Privacy-Preserving Link Prediction

Didem Demirag[1](✉), Mina Namazi[2], Erman Ayday[3], and Jeremy Clark[1]

[1] Concordia University, Montreal, QC, Canada
d_demira@encs.concordia.ca, j.clark@concordia.ca
[2] Open University of Catalonia, Barcelona, Spain
mnamaziesfanjani@uoc.edu
[3] Case Western Reserve University, Cleveland, OH, USA
exa208@case.edu

Abstract. Consider two data holders, ABC and XYZ, with graph data (*e.g.,* social networks, e-commerce, telecommunication, and bioinformatics). ABC can see that node A is linked to node B, and XYZ can see node B is linked to node C. Node B is the common neighbour of A and C but neither network can discover this fact on their own. In this paper, we provide a two party computation that ABC and XYZ can run to discover the common neighbours in the union of their graph data, however neither party has to reveal their plaintext graph to the other. Based on private set intersection, we implement our solution, provide measurements, and quantify partial leaks of privacy. We also propose a heavyweight solution that leaks zero information based on additively homomorphic encryption.

Keywords: Link prediction · Common neighbour · Privacy preserving graph mining · Private set intersection · Social network graphs

1 Introduction

Link prediction discovers important linkages between nodes in a graph. Based on the analysis of these linkages, it helps the data holder to forecast what future connections might emerge between the nodes, and to predict if there are missing links in the data. Some common applications include: (i) in social networks, to recommend links between users; (ii) in e-commerce or personalized advertisement, to recommend products to users; (iii) in telecommunication, to build optimal phone usage plans between the users; and (iv) in bioinformatics, to predict associations between diseases and attributes of patients or to discover associations between genes (or proteins) and different functions.

Link prediction is typically done on the local graph of a data holder or service provider. For instance, a social network, analyzing the common neighbours between its users decides whether to recommend links between the users. However, link prediction will be more accurate and correct by considering more information about the graph nodes. This can be achieved by merging two or

J. Garcia-Alfaro et al. (Eds.): DPM 2022/CBT 2022, LNCS 13619, pp. 35–50, 2023.
https://doi.org/10.1007/978-3-031-25734-6_3

more graph databases that include similar information, leading to "distributed link prediction" between two or more graph databases. For instance, two social networks may utilize the connections in their combined graph to provide more accurate link prediction for their users. Furthermore, distributed link prediction will enable different uses of link prediction, such as building connections between users and products based on the tastes of other similar users (*e.g.,* friends of a user). Such an application may be possible between graph databases of a social network and an e-commerce service provider. In some cases, collaboration is mutually beneficial to both parties. In others, one party can pay the other party to participate—one party gets better data and the other gets to monetize its data.

Distributed link prediction, although is a promising approach for more accurate and richer link prediction applications, also results in privacy concerns since it implies combining two or more different graph databases. In this scenario, threats against privacy can be categorized into three groups [22]: identity disclosure, link disclosure, and attribute disclosure. All these threats should be considered in a distributed link prediction algorithm, since it involves privacy-sensitive databases from multiple parties.

One promising solution for this privacy concern is cryptography to achieve distributed link prediction in a privacy-preserving way. Thus, in this paper, our goal is to develop a cryptographic solution for privacy-preserving distributed link prediction between multiple graph databases. We propose a solution based on private set intersection (PSI) to tackle this problem by considering both the efficiency of the solution and its privacy.

Via evaluations, we show that this solution provides good efficiency. For example, it can run in under 1 s (ignoring communication latency) for graphs based on a Flickr dataset with 40K nodes.

The proposed protocol does not provide perfect privacy (it leaks some intermediary values) and so we quantify this leakage to better understand if it is consequential enough to move to a fully private solution (which we also sketch).

1.1 Use Cases

Privacy-preserving distributed link prediction can be utilized in different settings. Here, we explain some of the possible applications.

Social Networks. In this setting, there are two social networks, Graph 1 and Graph 2. Graph 1 aims to understand whether there will be a link formed between nodes x and y by also utilizing the similarity of x and y in Graph 2, as distributed link prediction provides better accuracy compared to performing this operation locally.

E-commerce. Another application can be between a social network and an e-commerce service. In the previous use case, the link between two users is the main concern of the protocol. Unlike the previous case, here the links between a user and products are determined at the end of the protocol. In the e-commerce graph, there are links between the users and the products that they have bought. The aim here is to provide better advertising to users. The network will recommend product n to the user x if this user's friends also purchased the same product. For this purpose, the e-commerce network has to know the friends of user x in the social network. Unlike the previous use case, here link prediction cannot be done locally on the e-commerce graph, as the knowledge of the social network's structure should be utilized in order to do the recommendation.

Telecommunication. In this use case, an advertising company wants to propagate an advertisement in the telecom network. If user x is a target for that advertisement, the company would like to know which nodes are likely to form links with user x, in order to decide which nodes it will send the advertisement. The aim is to maximize the number of nodes that learn about the advertisement. Another application involves a social network graph and a phone operator graph. The phone operator wants to find out friends of user x in the social network, so that it offers the special services (e.g., discounts) to the users that are similar to user x.

Bioinformatics. Here, the first graph consists of patients and diseases and the aim is to predict the link between the patient i and the disease j. In the second graph, there are similar patients to patient i. Using these similar patients, and their connection to disease j, the link between patient i and disease j can be inferred.

1.2 Related Work

There is a rich literature on link prediction algorithms (without consideration of privacy) in a variety of network structures: multiple partially aligned social networks [25]; coupled networks [11]; and heterogenous networks [21]. Other works consider node similarity when two nodes in the graph do not share common neighbours [17]; unbalanced, sparse data across multiple heterogeneous networks [10]; missing link prediction using local random walk [19]; and the intersection of link prediction and transfer learning [24].

