Social Network De-Anonymization and Privacy Inference with Knowledge Graph Model

Jianwei Qian, Xiang-Yang Li, *Fellow, IEEE,* Chunhong Zhang, Linlin Chen, Taeho Jung, *Student Member, IEEE,* Junze Han

Abstract—Social network data is widely shared, transferred and published for research purposes and business interests, but it has raised much concern on users' privacy. Even though users' identity information is always removed, attackers can still de-anonymize users with the help of auxiliary information. To protect against de-anonymization attack, various privacy protection techniques for social networks have been proposed. However, most existing approaches assume *specific* and restrict network structure as background knowledge and ignore semantic level prior belief of attackers, which are not always realistic in practice and do not apply to arbitrary privacy scenarios. Moreover, the privacy inference attack in the presence of semantic background knowledge is barely investigated. To address these shortcomings, in this work, we introduce knowledge graphs to explicitly express arbitrary prior belief of the attacker for any individual user. The processes of de-anonymization and privacy inference are accordingly formulated based on knowledge graphs. Our experiment on data of real social networks shows that knowledge graphs can power de-anonymization and inference attacks, and thus increase the risk of privacy disclosure. This suggests the validity of knowledge graphs as a general effective model of attackers' background knowledge for social network attack and privacy preservation.

Index Terms—Social network data publishing, attack and privacy preservation, knowledge graph.

1 Introduction

Many online social networking sites like Facebook and Flickr have been generating tons of data every day, including users' profiles, relations and personal life details. Social network data can be released to third-parties for various purposes including targeted advertising, developing new applications, academic research, and public competition [16], [26], [48], [56]. However, publishing social network data could also result in privacy leakage and thus raise great concerns among the public. Naively removing user IDs before publishing the data is far from enough to protect users' privacy [6], [36].

The privacy issue in network data publishing is attracting increasing attention from researchers and social network providers [6], [24], [33]. Various privacy attack and protection techniques have been proposed, including k-anonymity [50] based techniques (*e.g.*, k-degree anonymity [33]), and graph mapping based de-anonymization (*e.g.*, [24]). Unfortunately, previous works have three main limitations. **First**, most of the prior attacks only focus on de-anonymization (also referred to as re-identification), that is, mapping a node in the network with a real person. Yet how the attacker acquires and infers users' privacy after de-anonymization was barely discussed before. **Second**, previous works have specific assumptions about the attacker's prior knowledge (also referred to as background information and we will use them interchangeably hereafter). Some assumes the attacker is weak and only has a specific type of information, such as node degrees [33]. Others assume that the attacker possesses a pure topological network (without user profiles) which overlaps with the published network data, such as [24]. The attacker is often assumed to be 100 percent sure of her prior knowledge. Conversely, some, if not much, of the attacker's knowledge is probabilistic in the real world. Under such ideal assumptions, the data anonymization methods proposed are unable to defend against arbitrary attackers who may possesses a large variety of knowledge of the auxiliary information about the target users. **Third**, they typically do not consider the scenario that the attacker could exploit correlations among attributes to make inference about users' sensitive attributes. Actually, the attacker can not only read users' attributes directly from the published data, but also infer some attributes according to others. Take salary as an example, it is correlated with multiple attributes including gender, education and occupation, *e.g.*, working as a doctor strongly implies high salary.

To overcome these limitations, our goal is to construct a comprehensive and realistic model of the attacker's knowledge and use this model to depict the privacy inferring process. Hopefully, our model can represent attackers that are more real than those in some relate work which are assumed to have only a limited knowledge of the target victims. The significance of our work lies in providing a better understanding of the attacker's prior knowledge and privacy inference, alerting social network publishers to the serious privacy leakage risks in reality, and leaving implications to researchers for designing stronger privacy protection (*e.g.*, anonymization) techniques without underestimating the attacker's capabilities.

