Slashdot090221), Real-6 (soc-Slashdot0922) and Real-7 (ego-Twitter) were from [1]

TABLE I
Statistics of real-world and synthetic datasets

Graph G	$\lvert V(G)\rvert$	$\lvert E(G)\rvert$	$\lvert V(G)\rvert/\lvert E(G)\rvert$	$\lvert V(\mathtt{DAG}(G))\rvert$	$\lvert E(\mathtt{DAG}(G)\rvert)$
Syn-1	3,073	37,615	0.08	3,073	37,615
Syn-2	5,651	15,968	0.35	5,651	15,968
Syn-3	4,880	27,946	0.17	4,880	27,946
Real-1	6,927	11,850	0.58	6,927	11,850
Real-2	34,546	421,578	0.08	21,608	281,030
Real-3	62,586	147,892	0.42	48,438	96,976
Real-4	75,879	508,837	0.15	42,176	61,995
Real-5	82,140	549,202	0.15	53,599	200,101
Real-6	82,168	948,464	0.09	10,559	28,331
Real-7	81,306	2,420,766	0.03	12,248	95,659

TABLE II
Statistics of m-2-hop

Graph G	|m-2-hop|	$\lvert TC(\mathtt{DAG}(G))\rvert$	$\lvert V(\mathtt{DAG}(G))\rvert^2$	Constr. time
Syn-1	2.11M	2.91M	9.44M	11s
Syn-2	7.30M	0.29M	31.93M	1min25s
Syn-3	9.24M	2.44M	23.81M	1min35s
Real-1	1.49M	400K	47.98M	22s
Real-2	19.13M	84.25M	466.90M	16min43s
Real-3	11.34M	18.17M	2.35G	28min51s
Real-4	12.02M	348.83M	1.78G	28min20s
Real-5	20.33M	371.90M	2.87G	52min38s
Real-6	434.15K	21.18K	111.49M	18s
Real-7	606.77K	37.78K	150.01M	32s

Query performance. We randomly selected 1k pairs of vertices as our queries. The total times at the $\mathcal{SP}$ side for running all queries were small, in particular, at most 0.18s and 1.4s for real datasets and synthetic datasets, respectively. Due to space constraints, we did not elaborate them.

VIII. Conclusion and Future Work

We proposed 2-hop labeling, namely m-2-hop, to provide privacy-preserving query services for reachability queries under the paradigm of data outsourcing. A MSC-based heuristic construction algorithm for m-2-hop. We verify through an experiment that m-2-hop is efficient for real-world and synthetic datasets. In the future, we plan to investigate the approaches for large graphs (*e.g.*, social networks) that cannot fit in the main memory. We also plan to integrate the large body of optimizations for 2-hop labeling into m-2-hop.

References

[1] Stanford large network dataset collection. *http://snap.stanford.edu/data/*.
[2] R. Bramandia et al. Incremental maintenance of 2-hop labeling of large graphs. *TKDE*, 22(5):682–698, 2010.
[3] N. Cao et al. Privacy-preserving query over encrypted graph-structured data in cloud computing. In *ICDCS*, pages 393–402, 2011.
[4] J. Cheng et al. Fast computation of reachability labeling for large graphs. EDBT, pages 961–979, 2006.
[5] J. Cheng et al. Fast computing reachability labelings for large graphs with high compression rate. EDBT, pages 193–204, 2008.
[6] B. Chor et al. Private information retrieval. *J. ACM*, 45:965–981, 1998.
[7] Z. Fan et al. Towards efficient authenticated subgraph query service in outsourced graph databases. *TSC*, 99, 2013.
[8] J. Gao et al. Neighborhood-privacy protected shortest distance computing in cloud. SIGMOD, 2011.
[9] X. He et al. Reachability analysis in privacy-preserving perturbed graphs. WI-IAT, pages 691–694, 2010.
[10] A. Kundu et al. Efficient leakage-free authentication of trees, graphs and forests. *IACR Cryptology ePrint Archive*, page 36, 2012.
[11] K. Mouratidis et al. Shortest path computation with no information leakage. *PVLDB*, 2012.
[12] H. Wang et al. Dual labeling: Answering graph reachability queries in constant time. In *ICDE*, 2006.
[13] P. Yi et al. Privacy preserving reachability query services. *Technical report*, 2013. *http://www.comp.hkbu.edu.hk/∼zfan/2013-02.pdf*.