Other research considers the use of cryptography for collaborating on graph-based data between two parties with privacy protections. However such works consider problems other than link prediction: merging and query performed on knowledge graphs owned by different parties [6]; whether one graph is a subgraph of the other graph [23]; single-source shortest distance and all-pairs shortest distance both in sparse and dense graphs [1]; all pairs shortest distance and single source shortest distance [4]; and transitive closure [14]; anonymous invitation-based system [2] and its extension to malicious adversarial model [3]. While it may be possible to transform some of these into finding common neighbours with a black-box approach, we provide a purpose-built protocol for common neighbour. Later in Sect. 2.3, we review potential cryptographic building blocks in the literature.

Table 1. Different similarity metrics in a graph.

Similarity metric	Definition
Common neighbours	$\lvert\varGamma(x)\cap\varGamma(y)\rvert$
Jaccard's coefficient	$\frac{\lvert\varGamma(x)\cap\varGamma(y)\rvert}{\lvert\varGamma(x)\cup\varGamma(y)\rvert}$
Adamic/Adar	$\sum_{z\epsilon\lvert\varGamma(x)\cap\varGamma(y)\rvert}\frac{1}{log(\lvert\varGamma(z)\rvert)}$
$Katz_{\beta}$	$\sum_{l=1}^{\infty}\beta^{l}.\lvert path_{x,y}^{\langle l\rangle}\rvert$ where $path_{x,y}^{\langle l\rangle}$:= {paths of length exactly l from x to y} weighted: $path_{x,y}^{\langle l\rangle}$:= weight of the edge between x and y unweighted: $path_{x,y}^{\langle l\rangle}$:= 1 iff x and y are 1-hop neighbours The weight is determined by the constant value β

2 Proposed Solution

2.1 Building Blocks from Data Mining

Link Prediction. Given a snapshot of a graph at time t, link prediction algorithms aim to accurately predict the edges that will be added to the graph during the interval from time t to a given future time t' [18].

Similarity Metrics. In Table 1, different metrics for calculating proximity are given. Common neighbours, Jaccard coefficient and Adamic-Adar index are regarded as the node-dependent indices and they only require the information about node degree and the nearest neighbourhood, whereas the Katz index is defined as path-dependent index that consider the global knowledge of the network topology [19]. While there are also other metrics that are used (some of which are shown in Table 1), we choose common neighbours, as it is one of the widely-used methods for link prediction.

Common Neighbours. Common neighbours is used to predict the existence of a link between two nodes based on the number of their common neighbours. If two nodes share common neighbours, it is more likely that they will be connected in the future. In a local graph, the result of the metric can directly be computed by determining the intersection of the neighbour sets of two nodes. Based on the cardinality of the set, the network decides whether to suggest a link between these two nodes. The cardinality is defined as

$$\text{common neighbours} = \lvert\varGamma(x)\cap\varGamma(y)\rvert,$$

where $\varGamma(x)$ and $\varGamma(y)$ are the set of neighbours of nodes x and y respectively.

Adapting Common Neighbours for Two Parties. For the use case in this paper, we look at the problem of computing common neighbours metric across two different graphs owned by different entities. For example, this could be two separate social networks, or a social network with an e-commerce network. Graph

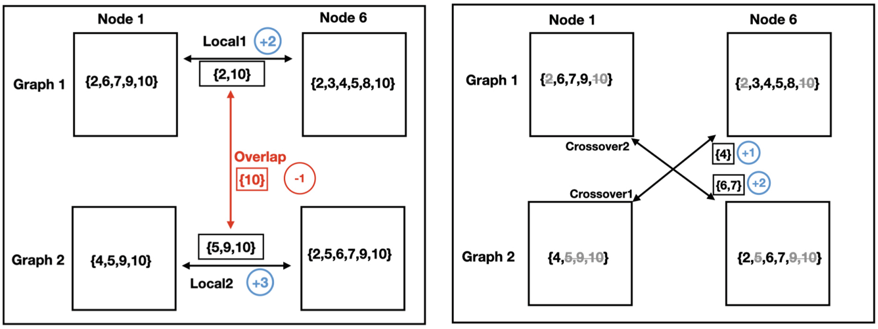


Fig. 1. An example computation between Graph 1 and Graph 2 to find the number of common neighbours of nodes 1 and 6 in their joint graph. Our contribution is to perform this computation in a privacy-preserving manner.

1 wants to perform link prediction between the nodes x and y by using the common neighbours information from Graph 2. CN denotes the total number of common neighbours that will be determined at the end of the protocol. We propose the following computation for two graphs, taking care to not double count any common neighbours:

$$\text{CN} = \text{local1} + \text{local2} + \text{crossover1} + \text{crossover2} - \text{overlap}$$

The variables are as follows:

- local1: number of common neighbours of node x and node y in Graph 1
- local2: number of common neighbours of node x and node y in Graph 2
- crossover1: number of common neighbours of node x from Graph 1 and node y from Graph 2
- crossover2: number of common neighbours of node y from Graph 1 and node x from Graph 2
- overlap: intersection of local1 and local2

Figure 1 illustrates an example of how CN is computed using the neighbours sets of both Graph 1 and Graph 2 (based on the graphs in Fig. 6). Graph 1 decides whether to suggest a link between nodes 1 and 6 based on this cardinality.

2.2 System Model

In our setting, there are two parties: Graph 1 and Graph 2, each having a graph structured network. Graph 1 wants to determine whether to suggest a link between the nodes x and y, not by only determining the common neighbours using its own graph, but also utilizing the graph structure of Graph 2. Graph 1 and Graph 2 compute common neighbours on their joint graphs without disclosing their respective graph structures. The result (number of common neighbours) is provided only to Graph 1, however the protocol can be run twice if

