Efficient Privacy-Preserving Link Discovery

Xiaoyun He[1], Jaideep Vaidya[1], Basit Shafiq[1], Nabil Adam[1],
Evimaria Terzi[2], and Tyrone Grandison[2]

[1]CIMIC, Rutgers University, USA
[2]IBM Almaden Research Center, USA
{xiaoyun,jsvaidya,basit,Adam}@cimic.rutgers.edu
{eterzi,tyroneg}@us.ibm.com

Abstract. Link discovery is a process of identifying association(s) among different entities included in a complex network structure. These association(s) may represent any interaction among entities, for example between people or even bank accounts. The need for link discovery arises in many applications including law enforcement, counter-terrorism, social network analysis, intrusion detection, and fraud detection. Given the sensitive nature of information that can be revealed from link discovery, privacy is a major concern from the perspective of both individuals and organizations. For example, in the context of financial fraud detection, linking transactions may reveal sensitive information about other individuals not involved in any fraud. It is known that link discovery can be done in a privacy-preserving manner by securely finding the transitive closure of a graph. We propose two very efficient techniques to find the transitive closure securely. The two protocols have varying levels of security and performance. We analyze the performance and usability of the proposed approach in terms of both analytical and experimental results.

Key words: Privacy, Link Discovery, Efficiency

1 Introduction

Link discovery is a process of identifying association(s) among different entities included in a complex network structure [1, 2]. These association(s) may represent any interaction among entities, for example, between people or even bank accounts. The need for link discovery arises in many applications including law enforcement, counter-terrorism, social network analysis, intrusion detection, and fraud detection.

Link discovery in these application domains often involves analysis of huge volumes of data distributed across different sources with different rules and regulations on data sharing. For instance, law enforcement investigations often involve finding links between individuals or discovering association of individuals with specific organizations or groups [2]. To discover such links, information needs to be sifted through various sources such as law enforcement databases, financial transactions, and phone records, etc. The information stored in such data repositories is often confidential. Given the sensitive nature of information that can be revealed from link discovery, privacy is a major concern for both individuals and organizations [3].

In the past years, there has been increasing interest in developing techniques for link discovery and analysis in network or graph structured data [1, 4–6]. However, none of these works have considered privacy issues in a distributed context while discovering links among entities. Recent work by Duan et al. [7] first presents a generalized HITS algorithm to rank the linked entities on weighted graph, they solve an orthogonal problem to what we present in this paper. There has also been increasing interest in privacy-preserving data mining [8–10], some of which work is of interest.

Recently, He et al.[2] have proposed an approach for privacy-preserving link discovery in a complex and distributed network structure. Specifically, the entities in the network structure are viewed as nodes in a graph with an edge between two nodes representing the association between the corresponding entities. Different portions of the graph (subgraphs) correspond to data repositories owned by different parties. For example, in the context of financial transactions, the graph nodes represent customer accounts and the links represent the transaction among accounts, which may belong to the same bank or different banks. Thus, the entire graph represents the global view including all information repositories. The problem of privacy-preserving link discovery can then be reduced to finding the transitive closure of a distributed graph in a secure manner. He et al.[2] further show that this can be done via a split matrix multiplication protocol, which itself can be implemented using the completely secure scalar product protocol of Goethals et al.[11]. However, due to the required large number of costly encryption / decryption operations, the approach is computationally quite prohibitive.

In this paper, we propose two different methods to improve the computational efficiency of secure link discovery. The first method is based on commutative encryption. This approach leaks a little more information (each party gets to know the final transitive closure matrix involving their own vertices as well a range on when these connections are formed). However, it is significantly more efficient and practical. In the second method, the secure transitive closure is computed over a much smaller graph composed of representative nodes heuristically selected. Instead of including all of its nodes for a complete secure transitive closure computation, each party chooses a fraction of its overall nodes as its representatives set to form its representative matrix, and the secure transitive closure computed over it. The global transitive closure is then approximated based on the representative transitive closure. Our experiments show the effectiveness of our proposed approach.