We are facing three challenges. **First**, it is hard to build such an expressive model that covers all of the attacker's prior knowledge, given that she may have various knowledge, varying from node profiles and degrees, to link relations and neighborhood

The research of Li is partially supported by China National Funds for Distinguished Young Scientists with No. 61625205, Key Research Program of Frontier Sciences, CAS, No. QYZDY-SSW-JSC002, NSFC with No. 61520106007, NSF ECCS-1247944, NSF CMMI 1436786, and NSF CNS 1526638.

J. Qian, L. Chen, T. Jung and J. Han are with the Department of Computer Science, Illinois Institute of Technology, Chicago, 60616 USA.
X.Y. Li is with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China.
C. Zhang is with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China.
X.Y. Li is the contact author.

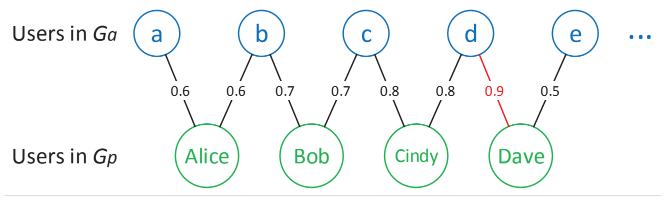


Fig. 1: **De-anonymization with background knowledge:** Although 2-anonymity is satisfied, users can still be de-anonymized by using background knowledge and solving maximum weighted bipartite matching.

subgraphs (called structural property), some of which are even probabilistic. **Second**, it is difficult to model the privacy inference steps, since the attacker may have various capabilities and techniques. She could be either computationally powerful or weak. She may have knowledge of some correlations among attributes, which are learned from information sources or by data mining. She may design her own algorithms to make inference with her prior knowledge and computation power. **Third**, it is challenging to quantify privacy disclosure. There has been no explicit and unified definition of privacy yet, let alone privacy disclosure's quantification.

In this paper, we will model the attacker's prior knowledge using knowledge graphs [21] and use them to express the two attacks stages – de-anonymization and privacy inference. We transform the problem of de-anonymizing a group of people into the maximum weighted bipartite matching problem. A simple example is given in Fig. 1, where the target real people (green circles) are connected to their candidate nodes (blue circle) in the published graph that are similar to them. The weights on the links represents the similarity scores. Solving the matching problem will contribute to the best de-anonymization. The details will be presented in Section 4.

Fig. 1 also reveals that previous k-anonymity based anonymization approaches are vulnerable to attacks when the attacker has more knowledge than assumed. For instance, suppose the published graph satisfies 2-anonymity. So for each target person, there are two nodes in the graph both of which are the possible match of this person. However, somehow the attacker gets to know that node d is very likely to be Dave by using her background knowledge (so she sets the weight of "d-Dave" to a greater value, say 0.9). By finding the maximum matching of the bipartite, all four users will be correctly de-anonymized, even though 2-anonymity is completely satisfied for Alice, Bob, and Cindy. This shows that de-anonymizing one user may induce a chain reaction in de-anonymizing other users. Therefore, it is indicated that k-anonymity based approaches are not resistant to the background knowledge attack.

In the knowledge graph, a confidence score is attached to each piece of knowledge to reflect the extent of the attacker's belief in it. During the second attack stage, the attacker makes inferences of users' private attributes and updates these confidence scores. Finally we can use the variation of the confidence scores to inflect the attacker's information gain and to quantify privacy disclosure. We will illustrate our solution in detail later.

In general, our contributions can be summarized as follows.