Table 2. PSI-based and Non-PSI based cryptographic building blocks

PSI based	PSI [9]	**Complexity:** Protocol complexity is linear in the sizes of the two sets. Both the client and the server performs exponentiations and modular multiplications. **Info leaked:** Intersection cardinality, no third party **Security setting:** semi-honest
	Delegated PSI [12]	**Complexity:** Computation and communication complexity of the protocol is linear in the size of the smaller set. For polynomial interpolation field operations are performed. Cloud server has to evaluate oblivious distributed key PRF instances, and unpack messages. The waiting time of packing messages by the backend server is the main computation cost **Info leaked:** Intersection cardinality, uses third party **Security setting:** semi-honest
	PSI with FHE [8]	**Complexity:** Communication overhead is logarithmic in the larger set size and linear in the smaller set size. While FHE is asymptotically efficient, it isn't in practice **Info leaked:** Input sizes and bit string length of the sets, no third party **Security setting:** Semi-honest
	PSI with OT [20]	**Complexity:** The circuit-based PSI protocol has linear communication complexity **Info leaked:** No info leaked as the result of a function on the intersection cardinality is the output, no third party **Security setting:** semi-honest
	Labeled PSI with FHE in malicious setting [7]	**Complexity:** Communication overhead is logarithmic in the larger set size and linear in the smaller set size. While FHE is asymptotically efficient, it isn't in practice **Info leaked:** No info is leaked, as the output is secret shared, no third party **Security setting:** Malicious
Non-PSI based	Privacy-preserving integer comparison [15] over each pair	**Complexity:** Privacy-preserving integer comparison protocol is run between every pair of nodes in the adjacency matrix created using the neighbour list from both graphs. The comparison protocol performs encryption, partial decryption, modular exponentiation, and multiplications **Info leaked:** No info is leaked, no third party **Security setting:** Malicious if ZKP added

Graph 2 also wants the result (otherwise, we assume Graph 1 is paying Graph 2 for this service). While creating our scheme, we make the following assumptions:

1. The identifiers in both graphs for the same nodes match. The graphs should prepare for the computation by sharing a schema and agreeing on unique identifiers (*e.g.*, an email address or phone number for human users).

2. Both graphs know the identity of the nodes for which the computation is being performed. In other words, edges involving these nodes are hidden, as well as all other edges and nodes.
3. If x and y are direct neighbours in Graph 1, Graph 1 has no need for the computation.
4. If x and y are direct neighbours in Graph 2, Graph 2 will halt before doing the computation and inform Graph 1. In this case, Graph 1 discovers a hidden link between x and y, which is stronger for prediction than the number of common neighbours.

Threat Model. The common public input to the computation will be the identifiers of two nodes known to Graph 1 and Graph 2. The private input of Graph 1 and Graph 2, respectively, is an assertion of their graph data. We assume Graph 1 and Graph 2 honestly input their correct data. This is a common assumption and resolving it involves having the data authenticated outside of the protocol, which is not a natural assumption for our use-cases. The second question is whether we can assume Graph 1 and Graph 2 follow the protocol correctly (semi-honest model) or exhibit arbitrary behaviour (malicious model). Given the strong assumption of data input, we find it natural to fit it to a semi-honest model of the protocol.

With these assumptions, we design the protocol so that Graph 2 learns nothing about Graph 1 other than the common input. On the other hand, Graph 1 learns the number of common neighbours on the joint set, which is the output of the multiparty computation (MPC). A fundamental limitation of MPC is that the output itself can leak information about the input. For example, if Graph 1 is malicious and is able to repeat this protocol many times with Graph 2, it can slowly reconstruct Graph 2's input by adaptively providing different inputs each time. For the purposes of this paper, we assume Graph 1 will not do this, because it is semi-honest, and further Graph 2 would not entertain so many executions of the protocol.

As an artifact of our protocol, Graph 1 also learns the intermediate values to compute the number of common neighbours: local1 + local2 + crossover1 + crossover2 - overlap. This extra information does allow a malicious Graph 1 to reconstruct Graph 2 with fewer queries, but in Sect. 3.3 we show that the impact of the leakage is immaterial. This can be prevented with heavier cryptography (Sect. 4.1). In addition, Graph 2 can force Graph 1 to compute the wrong result only if it behaves maliciously. In conclusion, our threat model provides reasonable privacy protection while being lightweight enough to be practical, and we suggest it represents a useful compromise for many real-world applications.

2.3 Building Blocks from Cryptography

Private set intersection (PSI) is a two-party cryptographic protocol that allows two entities, each with a set of data, to learn the intersection of their data without either learning any information about data that is outside the intersection [13]. After a detailed investigation of PSI variants and other related primitives, we