2 Preliminaries and Problem Definition

In this paper, we consider a distributed environment with k parties $P_1, \ldots, P_k$. The overall distributed network data is modelled as a simple directed graph $G(V, E)$, where V is a set of nodes with $|V| = n$, and $E \subseteq V \times V$ is a set of directed edges. Each party P_i owns part of graph G, denoted by $G(V_i, E_i)$, where $\cup_i V_i = V$, $|V_i| = m_i$ with $\sum_{i=1}^{k} m_i = n$, $V_i \cap V_j = \emptyset (i \neq j)$, and $\cup_i E_i \subseteq E$. Note that some edges in E may cross the boundaries of G_i and $G_j (i \neq j)$ with one node in G_i and another node in G_j. These edges are called inter-edges.

Given two nodes u, v in V we define the predicate DIRPATH(U,V) as evaluating to 1 if there exists a directed path between u and v, and otherwise evaluating to 0. Given any

two distinct nodes $u, v \in V$, we are interested in being able to check whether there is a path from u to v in the global graph G no matter the subgraph in which the two nodes u and v reside. That is, we are interested in evaluating predicate DIRPATH(u, v) for every pair of nodes $u, v \in V$ in a *privacy-preserving* manner. In other words, we require that the process of evaluating predicate DIRPATH should not reveal any additional information to any party after the computation. Evaluating DIRPATH in this setting leads to the following definition of the PRIVACY-PRESERVING LINK DISCOVERY.

Problem 1. Consider directed graph $G(V, E)$ split among k parties $P_1, \ldots, P_k$ as described above. For every pair of nodes $u, v \in V$ evaluate the value of predicate DIRPATH (u, v) in G in a privacy-preserving manner.

We also require the following definitions:

Definition 1. *Given party P_i that keeps graph $G_i(V_i, E_i)$ we define the* set of inter-nodes *$V_I^{(i)}$ to be the union set of nodes that either start or end inter-edges.*

In addition, we denote V_{I} as the set union of all $V_{\mathrm{I}}^{(i)} (i = 1, 2, \ldots, k)$.

Definition 2. *Given a node $u \in V_i$, its inter-degree $D_I(u)$ is defined as the total number of edges which are either $(u, v) \in E$ or $(v, u) \in E$, and $v \in V_I^{(j)} (i \neq j)$.*

Definition 3. *Given a node $u \in V_i$, its local-degree $D_L(u)$ is defined as the total number of edges which are either $(u, v) \in E_i$ or $(v, u) \in E_i$.*

Definition 4. *Given a node $u \in V_i$, its combined-degree $D_C(u)$ is defined as the sum of $D_I(u)$ and $D_L(u)$.*

In addition, given a graph $G = (V, E)$, its final full transitive closure is denoted by TC. In our second proposed approach, we derive an approximate transitive closure of G, denoted by TC'. To assess the effectiveness of our proposed heuristic approach, we also define two measures as follows.

Definition 5. *The total accuracy is defined as the total number of matched elements between TC and TC' divided by n^2 (that is, $\frac{total\ number\ of\ matches}{n^2}$), where n is the size of the adjacency matrix.*

Definition 6. *The edge accuracy is defined as the total number of matched non-zero elements between TC and TC' divided by the total number of non-zero elements (that is, $\frac{total\ number\ of\ matched\ non-zeros}{total\ number\ of\ non-zeros}$).*

Note that, in the above definitions, matched elements mean that, given a specified row and column, the corresponding entries in the matrices of TC and TC' have the same value. The reason that we would like to include both total accuracy and edge accuracy as performance metrics is that, in the case of sparse graphs, total accuracy can be very high while having low edge accuracy. In general, a high edge accuracy is an indicator of good performance. In the following, whenever accuracy is mentioned, it means edge accuracy unless otherwise stated.

3 Overview of Secure Transitive Closure

In this section, we briefly provide an overview of secure transitive closure introduced in [2]. For more details, we refer the readers to [2]. The transitive closure [12] of a graph $G = [V, E]$ is a graph $G^* = [V, E^*]$ with edge $(i, j) \in E^*$ if and only if there is a path from vertex i to vertex j in the graph G. A simple matrix multiplication method can be used to compute the transitive closure of a graph. If A represents the adjacency matrix of graph G, then A^n represents the transitive closure G^*, where n is the number of vertices in G.