1) To the best of our knowledge, we are the first to apply knowledge graphs to model the attacker's background knowledge in social network data publishing. This is shown to be more realistic and complete than previous attacker models. (Section 2)

2) We utilize this model to depict the privacy inferring process,

$G(\mathcal{V}, \mathcal{E})$	Original graph
G_a	Anonymized graph
G_p	Prior attack graph
G_q	Posterior attack graph
$\mathcal{V}^U$	User node set
$\mathcal{V}^A$	Attribute node set
$\mathcal{E}^{UU}$	User-to-user links
$\mathcal{E}^{UA}$	User-to-attribute links
$\mathcal{E}^{AA}$	Attribute-to-attribute links
n_a	Node number of G_a
n_p	Node number of G_p
a	A user node in G_a
p	A user node in G_p
$t = \langle s, p, o \rangle$	A triple with subject s, predicate p, and object o
$c, c(t)$	Confidence score of a triple/link
$S_A(p, a)$	Attribute similarity of two user nodes p, a
$S_R(p, a)$	Relational/Structural similarity of two user nodes p, a
$S(p, a)$	Node similarity of two users p, a
$\hat{S}(p, a)$	Neighborhood similarity of two users p, a
$Sim(G_p, G_a)$	Similarity of two graphs G_p, G_a

TABLE 1: Frequently used symbols

i.e., how an attacker de-anonymizes and infers users' private attributes, which helps to determine and measure privacy disclosure. (Section 3)

3) On this model, we propose a de-anonymization scheme, for which two heuristics are designed, and we also present an approach to privacy inference. (Section 4)

4) We present an experiment of our attack on two real-life social network datasets. The extensive evaluation shows that our attack technique is realistic and powerful. For example, by knowing the noisy information of 0.5% of the total users in a social network, the attacker can successfully de-anonymize over 60% of them in our study, *i.e.*, matching correctly 60% of the users in the prior knowledge graph to nodes in the anonymized graph. (Section 5)

The major additional contribution of this paper over our conference paper [43] is that we design another heuristic method (LSH based de-anonymization in Section 4.4) and presented an evaluation for it on the two datasets.

2 Preliminaries and Modeling

We here present needed background on knowledge graph, data model, attack model, the attacker's knowledge model, privacy concepts, and locality-sensitive hashing.

2.1 Network Data

Network data refers to data that has a structure of graph, of which the most important is social network data. It describes entities (often people) and the relationships between them. Social networking sites and instant-messaging programs allow users to explicitly define such "friend" or "follow" relationships. Making such data available can be invaluable to social studies and understand the dynamics of communities. However, the release of data is severely restrained due to concerns about the privacy of individuals.

2.2 Knowledge Graph

A knowledge graph (KG) [21] is a network of all kinds of entities related to a specific domain or topic. The entities are not limited to real objects and abstract concepts, but instances of numbers, datasets and documents. It is a directed graph where a node represents an entity, a directed link relates one entity to another, and each link is associated with a predicate that represents the relationship. Each link together with its two endpoints in this graph stands for a piece of knowledge.

Knowledge graphs are usually stored in the form of RDF triples, *i.e.*, ⟨subject, predicate/relation, object⟩, each representing a link. Note multiple links between two nodes are allowed, since there could be multiple relations between two entities. For example, $\langle$Tom Cruise, was born in, the USA$\rangle$, $\langle$Tom Cruise, has nationality, the USA$\rangle$. Actually, if the knowledge graph contains the triple $\langle s, r, o\rangle$, it can still contain $\langle s', r, o\rangle$, $\langle s, r', o\rangle$, $\langle s, r, o'\rangle$ [15]. Formally, a knowledge graph is denoted as $G = (\mathcal{V}, \mathcal{E})$, with a mapping $\tau : \mathcal{V} \to \mathcal{C}$ that maps an entity (a subject or object) to a category, and a mapping $\rho : \mathcal{E} \to \mathcal{R}$ that maps a link to a relational type. Existing large-scale knowledge graphs include Freebase, YAGO, Knowledge Vault, Google Knowledge Graph, DBpedia, etc. They are widely applied to improving web search and question answering services. Following Knowledge Vault [15], each link/triple is assigned a confidence score in our methods, which implies the probability of this knowledge being true.