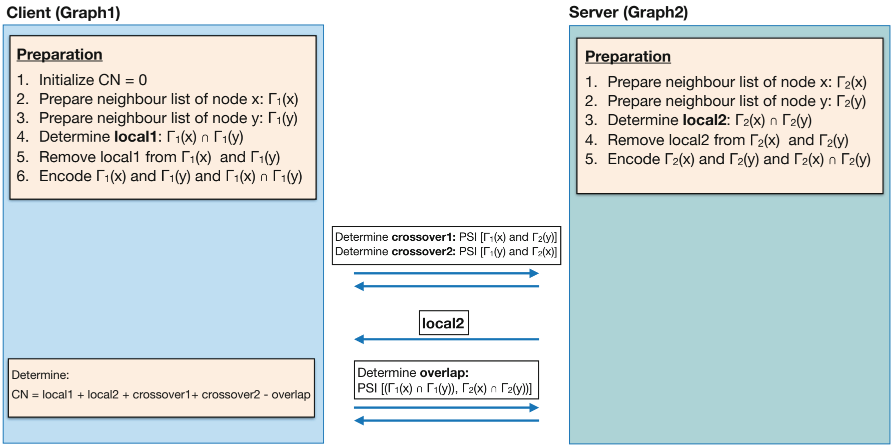

Client (Graph1)
Preparation
1. Initialize CN = 0
2. Prepare neighbour list of node x: $\Gamma_1(x)$
3. Prepare neighbour list of node y: $\Gamma_1(y)$
4. Determine **local1**: $\Gamma_1(x) \cap \Gamma_1(y)$
5. Remove local1 from $\Gamma_1(x)$ and $\Gamma_1(y)$
6. Encode $\Gamma_1(x)$ and $\Gamma_1(y)$ and $\Gamma_1(x) \cap \Gamma_1(y)$
Determine:
CN = local1 + local2 + crossover1+ crossover2 - overlap
Determine **crossover1:** PSI [$\Gamma_1(x)$ and $\Gamma_2(y)$]
Determine **crossover2:** PSI [$\Gamma_1(y)$ and $\Gamma_2(x)$]
local2
Determine **overlap:**
PSI [($\Gamma_1(x) \cap \Gamma_1(y)$), $\Gamma_2(x) \cap \Gamma_2(y)$)]
Server (Graph2)
Preparation
1. Prepare neighbour list of node x: $\Gamma_2(x)$
2. Prepare neighbour list of node y: $\Gamma_2(y)$
3. Determine **local2**: $\Gamma_2(x) \cap \Gamma_2(y)$
4. Remove local2 from $\Gamma_2(x)$ and $\Gamma_2(y)$
5. Encode $\Gamma_2(x)$ and $\Gamma_2(y)$ and $\Gamma_2(x) \cap \Gamma_2(y)$

Fig. 2. Overview of the proposed PSI-based solution. PSI is called three times in the protocol for determining crossover1, crossover2 and overlap. PSI itself is described in Fig. 3

choose [9] as the core scheme to deploy for our link prediction. Our scenario requires a scheme that calculates only the cardinality (sometimes called PSI-CA) of the intersection of the two sets in an efficient and scalable way with minimum information leakage with no third party's assistance. A security model for semi-honest adversaries is sufficient, and we leave additional (stronger) security guarantees for future work. A summary of relevant cryptographic primitives is given in Table 2 and we provide more details of each primitive in the full version of the paper[1].

2.4 Proposed Protocol

We use PSI scheme proposed in [9] to perform distributed link prediction between two graph databases. Figure 2 shows the interactive protocol between Graph 1 and Graph 2. Graph 1 wants to learn the common neighbour index to determine whether to suggest a link between the nodes x and y. Both Graph 1 and Graph 2 locally determine the neighbour sets of x and y (local1 and local2, respectively). In order to determine crossover1, crossover2, and overlap, Graph 1 and Graph 2 run three separate PSI protocols among themselves. Each PSI leaks a certain amount of information and we discuss this partial information leak in Sect. 3.3. At the end of the protocol, Graph 1 learns the exact cardinality of common neighbours of nodes x and y on the joint graph. Figure 3 shows the details of the PSI protocol for calculating crossover1 (the calculations for crossover2 and overlap are also the same). It is an interactive protocol between Graph 1 and Graph 2, with offline and online stages. At the offline stage, Graph 1 masks its set and Graph 2 masks its set and shuffles it. During the online stage, Graph 2

[1] Full paper.

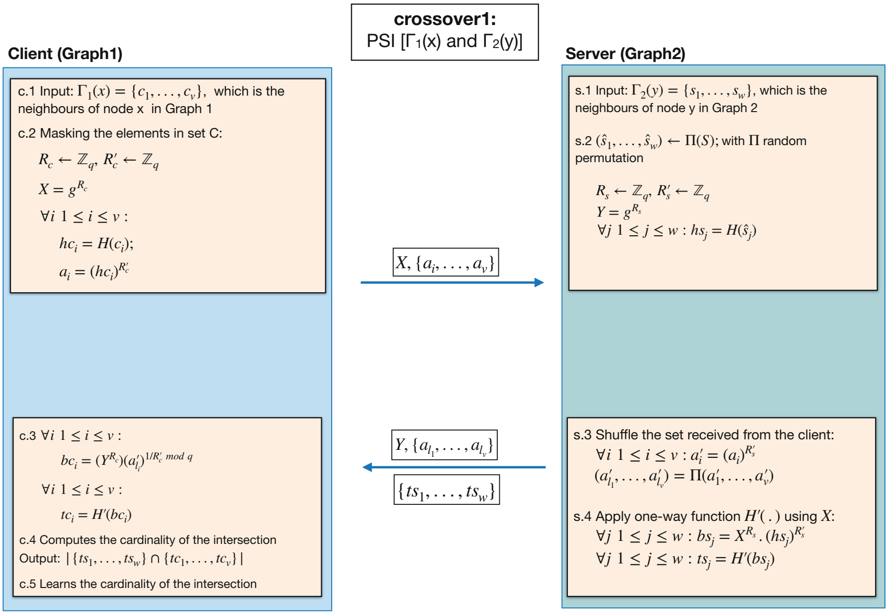

crossover1:
PSI [Γ_1(x) and Γ_2(y)]
Client (Graph1)
c.1 Input: $\Gamma_1(x) = \{c_1, \ldots, c_v\}$, which is the neighbours of node x in Graph 1
c.2 Masking the elements in set C:
$R_c \leftarrow \mathbb{Z}_q, R'_c \leftarrow \mathbb{Z}_q$
$X = g^{R_c}$
$\forall i \; 1 \leq i \leq v:$
$hc_i = H(c_i);$
$a_i = (hc_i)^{R'_c}$
c.3 $\forall i \; 1 \leq i \leq v:$
$bc_i = (Y^{R_c})(a'_{l_i})^{1/R'_c \bmod q}$
$\forall i \; 1 \leq i \leq v:$
$tc_i = H'(bc_i)$
c.4 Computes the cardinality of the intersection
Output: $|\{ts_1, \ldots, ts_w\} \cap \{tc_1, \ldots, tc_v\}|$
c.5 Learns the cardinality of the intersection
$X, \{a_i, \ldots, a_v\}$
$Y, \{a_{l_1}, \ldots, a_{l_v}\}$
$\{ts_1, \ldots, ts_w\}$
Server (Graph2)
s.1 Input: $\Gamma_2(y) = \{s_1, \ldots, s_w\}$, which is the neighbours of node y in Graph 2
s.2 $(\hat{s}_1, \ldots, \hat{s}_w) \leftarrow \Pi(S)$; with Π random permutation
$R_s \leftarrow \mathbb{Z}_q, R'_s \leftarrow \mathbb{Z}_q$
$Y = g^{R_s}$
$\forall j \; 1 \leq j \leq w : hs_j = H(\hat{s}_j)$
s.3 Shuffle the set received from the client:
$\forall i \; 1 \leq i \leq v : a'_i = (a_i)^{R'_s}$
$(a'_{l_1}, \ldots, a'_{l_v}) = \Pi(a'_1, \ldots, a'_v)$
s.4 Apply one-way function $H'(\,.\,)$ using X:
$\forall j \; 1 \leq j \leq w : bs_j = X^{R_s}.(hs_j)^{R'_s}$
$\forall j \; 1 \leq j \leq w : ts_j = H'(bs_j)$