To address the privacy concerns, the approach proposed in [2] enables secure computation of the transitive closure of a distributed graph without requiring parties to reveal any details about their subgraphs. Specifically, the protocol for secure transitive closure computation is run by all k parties that own a portion of the distributed graph. Let $A^{(i)}$ denote the adjacency matrix corresponding to the subgraph $G_i = (V_i, E_i)$ owned by party P_i. $A^{(i)}$ is a $n \times n$ matrix, where the matrix entry $A^{(i)}[p, q] = 1$ if the edge $(p, q) \in E_i$. All other entries in the matrix $A^{(i)}$ are set to zero. Therefore, the overall adjacency matrix A of the distributed graph is given by: $A = \sum_{i=1}^{k} A^{(i)}$.

The transitive closure A^n of the distributed graph is computed iteratively through matrix multiplication, with the output of the last iteration used in this iteration . For instance in the rth iteration, $(r \leq n)$, the matrix A^{2r} is computed as follows:

$$A^{2r} = A^r A^r = \sum_{i=1}^{k} O^{(i)} \sum_{j=1}^{k} O^{(j)} \tag{1}$$

The Split Matrix Multiplication Equation 1 used for computation of the transitive closure involves pair wise multiplication of the output split matrices of each party. It is obvious that each party P_i can locally compute $O^{(i)}O^{(i)}$. Therefore, the secure computation of A^{2r} comes down to securely computing $O^{(i)}O^{(j)}(\forall i \neq j)$. Since the matrix multiplication essentially is the scalar product operations, $O^{(i)}O^{(j)}$ can be computed by invoking the secure scalar product protocol proposed in [11].

It is important to note that actual adjacency matrix of the distributed graph in each iteration is never known completely to any party. Rather, as the output of each iteration, each party P_i gets a matrix $O^{(i)}$ consisting of random shares of the global adjacency matrix. Thus, for any given row p and column q $(1 \leq p, q \leq n)$, and iteration r, $\sum_{i=1}^{k} O^{(i)}(p, q) = A^r(p, q)$. In the end, the values of the final matrix A^n are split randomly and returned to each party as matrices $O^{(1)}, O^{(2)}, \ldots, O^{(k)}$.

However,the above approach in [2] is computationally prohibitive. With $log_2 n$ iterations, the total number of encryptions and decryptions required performed is $(k^2 - k) \cdot n^2 \cdot log_2 n$, while the total number of exponentiations and multiplications performed is $(k^2 - k) \cdot n^3 \cdot log_2 n$. Overall the encryption/decryption time dominates. Although the split matrix multiplication approach requires $O(n^2)$ encryption/decryptions, the computational time for large distributed graphs will be significantly high due to the high computational cost of encryption/decryption.

Algorithm 1 Efficient Secure Transitive Closure

Require: k parties, $P_1, \ldots, P_k$, Party P_i has m_i vertices
Require: Let $n = \sum_{i=1}^{k} m_i$, represent the total number of vertices
Require: Let the matrix $A^{(i)}(n \times n)$ represent the local adjacency matrix of party P_i (i.e., the matrix entry $A^{(i)}[p, q] = 1$ if the edge $(p, q) \in E_i$, otherwise 0.)

```
1: for j ← 1 ... ⌈log_2 n⌉ do
2:   for p ← 1 ... n do
3:     for q ← 1 ... n do
4:       {Assume P_i owns vertex p}
5:       At P_i: Initiate boolean scalar product protocol described in Algorithm 2 to get output
         value x
6:       if x = 0 then
7:         At P_i: A^(i)[p, q] ← 0
8:       else
9:         At P_i: A^(i)[p, q] ← 1
10:      end if
11:    end for
12:  end for
13: end for
```

4 A Commutative Encryption based Approach

In this section, we provide an alternative approach that uses a much more efficient protocol for the scalar product requiring relatively fewer encryption/decryption operations. The protocol is depicted in Algorithm 1. This approach also uses split matrix multiplication for secure computation of the transitive closure matrix. However, it employs commutative encryption for computation of the scalar product and works only if boolean values are used. In other words, the split matrices generated in each iteration of the split matrix multiplication needs to be converted into boolean values. This will result in leakage of additional information to the different parties. In particular, each party will know the portion of the final transitive closure matrix involving the party's own vertices. Additionally, each party will know in which iteration a zero value in its local output matrix changes to a non-zero value. As a result, the party will know the range on the number of links to which its local vertices are connected to external vertices. For example, if the matrix entry $O^{(i)}[p, q]$ changes its value from zero to non-zero in the 3rd iteration, then the shortest path between vertex p and q consists of at least 4 and at most 8 links. One way to reduce this leakage is to use a hybrid approach, where the approach proposed in [2] is used for the first few iterations before switching to the new approach.