2.3 Social Network Data Model

A typical social network dataset is comprised of the structural/topological data and users' profiles. In the literature, a social network is usually represented as a graph where nodes stand for users and links stand for users' relations. In this paper, however, we use a different notation. We model a social network with a knowledge graph $G(\mathcal{V}, \mathcal{E})$, in which $\mathcal{V}$ is a set of nodes corresponding to entities of the network, and $\mathcal{E}$ is a set of links corresponding to relations between the entities. The difference is that nodes can stand for any type of entities, including users **and** their attributes such as genders, locations, numbers, etc. They are referred to as *user nodes* $\mathcal{V}^U$ and *attribute nodes* $\mathcal{V}^A$ respectively. We would use node for short to refer to user node when there is no ambiguity. Thus, we have $\mathcal{V} = \mathcal{V}^U \cup \mathcal{V}^A$. Links in the knowledge graph convey a wide variety of information. Basically they consist of three types: *user-to-user links*, *user-to-attribute links* and *attribute-to-attribute links*. Accordingly, we have $\mathcal{E} = \mathcal{E}^{UU} \cup \mathcal{E}^{UA} \cup \mathcal{E}^{AA}$. User-to-user links express users' relations, *e.g.*, $\langle$Bob, is colleague, Alice$\rangle$. User-to-attribute links specify users' attributes, *e.g.*, $\langle$Bob, has gender, male$\rangle$. And attribute-to-attribute links describe the correlations or some properties between attributes, *e.g.*, $\langle$Doctor, has salary, $>$ 50K$\rangle$. Given a link/triple $e \in \mathcal{E}$, $s(e)$, $p(e)$, $o(e)$, and $c(e)$ refers to the subject, predicate, object and confidence score of e. In other words, a link e is often denoted as $\langle s(e), p(e), o(e), c(e)\rangle$ while sometimes $c(e)$ is omitted for simplicity. Since links in the knowledge graph are directed, we assume users in the social network have one-way relations, such as "follower" (a two-way relation such as "friends" can be regarded as two one-way relations).

To protect users' privacy, the data publisher anonymizes a social network data G with its privacy preservation techniques, such as perturbation and sanitization on links and nodes. The published anonymous graph data is modeled by $G_a(\mathcal{V}_a, \mathcal{E}_a)$ (*anonymized graph*), in which every user/node's identification information (such as name) is removed, and there might be attributes generalized and nodes and links added or removed.

2.4 Attack Model

In the scenario of privacy-preserving data publishing, the data **publisher** holds a dataset and she can access all the information within the dataset, including users' identities and sensitive attributes. The data publisher is trusted and will protect users' privacy before and after publishing the dataset. And she would not disclose more information to other people or institutions than the published data.

The **attacker** has the desire of learning private information of a specific target or a group of users, or even all the users. She has access to the published datasets and has the capability of acquiring information, logical reasoning and computation. In addition, she might have a variety of background information known as prior knowledge, which enables her to eliminate some values from the set of sensitive attribute values and then infer the sensitive value with high confidence.

2.5 Attacker's Knowledge Model

We assume that the attacker is able to gather information as prior knowledge to de-anonymize the published dataset. We define knowledge formally as follows.

Definition 1 (Knowledge). Knowledge is an awareness and belief about the probability of a triple $t = \langle s, r, o\rangle$ being true, where $s \in \mathcal{S}$, $o \in \mathcal{O}$, $r \in \mathcal{R}$.

Knowledge sometimes can be denoted as the decrease in the entropy about the event distribution. The attacker's background knowledge can be obtained through multiple manners, *e.g.*, data aggregation, data mining, collaborative information systems, knowledge/data brokers, etc. The prior knowledge includes the following types.

- *Common sense*, *e.g.*, a male can never have uterine cancer.
- *Statistical information*, that is, relevant demographic information previously released by governmental or research institutions.
- *Personal information*, *i.e.*, information about a specific user learnt from her webpage and real life.
- *Network structural information e.g.*, number of neighbors, ego network.