Fig. 3. PSI protocol for determining crossover1 (adapted from [9])

receives the masked set of Graph 1, masks it with its own randomness and shuffles it. When Graph 1 receives the sets, it removes the randomness and determines the intersection of two sets. PSI is described in the multiplicative subgroup $\mathbb{G}_q$ of $\mathbb{Z}_p^*$, where p and q are large primes, $q \mid p - 1$ and $g \in \mathbb{G}_q$ is the generator. $|p| = 1024$ bits and $|q| = 160$ bits. This is for experimental purposes only, these parameters should be at least doubled in length to meet the current, accepted security level[2]. H and H′ are hash functions that are modeled as random oracles.

3 Evaluation

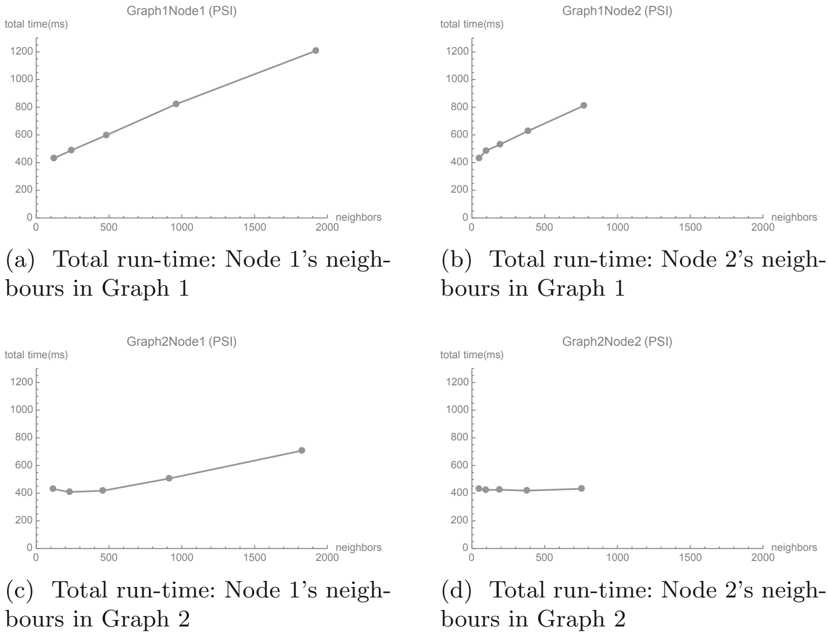
3.1 Performance

We implemented the proposed distributed link prediction algorithm and evaluated it considering different aspects. We used the implementation of PSI defined in [9] where q and p are 160 and 1024 bits, respectively. We ran our experiments on macOS High Sierra, 2.3 GHz Intel Core i5, 8 GB RAM, and 256 GB hard disk. We ran each experiment for 20 times and reported the average.

We used the Flickr dataset in our experiments and using the SALab tool[3], we generated two graphs based on Flickr. Node and edge similarities are set to 0.5. Graph 1 and Graph 2 has 37377 and 37374 nodes; 1886280 and 1900553 edges

[2] NIST Special Publication 800-131A Revision 2.

[3] GitHub: SALab.



(a) Total run-time: Node 1's neighbours in Graph 1

(b) Total run-time: Node 2's neighbours in Graph 1

(c) Total run-time: Node 1's neighbours in Graph 2

(d) Total run-time: Node 2's neighbours in Graph 2

Fig. 4. Total run-time (in milliseconds) of common neighbour on joint graph according to varying sizes of neighbours for Node 1 and Node 2.

respectively. We picked two random nodes, determined their neighbour sets in each graph and ran our experiments using them. In Graph 1, node 1 has 120 neighbours; 48 for node 2 in graph 1; 114 for node 1 in graph 2; and 47 for node 2 in graph 2. Figure 4 shows the total run-time of the common neighbour protocol on the joint graph if we vary the size of neighbours for Node 1 and Node 2 in both graphs.

3.2 Utility of the Protocol

We illustrate the additional common neighbour information gained by a graph when it collaborates with a second graph. In our first experiment, we created two graphs with the same Barabasi-Albert Distribution where 22 edges are added at each step. Both graphs contain 4039 nodes. Average number of common neighbours for each pair is 0.9 in Graph 1. When we consider the merged graph of two networks, average number of common neighbours for each pair increases to 3.3. This shows that distributed link prediction provides significantly more accurate results (compared to local link prediction) and is worth pursuing.