4.1 Commutative Encryption based Scalar Product

With boolean vectors, it is possible to get a more efficient scalar product. To see this, note that if we encode the vectors as sets (with position numbers as elements), the scalar product is the same as the size of the intersection set. For example, assume we have vector $\boldsymbol{X} = (1, 0, 0, 1, 1)$ and $\boldsymbol{Y} = (0, 1, 0, 1, 0)$. Then the scalar product

$\boldsymbol{X} \cdot \boldsymbol{Y} = \sum_{i=1}^{5} x_i * y_i$. Now, the corresponding set encodings are $XS = (1, 4, 5)$ and $YS = (2, 4)$. Once can see that the size of the intersection set $|XS \bigcap YS| = 1$ is exactly the same as the scalar product. This idea is used to compute the scalar product.

The basic idea is to use commutative encryption to encrypt all of the items in each party's set. Commutative encryption is an important tool used in many cryptographic protocols. An encryption algorithm is commutative if the order of encryption does not matter. Thus, for any two encryption keys $E1$ and $E2$, and any message m, $E1(E2(m)) = E2(E1(m))$. The same property applies to decryption as well – thus to decrypt a message encrypted by two keys, it is sufficient to decrypt it one key at a time. The basic idea is for each source to encrypt its data set with its keys and pass the encrypted data set to the next source. This source again encrypts the received data using its encryption keys and passes the encrypted data to the next source until all sources have encrypted the data. Since we are using commutative encryption, the encrypted values of the set items across different data sets will be equal if and only if their original values are equal. Thus, all the intersection of the encrypted values gives the logical AND of the vectors, and counting the size of the intersection set gives the total number of 1s (i.e., the scalar product). The encryption prevents any party from knowing the actual value of any local item. This scalar product method only works for boolean vectors, but it will still work in this context, since after each iteration the non-zero values in the local adjacency matrix are set to one by the party owning the corresponding data point.

In our case, for the scalar product, the first vector is owned completely by one party while the second vector is split between all of the parties. One can simply compute all of the local scalar products to add up the sum to get the global scalar product. However, this creates a serious security problem. To see this, assume that a party P_i owning m_i vertices, gets local scalar products from another party P_j owning m_j vertices. Remember that each scalar product gives one linear equation in unknowns. Since party P_i owns m_i vertices, it gets m_i linear equations in m_j unknowns. If $m_i > m_j$, this will completely breach the security of party P_j. Thus, if there is even one party that has more vertices than any of the other parties, it can completely breach the security of the other parties. Since this situation is quite likely, local scalar products cannot be used.

Instead to ensure security, we must carry out the entire scalar product in one go. To do this securely, we must ensure that all of the vectors are encrypted and permuted by all of the parties, thus ensuring that no linkage between vectors can be done. Now, after intersection a party can only learn the total scalar product (not any of its components). Algorithm 2 gives the complete details. This still gives it some linear equations – in fact, it gives in m_i linear equations in $n - m_i$ unknowns. As long as m_i is not more than half of the total number of vertices, security is not breached. In most situations this will be true and this protocol can be used. In cases where this is not true, there is no alternative to the first completely secure protocol.

4.2 Complexity Analysis

We now analytically show that this method is more efficient than the approach in [2]. Assume that c denotes the total number of 1s in the global adjacency matrix in a particular iteration. Further assume that these are split into c_i $1s$ for each row and c_i' $1s$ for each column. Thus, $c = \sum_{i=1}^{n} c_i$. Similarly, $c = \sum_{i=1}^{n} c_i'$.