Following Zhou *et al.* [61], we further classify structural information that an attacker may possess into five categories.

- *Node degrees*. Here degree might have different meanings in different scenarios.
- *Link relations*, *e.g.*, Bob follows Alice in Twitter, or Bob is a colleague of Alice.
- *Neighborhood subgraphs*, including the neighborhood users and relations of some target individuals. This knowledge may span multiple networks and include the target's alter egos in these networks [36].
- *Compromised subgraphs*, either planted by the attacker before the graph is published (called seed attack), or divulged by a collusion of a few users [39].
- *Graph properties*, such as closeness, centrality, betweenness, path length, reachability [18]. Sometimes, the global graph properties, such as the graph being a power-law graph, can also be exploited for attacking [24].

All of these information can be represented in a knowledge graph (including structural information, which can be seen as attributes), denoted by $G_p(\mathcal{V}_p, \mathcal{E}_p)$ (we call it *prior attack graph*). We assume the attacker targets at only a small number n_p of users from the global set $\mathcal{V}_a$ of users in the anonymized graph G_a, that is, $n_p << n_a$ (n_a represents total node number in G_a). This is more realistic than many prior works that assume $\mathcal{V}_p^U = \mathcal{V}_a^U$. Sometimes the attacker has imperfect and incomplete knowledge of properties or relations of users (called *probabilistic knowledge*, as the opposite of *certain knowledge*), so a confidence score (denoted as c) is attached to every link in the knowledge graph to express such uncertainty. Certain knowledge has a score of 1 or 0, meaning definitely true or definitely false.

In reality, a piece of knowledge may depend on another, either of the same or of a different type. Following [15], the knowledge correlations are classified into three types.

- *Mutual exclusion*. Given a subject s and a predicate r, for objects $o_1 \bigcap o_2 = \emptyset$, we have $c(\langle s, r, o_1 \vee o_2\rangle) = c(\langle s, r, o_1\rangle) + c(\langle s, r, o_2\rangle)$, and $c(\langle s, r, o_1 \bigcap o_2\rangle) = 0$. For example, $\langle \text{Jackie, has gender, female}\rangle$ and $\langle \text{Jackie, has gender, male}\rangle$ are mutually exclusive, *i.e.*, they cannot be true simultaneously.
- *Inclusion*. Given a subject s and a predicate r, for objects $o_1 \subseteq o_2$, we have $c(\langle s, r, o_1\rangle) \leq c(\langle s, r, o_2\rangle)$. For example, $\langle \text{Jackie, lives in, New York}\rangle$ is included by $\langle \text{Jackie, lives in, the USA}\rangle$.
- *Soft correlation*. For instance, the fact $\langle \text{Jackie, has occupation, waitress}\rangle$ implies that there is little possibility she has a yearly salary of over 100K dollars.

These correlations are very critical to the inference process of the attack. How the attack exploits them to make inference over users' attributes will be discussed later.

When the dataset is published, the attacker utilizes her prior attack graph and the ability of reasoning and computation to make inference on some facts, which can be either deterministic or probabilistic. Therefore, the attacker's knowledge graph would be updated while she is making inference. Links could be added or deleted, and their weights may increase or decrease. The final knowledge graph the attacker has after the reasoning process is denoted as G_q (*posterior attack graph*).

2.6 Privacy Disclosure

Privacy disclosure can be either deterministic or probabilistic. If the attacker has a large variation between the prior and posterior beliefs (which can also be quantified as the entropy), we call it a successful attack, that is, the target's privacy is disclosed. Privacy disclosure can be further classified into *attribute disclosure* and *relation disclosure*. Attribute disclosure means sensitive attribute of a given target is inferred by the attacker. The disclosed attribute is sensitive, at least in the viewpoint of the target. Similarly, relation disclosure refers to the situation where the attacker learns the secret relation between two targets. Note identity disclosure from a published dataset itself can hardly be deemed as loss of privacy, but it can result in attribute disclosure or relation disclosure indirectly.