In our second experiment, we created two graphs with the same number of nodes. Both Graph 1 and Graph 2 have 200 nodes. Graph 1 is created with Barabasi-Albert Distribution, where 22 edges are added at each step. Graph 2 is also created in the same setting and, we computed the average number of

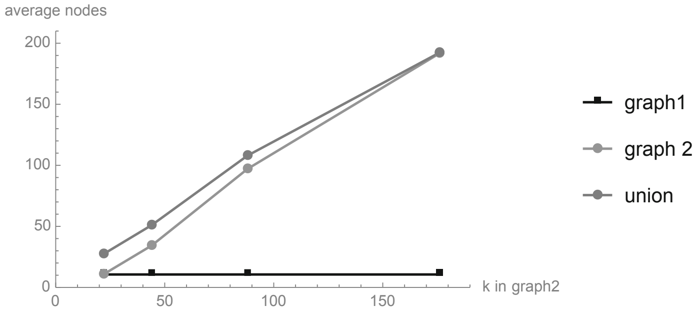


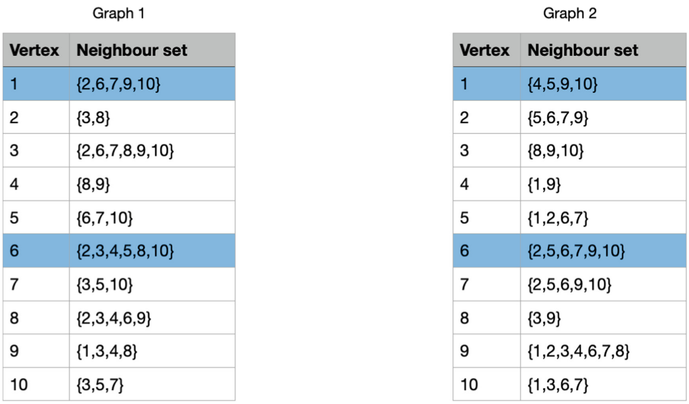
Fig. 5. The change in the average number of common neighbours of two pairs according to connectedness of Graph2. k is the number of edges added at each step in Barabasi-Albert distribution. As k increases in Graph 2, Graph 1 benefits more and more from performing the protocol with Graph 2.

neighbours of each pair in the union and in Graph 2, with increasing values of k. In this setting, as Graph 2 becomes more connected, it benefits less from the distributed link prediction, as the connectedness of Graph 2 becomes more similar to the union. This is shown in Fig. 5.

3.3 Security

The privacy and integrity of our proposed solution is largely subsumed by the security of the underlying PSI protocol [9]. This protocol is shown to be secure under the decisional Diffie-Hellman problem in an appropriate group with semi-honest adversaries. The security proof is in the random oracle model. We make three sequential calls to the protocol. While (universal) composability of the PSI protocol is left by the authors for future work, there is no obvious issue with running the protocol multiple times. For safety, Graph 1 can wait for the first PSI to finish before starting the second one. The PSI protocol leaks (an upper-bound) on the size of each party's graph, and its output is the cardinality of the intersection. Our protocol, with the three PSI calls, leaks the cardinality of four intermediary values (local2, crossover1, crossover2, and overlap) in computing the common neighbours. We now quantify the impact of this leakage on what Graph 1 can learn about Graph 2 beyond the number of common neighbours.

Leakage of Partial Information. In our setting, there are three different categories of threats to privacy: identity disclosure, link disclosure, and attribute disclosure. As PSI does not leak the nodes in the set intersection, we do not learn about the identities or the attributes related to the nodes. In PSI [9], the cardinality of intersection leaks information about the possible combination of nodes in the intersection set. Graph 1, who learns the size of the intersection set, can compute these combinations (which corresponds to link disclosure). The only time Graph 1 learns the identity of a node (which means that the node is



Graph 1

Vertex	Neighbour set
1	{2,6,7,9,10}
2	{3,8}
3	{2,6,7,8,9,10}
4	{8,9}
5	{6,7,10}
6	{2,3,4,5,8,10}
7	{3,5,10}
8	{2,3,4,6,9}
9	{1,3,4,8}
10	{3,5,7}

Graph 2

Vertex	Neighbour set
1	{4,5,9,10}
2	{5,6,7,9}
3	{8,9,10}
4	{1,9}
5	{1,2,6,7}
6	{2,5,6,7,9,10}
7	{2,5,6,9,10}
8	{3,9}
9	{1,2,3,4,6,7,8}
10	{1,3,6,7}

Fig. 6. Sample graphs with 10 nodes. We refer to this sample graph in Sect. 2.1 to explain how common neighbours are computed among two parties and in Sect. 3.3 to quantify the partial information leak.

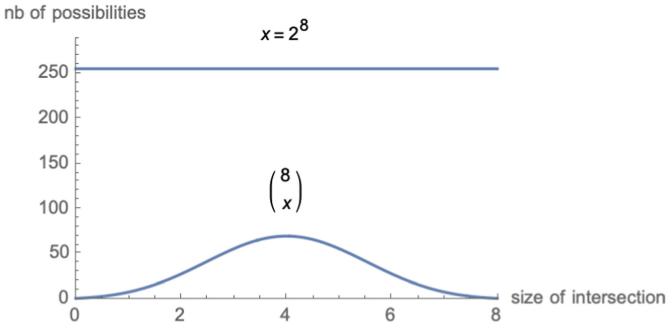


Fig. 7. Number of possible combinations of intersection set according to cardinality of intersection.

in the set that Graph 2 owns) is when Graph 1 has only one node in its set and the cardinality of the intersection it receives as the result of the PSI protocol is 1. Consider the case where Node 1 in Graph 1 has only the node 7 in the set, in Fig. 1. When Graph 1 learns that Crossover2 is 1, it can infer that 7 is connected to Node 6 in Graph 2. Therefore, Graph 1 learns the identity of one of the nodes in the neighbour set of Node 6 in Graph 2 and consequently, the link between 7 and Node 6.

We refer to the graphs in Fig. 6 in order to illustrate what type of information is leaked during the PSI protocol run between Graph 1 and Graph 2, each having 10 nodes. Graph 1 wants to utilize the graph structure of Graph 2 to decide whether to recommend a link between nodes 1 and 6. At the beginning of the protocol, Graph 2 sends the size of the common neighbours of node 1 and node 6 (which corresponds to local2 and its size is 3) to Graph 1. Hence, Graph 1 learns that there are $\binom{8}{3}$ possibilities for local2 as opposed to 2^8 (we choose from

8 nodes as we assume that 1 and 6 are not neighbours in both of the graphs). When we look at the end cases: (i) if the size of intersection is 0, Graph 1 does not learn any extra information (for node 1, there are 2^8 possibilities with the condition that for each possibility, node 1 and node 6 do not have any nodes in the intersection); and (ii) if the size of the intersection is 8, Graph 1 learns that nodes 1 and 6 are connected to all of the 8 nodes. Figure 7 illustrates the number of possibilities learned by Graph 1 for each size of the intersection. It also shows that even at the worst case, there is still a lot of information that is not learned by Graph 1.