Algorithm 2 Commutative Encryption based Boolean Scalar Product

Require: k parties, $P_1, \ldots, P_k$
Require: Party P_1 has input vector $\boldsymbol{X} = \{x_1, \ldots, x_n\}$
Require: Party $P_1, \ldots, P_n$ each have input vectors $\boldsymbol{Y_i} = \{y_1, \ldots, y_{m_i}\}$, where m_i represents the number of vertices owned by each party, such that $\sum_{i=1}^{k} m_i = n$
Require: Assume that $\boldsymbol{Y} = [\boldsymbol{Y_1} \ldots \boldsymbol{Y_k}]$
Require: P_1 gets output o such that $o = \boldsymbol{X} \cdot \boldsymbol{Y}$
Require: A global position encoding scheme
1: Each party P_i generates a private and public key pair (sk_i, pk_i) for a commutative encryption system.
2: P_1 converts its vector $\boldsymbol{X}$ to the position set XS
3: Each party P_i converts its local vector $\boldsymbol{Y_i}$ to the position set YS_i based on the global encoding scheme
4: **for** $i = 1 \ldots n$ **do**
5: P_1 encrypts the position set XS with its key E_{pk_1} to get the encrypted vector EXS
6: **end for**
7: **for** $j = 1 \ldots k$ **do**
8: Each Party P_j encrypts its local position set YS_j with its key E_{pk_j} to get the encrypted position set EYS_j
9: **end for**
10: Each party passes its encrypted position set to the next party for encryption with its key until all sets are encrypted by all parties
11: At P_k: $EYS \leftarrow \phi$
12: **for** $j = k \ldots 2$ **do**
13: Party P_j merges its completely encrypted set with the global encrypted set EYS, i.e. $EYS \leftarrow EYS \cup EYS_j$
14: Party P_j arbitrarily permutes EYS and sends it to party P_{j-1}
15: **end for**
16: At P_1: Receive EYS from P_2 and merge EYS_1 into it (i.e., $EYS \leftarrow EYS \cup EYS_1$)
17: P_1 intersects the completely encrypted set EXS with the complete encrypted set EYS to get the output o

In each iteration, for each point in the global adjacency matrix, one efficient commutative encryption based scalar product is carried out. Thus, for row p and column q, the scalar product requires $c_p * k + c'_q * k$ encryptions. Thus, the total cost of each iteration can be given by summing the total number of iterations required for each row and column. However, this assumes that we reencrypt for every row and column for each scalar product, which is quite unnecessary. In reality, it is sufficient to encrypt each row and each column only once. The same encryptions can be used for successive scalar products without revealing any extra information. Thus, total cost, TC is

$$TC = \sum_{p=1}^{n} c_p * k + \sum_{q=1}^{n} c'_q * k = k \sum_{p=1}^{n} c_p + k \sum_{q=1}^{n} c'_q = kc + kc = 2kc$$

In general, c can range between n and n^2. Therefore, in the best case, $TC = 2kn$, while in the worst case, $TC = 2kn^2$. It is important to note that for large distributed graphs typically the values of c are asymptotically closer to the best case value rather

than the worst case. For instance in the distributed graph linking financial transactions across different bank accounts, it is unlikely that a single transaction can be linked to all transactions or even a fraction of these. We can safely assume that the number of transactions that can be linked to a single transaction will always be bounded by a constant, i.e., $c = O(n)$.

5 A Heuristic Approach based on Representative Selection

In this section, we present a heuristic approach to improve the efficiency of secure computation for those situations where the commutative encryption based approach cannot be used. The basic idea is to have each party choose a fraction of its overall nodes as its representatives set to form the representative matrix. Then, the secure transitive closure is done only over the representative matrix. The global transitive closure is now inferred using the representative transitive closure. With a small representative matrix, this clearly leads to significantly smaller computation costs. However, this pays a price in accuracy. While the links between the representative are accurately discovered, for the remaining nodes, the links may or may not be discovered. While there will be no false positive (a link found where none exists), there can be significant false negatives, based on how few representatives are chosen. Algorithm 3 gives the details.