2.7 Locality-Sensitive Hashing

Locality-Sensitive Hashing (LSH) maps elements to the same value with a high probability if they are close to each other and different value otherwise. The space of hash values is much smaller than the space of elements, so LSH is often used to reduce the dimension of high-dimensional data or group data into buckets. LSH can be implemented using error correction codes [13], [17], bit sampling [20], random projection [9], stable distributions [14], and MinHash [30]. Different LSH schemes apply to different data sets. In one of our de-anonymization algorithms, we group users into buckets using LSH so that similar users are more likely in the same bucket than dissimilar users are. As a result, mapping space is greatly reduced by comparing a user to other users in the same bucket only instead of all the users in the network.

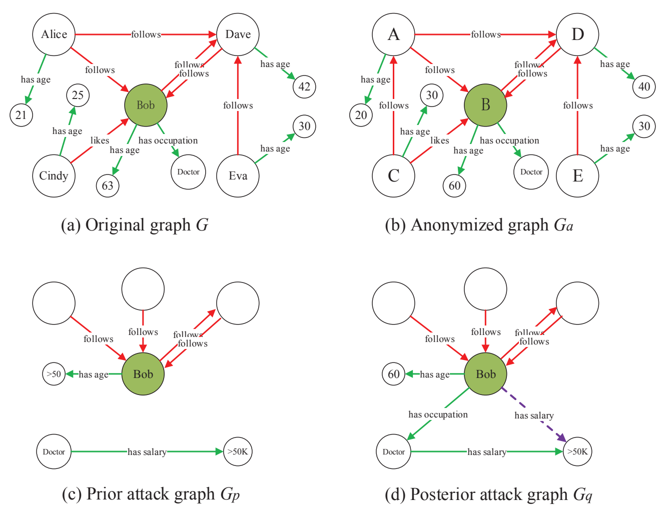
MinHash based LSH scheme [30] would be adopted for one of our cases. Minhash was first introduced by Andrei Broder [8] for quickly estimating the Jaccard similarity of two sets. A set of elements is stored in the form of a membership vector. The length of the vector $|\vec{v}|$ equals the size of the element space. Each bit value (1 or 0) represents whether an element is contained in the set. In our case, the set of a user's attributes is transformed to a membership vector where bits corresponding to her attributes are set to 1. Given a random permutation π over $[1, 2, \cdots, |\vec{v}|]$, the MinHash signature of $\vec{v}$ for this π is the index of the first bit whose value is 1 when all bits are visited in the order of π. Different MinHash signatures of $\vec{v}$ are generated for different permutations. Suppose for each vector we have $b \times r$ MinHash signatures generated with the same permutations, they are divided into b bands containing r signatures each. There is a hash function that hashes the r signatures to a bucket; the bucketing is done separately for different band. Similar vectors are very likely to produce the same MinHash signatures and thus hashed to the same bucket in a band. Two vectors are considered as a candidate pair as long as they are hashed to the same bucket in at least one band. Given two vectors with Jaccard similarity s, the probability that they become a candidate pair in this scheme is $1-(1-s^r)^b$. This function has the form of an S-curve. Pairs with greater s are very likely to become candidates while those with smaller s are very unlikely to be candidate pairs. Thus LSH is implemented with MinHash.

3 Knowledge Graph Based Attack Architecture Overview

The privacy attack process typically contains the following steps. A social network data is anonymized using various anonymization techniques (*e.g.*, ID removal, data perturbation), and then published. The attacker will first construct the prior attack graph, then apply de-anonymization and privacy inference techniques to infer private attributes, and finally measure privacy disclosure to recover some successfully attacked subgraph.