For a graph generated using the Flickr data set that has 37377 nodes, the average number of neighbours of a node is 50. So, for two nodes with average number of neighbours, there are $\binom{37377}{50}$ possibilities for their intersection, which is a very large number.

4 Discussion

4.1 Strengthening the Privacy

Here, we discuss a solution based on additively homomorphic encryption (*e.g.,* exponential Elgamal or Paillier) that hides all partial information such that Graph 1 learns only the cardinality. At the beginning of the protocol which is shown in Fig. 8, Graph 1 determines the neighbour sets of nodes x and y, determines local1 and removes local1 from these sets. It encrypts each element in these sets and the cardinality of local1. Graph 2 performs the same steps for local2. In order to determine crossover1, an encrypted matrix is created: each encrypted element in the neighbour set of node x at Graph 1 is compared against all of the encrypted elements in the neighbour set of node y at Graph 2. The same is done for crossover2 between the neighbour set of y at Graph 1 and the neighbour set of node x at Graph 2. In order to compare two encrypted values, we adapt the protocol proposed in [15]. The comparison function takes two encrypted values and the output is either the encryption of 0 if these encrypted values are different or the encryption of 1 otherwise. The sum over all the elements in the matrix is determined using homomorphic addition both for crossover1 and crossover2. Overlap, which is the intersection of crossover1 and crossover2, is also determined in a similar way using privacy-preserving integer comparison. The final common neighbours cardinality (CN), which is encrypted, is determined as CN= local1 + local2 + crossover1 + crossover2 - overlap. Even though this approach strengthens privacy, it is significantly more costly compared our proposed solution based on PSI. We think for most applications, the efficiency is a larger concern than the partial leakage from our protocol, but it is possible that entities might prefer complete privacy for very sensitive data.

4.2 Complexity

Our protocol's complexity is linear in the sizes of the neighbour set of the nodes. In this paper, we discuss the setting for performing distributed link prediction

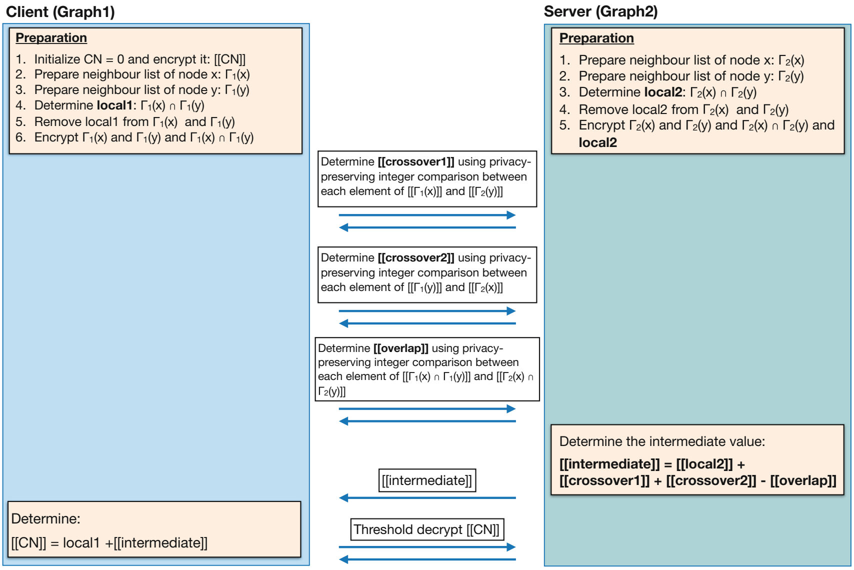

Client (Graph1)
Server (Graph2)
Preparation
1. Initialize CN = 0 and encrypt it: [[CN]]
2. Prepare neighbour list of node x: Γ1(x)
3. Prepare neighbour list of node y: Γ1(y)
4. Determine local1: Γ1(x) ∩ Γ1(y)
5. Remove local1 from Γ1(x) and Γ1(y)
6. Encrypt Γ1(x) and Γ1(y) and Γ1(x) ∩ Γ1(y)
Preparation
1. Prepare neighbour list of node x: Γ2(x)
2. Prepare neighbour list of node y: Γ2(y)
3. Determine local2: Γ2(x) ∩ Γ2(y)
4. Remove local2 from Γ2(x) and Γ2(y)
5. Encrypt Γ2(x) and Γ2(y) and Γ2(x) ∩ Γ2(y) and local2
Determine [[crossover1]] using privacy-preserving integer comparison between each element of [[Γ1(x)]] and [[Γ2(y)]]
Determine [[crossover2]] using privacy-preserving integer comparison between each element of [[Γ1(y)]] and [[Γ2(x)]]
Determine [[overlap]] using privacy-preserving integer comparison between each element of [[Γ1(x) ∩ Γ1(y)]] and [[Γ2(x) ∩ Γ2(y)]]
Determine the intermediate value:
[[intermediate]] = [[local2]] + [[crossover1]] + [[crossover2]] - [[overlap]]
[[intermediate]]
Determine:
[[CN]] = local1 +[[intermediate]]
Threshold decrypt [[CN]]

Fig. 8. Overview of the solution with stronger privacy. Note that $[[\Gamma_1(x)]]$ means that each element in the neighbour set of node x in Graph 1 is encrypted.

between two particular nodes in two networks. A network may want to expand this computation for every possible pair in its graph. The complexity depends on the size (which affects the number of possible pairs) and the density of the network (the sizes of neighbour sets affects PSI run-time). PSI runs in linear time complexity. Our protocol is for a specific pair of nodes. The number of pairs in a graph with n nodes is n^2, so running our protocol (or any protocol based on PSI) one an entire graph will require running a linear time operation on a quadratic number of nodes: thus, cubic time complexity in the worst-case. Reductions in this complexity (*e.g.*, based on heuristics for prioritizing which nodes to look at) is an interesting future work.