We need to further discuss two issues – how are the representatives chosen, and how is the global transitive closure inferred from the representative transitive closure. We first discuss the second issue: Specifically, given any pair of nodes (u, v), where $u \in V_i$ and $v \in V_j (i \neq j)$, evaluate DIRPATH(u, v). The following 3 cases may occur:

- both u and v are representatives: in this case, we can directly get the answer from the transitive closure T_R.
- one of u and v is a representative: without loss of generality, we assume that u is a representative. If we can find a node v' which is a representative of the party who owns v and we also know there is a path from u to v' and a path from v' to v based on step 3 and step 1, respectively, then we say a path exists from u to v. Otherwise, no path exists between them.
- neither u nor v is a representative: If we can find a node u' (resp. v') which is a representative of the party who owns u(resp. v) and we also know there is a path from u to u' and a path from v' to v based on step 1, as well as a path exists from u' to v' based on step 3, then we say a path exists from u to v. Otherwise, no path exists between them.

Now, for the first question – how do we select the representatives to maximize accuracy for a given level of efficiency.

Intuitively, the representatives should be chosen from the set of the inter-nodes in each subgraph. This makes sense, since these are the only nodes involved in any edges with intra-edges. These are our only sources of cross-graph information. As we have explained above, combining our cross-graph path information with the local paths in each subgraph will help us to discover the path between any pair of nodes residing in different subgraphs. Assuming that the inter-nodes is a fraction of total nodes, choosing inter-nodes as representatives would easily reduce the required secure computations.

If we include all the inter-nodes as representatives, obviously we would get the exact results of all path information.

In this paper, we employ a greedy heuristic to choose representatives from within the inter-nodes. The idea is to choose representative nodes with high degrees. The intuition is that a node with higher degree should be involved in more paths and thus contribute more information.

One natural and seemingly better selection criteria is to have a greedy global selection - we choose representatives with high degree in the overall global graph G. However, given that each party can only see its own subgraph, this is not ideal, since it would not address privacy concerns.

Instead, we take the approach of greedy local representatives selection. In a greedy local selection, we choose the inter-nodes which have high degree in each party's local subgraph. Again, three different kinds of degree could be used - inter-degree, local-degree, and combined-degree (as we have defined in Section 2). In each case, the corresponding degree of each node is computed. Then, each party keeps a specified percentage of its local nodes with highest degrees being its representatives, which are used for forming representative matrix R. Algorithm 4 presents the details of this. As we show in the experimental evaluation below, this works quite well.

5.1 Experimental Evaluation

In this section we experimentally evaluate the effectiveness of our proposed algorithms. Synthetic random graphs are generated for the test datasets. In fact, random graphs are widely used in the probabilistic method, where one tries to prove the existence of graphs with certain properties. The existence of a property on a random graph implies, via the famous Szemeredi regularity lemma, the existence of that property on almost all graphs [13].

Since we need to have k sub-graphs making up a global graph, we use the igraph package [1] to generate a specific type of global graph G (i.e, Erdős-Renyi random graph). Then, we uniformly at random choose a certain number of nodes (i.e, a specified percentage of the total number of nodes n) to induce each subgraph G_i and the corresponding inter-edges between these subgraphs. In the ER $G(n, p)$ model, a graph is constructed by connecting nodes randomly, where n is the total number of nodes, and p is the probability that each edge is included in the graph, with the presence or absence of any two distinct edges in the graph being independent. We partition the global graph G into 4 equal-size subgraphs, each of which is assumed to be owned by a party. This partition also results in a number of inter-edges connecting the subgraphs.

Figure 1 shows the accuracy results of the greedy local, greedy global, and random approaches with the representative rate goes from 10% up to 100%. All the tests are done on the Erdős-Renyi graphs with the number of nodes n = 1000 and probability p = 0.1%. In the results, lid, lld, and lcd to stand for the local inter-degree, local local-degree, and local combined-degree approach, respectively. In the global case, the global inter-degree, global local-degree, global combined-degree is denoted by gid, gld, and gcd, respectively. The representative rate is the fraction of representative nodes chosen

[1] http://cneurocvs.rmki.kfki.hu/igraph/

Algorithm 3 Secure Representatives Approach

Require: k parties $P_1, \ldots, P_k$
Require: Let the matrix $A^{(i)}(n_i \times n_i)$ represent the local adjacency matrix of party P_i
Require: Let the matrix $R(|V_{\mathrm{R}}| \times |V_{\mathrm{R}}|)$ be the representative matrix (i.e., the matrix entry $R[p,q] = 1$ if $p \in V_{\mathrm{R}}^{(i)}, q \in V_{\mathrm{R}}^{(j)}$, otherwise 0.)
1: Each party P_i computes its local transitive closure $T_L^{(i)}$ with input matrix $A^{(i)}$
2: Each party P_i engages in heuristically choosing representatives (Algorithm 4) to have the matrix R
3: Each party P_i participates secure transitive closure computation described in Section 3 with input matrix R to get transitive closure T_R
4: Given a pair of nodes (u, v), where $u \in V_i$ and $v \in V_j (i \neq j)$:
5: **if** both u and v are representatives {i.e.($u \in V_{\mathrm{R}}^{(i)}$ and $v \in V_{\mathrm{R}}^{(j)}$)} **then**
6: DIRPATH$(u, v) \leftarrow T_R(u, v)$
7: **else if** one of u and v is a representative {w.l.o.g assume that u is the representative i.e., $u \in V_{\mathrm{R}}^{(i)}$ and $v \notin V_{\mathrm{R}}^{(j)}$} **then**
8: **if** $\exists v' \in V_{\mathrm{R}}^{(j)}$ and $T_R(u, v') \neq 0$ and $T_L^{(i)}(v', v) \neq 0$ **then**
9: DIRPATH$(u, v) \leftarrow 1$
10: **else**
11: DIRPATH$(u, v) \leftarrow 0$
12: **end if**
13: **else**
14: {neither is a representative, i.e., $u \notin V_{\mathrm{R}}^{(i)}$ and $v \notin V_{\mathrm{R}}^{(j)}$}
15: **if** $\exists u' \in V_{\mathrm{R}}^{(i)}$ and $\exists v' \in V_{\mathrm{R}}^{(j)}$ s.t. $T_L^{(i)}(u, u') \neq 0$ and $T_L^{(j)}(v', v) \neq 0$ and $T_R(u', v') \neq 0$ **then**
16: DIRPATH$(u, v) \leftarrow 1$
17: **else**
18: DIRPATH$(u, v) \leftarrow 0$
19: **end if**
20: **end if**
21: **return** DIRPATH(u, v)

either locally or globally from the inter-nodes. Each approach is run on the same graph, and the results averaged over five runs (with different graphs).

Figure 1(a) shows that, in the greedy local case, both the combined-degree and the inter-degree approaches achieve a better accuracy than the local-degree one, and the combined-degree is slightly better than the inter-degree. The greedy global approach in figure 1(b) looks similar to the greedy local one. In addition, in both greedy local and greedy global cases, the combined-degree has the best performance. Figure 1(c) compares the global and local combined-degree with random selection. Clearly, both global and local approaches perform much better than the random one. More importantly, we can see that the greedy local combined-degree approach almost performs the same as the greedy global combined-degree. Hence, it demonstrates that our proposed greedy local heuristics approach is promising.

While these results are preliminary, we have run more experiments varying other parameters as well as the graph generation model. We do not report these due to space

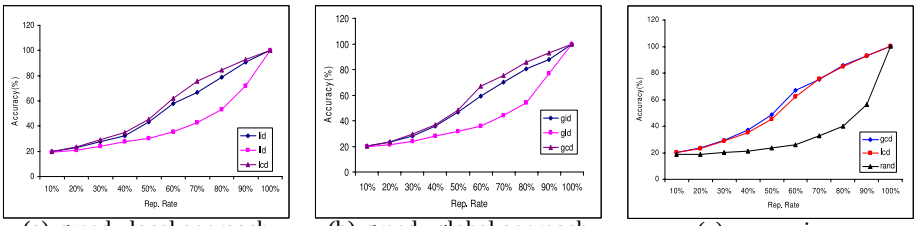
Algorithm 4 Choose Representatives using Heuristics (DegreeType, Percentage): a greedy local approach