5 Concluding Remarks

For better accuracy, link prediction can be performed on merged graphs belonging to different parties. This leads to privacy concerns as parties do not want to reveal sensitive data related to their network structures. Therefore, in this work, we proposed a PSI-based, privacy preserving distributed link prediction scheme among two graph databases. In our current proposed scheme, Graph 2 learns among which nodes Graph 1 is computing the common neighbours metric. As a future work, a scheme that allows Graph 2 to hide the identities of these nodes from Graph 1 can be proposed. We might also explore if more recent PSI proposals [5,16] produce faster results.

Acknowledgements. We thank the reviewers who helped to improve our paper. J. Clark acknowledges support for this research project from the National Sciences and Engineering Research Council (NSERC), Raymond Chabot Grant Thornton, and Catallaxy Industrial Research Chair in Blockchain Technologies and NSERC through a Discovery Grant. E. Ayday acknowledges that research reported in this paper was partly supported by the National Science Foundation (NSF) under grant number NSF CCF 2200255 and Cisco Research University Funding grant number 2800379. M. Namazi acknowledges that this work was partially funded by the Spanish Government through grant RTI2018-095094-B-C22.

References

1. Anagreh, M., Laud, P., Vainikko, E.: Parallel privacy-preserving shortest path algorithms. Cryptography **5**(4), 27 (2021)
2. Boshrooyeh, S.T., Küpçü, A.: Inonymous: anonymous invitation-based system. In: Garcia-Alfaro, J., Navarro-Arribas, G., Hartenstein, H., Herrera-Joancomartí, J. (eds.) ESORICS/DPM/CBT -2017. LNCS, vol. 10436, pp. 219–235. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67816-0_13
3. Boshrooyeh, S.T., Küpçü, A., Özkasap, Ö.: Anonyma: anonymous invitation-only registration in malicious adversarial model. Cryptology ePrint Archive (2019)
4. Brickell, J., Shmatikov, V.: Privacy-preserving graph algorithms in the semi-honest model. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 236–252. Springer, Heidelberg (2005). https://doi.org/10.1007/11593447_13
5. Chandran, N., Gupta, D., Shah, A.: Circuit-psi with linear complexity via relaxed batch OPPRF. Cryptology ePrint Archive (2021)
6. Chen, C., Cui, J., Liu, G., Wu, J., Wang, L.: Survey and open problems in privacy preserving knowledge graph: merging, query, representation, completion and applications. arXiv preprint arXiv:2011.10180 (2020)
7. Chen, H., Huang, Z., Laine, K., Rindal, P.: Labeled psi from fully homomorphic encryption with malicious security. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1223–1237 (2018)
8. Chen, H., Laine, K., Rindal, P.: Fast private set intersection from homomorphic encryption. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1243–1255 (2017)
9. De Cristofaro, E., Gasti, P., Tsudik, G.: Fast and private computation of cardinality of set intersection and union. In: Pieprzyk, J., Sadeghi, A.-R., Manulis, M. (eds.) CANS 2012. LNCS, vol. 7712, pp. 218–231. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35404-5_17
10. Dong, Y., et al.: Link prediction and recommendation across heterogeneous social networks. In: 2012 IEEE 12th International Conference on Data Mining (ICDM), pp. 181–190. IEEE (2012)
11. Dong, Y., Zhang, J., Tang, J., Chawla, N.V., Wang, B.: CoupledLP: link prediction in coupled networks. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 199–208. ACM (2015)
12. Duong, T., Phan, D.H., Trieu, N.: Catalic: delegated PSI cardinality with applications to contact tracing. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 870–899. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64840-4_29

13. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_1
14. He, X., Vaidya, J., Shafiq, B., Adam, N., Terzi, E., Grandison, T.: Efficient privacy-preserving link discovery. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS (LNAI), vol. 5476, pp. 16–27. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01307-2_5
15. Hubaux, J.P., et al.: Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data. In: Proceedings of USENIX Security Workshop on Health Information Technologies (HealthTech 2013), number EPFL-CONF-187118 (2013)
16. Karakoç, F., Küpçü, A.: Linear complexity private set intersection for secure two-party protocols. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) CANS 2020. LNCS, vol. 12579, pp. 409–429. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-65411-5_20
17. Leicht, E.A., Holme, P., Newman, M.E.: Vertex similarity in networks. Phys. Rev. E **73**(2), 026120 (2006)
18. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. Am. Soc. Inform. Sci. Technol. **58**(7), 1019–1031 (2007)
19. Liu, W., Lü, L.: Link prediction based on local random walk. EPL (Europhys. Lett.) **89**(5), 58007 (2010)
20. Pinkas, B., Schneider, T., Tkachenko, O., Yanai, A.: Efficient circuit-based PSI with linear communication. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11478, pp. 122–153. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17659-4_5
21. Tang, J., Lou, T., Kleinberg, J., Wu, S.: Transfer learning to infer social ties across heterogeneous networks. ACM Trans. Inf. Syst. (TOIS) **34**(2), 1–43 (2016)
22. Wu, X., Ying, X., Liu, K., Chen, L.: A survey of privacy-preservation of graphs and social networks. In: Aggarwal, C., Wang, H. (eds.) Managing and Mining Graph Data. ADBS, vol. 40, pp. 421–453. Springer, Boston (2010). https://doi.org/10.1007/978-1-4419-6045-0_14
23. Xu, Z., Zhou, F., Li, Y., Xu, J., Wang, Q.: Privacy-preserving subgraph matching protocol for two parties. Int. J. Found. Comput. Sci. **30**(04), 571–588 (2019)
24. Yu, K., Chu, W.: Gaussian process models for link analysis and transfer learning. In: Advances in Neural Information Processing Systems, pp. 1657–1664 (2008)
25. Zhang, J., Yu, P.S., Zhou, Z.H.: Meta-path based multi-network collective link prediction. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1286–1295. ACM (2014)