Require: k parties $P_1, \ldots, P_k$ each holding subgraph $G_i(V_i, E_i)$ as parts of global graph $G(V, E)$
Require: DegreeType: chosen from inter-degree, local-degree, or combined-degree
Require: Percentage: representative rate as opposed to the total number of the inter-nodes ($|V_{\mathrm{I}}|$)
1: At P_i:
2: the representative set $RS_i \leftarrow V_{\mathrm{I}}^{(i)}$
3: **for** each node $u \in V_{\mathrm{I}}^{(i)}$ **do**
4: **if** DegreeType = inter-degree **then**
5: Count the inter-degree $D_{\mathrm{I}}(u)$ of u
6: **else if** DegreeType = local-degree **then**
7: Count the local-degree $D_{\mathrm{L}}(u)$ of u
8: **else**
9: Count the combined-degree $D_{\mathrm{C}}(u)$ of u
10: **end if**
11: **end for**
12: Sort RS_i in terms of the degree counts
13: Keep Percentage*$|V_{\mathrm{I}}|$ of nodes with the highest degree in RS_i
14: Each party participates the forming of the matrix R using RS_i (Similar to the formation of adjacency matrix A discussed in Section 2)
15: **return** R

limitations, but they are quite similar and show that the representatives approach gives a compelling tradeoff of accuracy for efficiency.

6 Concluding Remarks

In this paper, we have proposed two different approaches to improve the efficiency for privacy-preserving link discovery in a complex and distributed network structure. The first approach trades off security for efficiency, while the second trades off accuracy for efficiency. Both of our approaches can reduce the prohibitive computational complexity of the currently existing solution for secure link discovery.



(a) greedy local approach (b) greedy global approach (c) comparison

Fig. 1. Local vs. global vs. random approaches

In our future work, we will consider other features of interest such as the degree of closeness of the entities (i.e., number of common neighbors, number of distinct paths, length of the shortest path, etc) for link discovery and analysis. A more challenging problem is to figure out the maximum flow from one entity to another (the max flow problem). This can be instrumental in computing the amount of resources transported through multiple intermediaries which would be great interest in financial fraud detection.

References

1. Getoor, L., Diehl, C.P.: Link mining: a survey. SIGKDD Explorations **7**(2) (2005) 3–12
2. He, X., Shafiq, B., Vaidya, J., Adam, N.: Privacy-preserving link discovery. In Wainwright, R.L., Haddad, H., eds.: SAC, ACM (2008) 909–915
3. Sweeney, L.: Privacy-enhanced linking. SIGKDD Explorations **7**(2) (2005) 72–75
4. Xu, J.J., Chen, H.: Fighting organized crimes: using shortest-path algorithms to identify associations in criminal networks. Decision Support Systems **38**(3) (2004) 473–487
5. Ganiz, M.C., Pottenger, W.M., Yang, X.: Link analysis of higher-order path in supervised learning datasets. In: In proceedings of the 4th SIAM Workshop on Link Analysis, Counterterrorism and Security. (April 2006)
6. Mooney, R., Melville, P., Tang, L., Shavlik, J., Dutra, I., Page, D., Costa, V.: Relational data mining with inductive logic programming for link discovery. In: In Proceedings of the National Science Foundation Workshop on Next Generation Data Mining, Baltimore, Maryland (2002)
7. Duan, Y., Wang, J., Kam, M., Canny, J.: A secure online algorithm for link analysis on weighted graph. In: In Proceedings of SIAM Workshop on Link Analysis, Counterterrorism and Security. (April 2005)
8. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proceedings of the 2000 ACM SIGMOD Conference on Management of Data. (2000) 439–450
9. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Advances in Cryptology – CRYPTO 2000, Springer-Verlag (August 20-24 2000) 36–54
10. Vaidya, J., Clifton, C., Zhu, M.: Privacy-Preserving Data Mining. 1st edn. Advances in Information Security. Springer-Verlag (2005)
11. Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On Secure Scalar Product Computation for Privacy-Preserving Data Mining. In Park, C., Chee, S., eds.: The 7th Annual International Conference in Information Security and Cryptology (ICISC 2004). Volume 3506. (December 2–3, 2004) 104–120
12. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. McGraw-Hill Book Company, New York (1990)
13. Bollobs, B.: Random Graphs. 2nd edn. Cambridge University Press (2001)