3.1 Prior Attack Graph Construction

For de-anonymization, the attacker first construct a prior attack graph, consisting of following three steps: 1) Initialize with certain knowledge, 2) Add probabilistic knowledge, and 3) Complement according to correlations. More sophisticated methods like knowledge graph identification [42] can be used in the construction. The attacker now possesses two graphs, the published anonymous data graph G_a and a prior attack graph G_p. She will take them as inputs, and construct a posterior attack graph as the output. This process can be divided into two parts: *de-anonymization* and *privacy inference*.



(a) Original graph G (b) Anonymized graph G_a (c) Prior attack graph G_p (d) Posterior attack graph G_q

Fig. 2: **An example of de-anonymization and privacy inference:** The publisher removes users' identifiers and adds/removes nodes/links in the original graph G, then publishes the anonymized graph G_a. The attacker models her prior knowledge by G_p and use it to attack G_a and learn users' privacy. Finally,her posterior knowledge is modeled by G_q.

3.2 De-anonymization

In the de-anonymization step, the attacker tries to map the target users in G_p to nodes in G_a, which is mainly based on the similarity of their attributes and relations. After the mapping, the attributes attached with the anonymous node is also linked to the target user. The updated prior attack graph is denoted as G'_p.

Fig. 2 gives a simple example. Suppose Bob is the target of the attack in the social network. We assume all the links here have confidence score of 1. The original network data (Fig. 2(a)) is published after anonymization and perturbations (Fig. 2(b)). Users' names are removed and replaced with pseudonyms. Their ages are generalized. The publisher might add/delete a few links randomly, *e.g.*, adding a link from Cindy to Alice. These perturbations are supposed to be subtle such that the utility of the network data is well preserved. The attacker constructs a knowledge graph around Bob as the prior attack graph (Fig. 2(c)). It is known to the attacker that Bob has three followers and one following, and that Bob has an age of over 50. Then the attacker maps Bob in the prior attack graph to a node in the published graph. He first finds that node B and D in Fig. 2(b) both have 3 followers and one following, which is in accordance with Bob, so B and D are added to the candidate set. Then the attacker notices that B has an age of 60 while D is 40. Obviously, the former is consistent with the knowledge that Bob is older than 50. Thus, Bob is mapped to node B. Accordingly, the attributes attached with node B in the published graph is also linked to Bob. In this case, the attacker knows that Bob is a doctor, so the link $\langle$Bob, has occupation, doctor$\rangle$ is added to the graph (Fig. 2(d)).

3.3 Privacy Inference

In this step, the attacker complements and updates the attack graph by inferring private attributes and relations that are not contained in G_a or cannot be learned from G_a. The intrinsic knowledge correlations play a key role in privacy inference. As depicted in Fig. 2(c), the attacker knows a doctor has a salary of over 50K dollars. The attacker has known that Bob is a doctor at the de-anonymization step, so she infers that Bob has a salary of ">50K" and adds the corresponding link (the purple dashed line in Fig. 2(d)) to her knowledge graph.

3.4 Privacy Disclosure Determination

After inferring Bob's private attributes and relations, the attacker finally has a new knowledge graph, *i.e.*, posterior attack graph G_q. Graph G_q evolves from G_p, but there could be new links or nodes added and confidence scores of links could be increased or decreased, compared with the latter. The variations of original and current confidence scores can be regarded as a metric of the extent of privacy leakage. As for a newly added link, it does not exist in G_p, which means the attacker has no idea whether it is true or false. So it is reasonable to set its original confidence score to follow the common distribution known by everyone (such as uniform distribution over all possible values, or normal distribution, or power-law distribution). Thus, we define privacy leakage as follows.

Definition 2 (Privacy disclosure). Given a piece of knowledge/triple t which is considered to be sensitive to the user, and t has a score of $c_p(t)$ in G_p and a score of $c_q(t)$ in G_q, the privacy t is considered to be disclosed if and only if it satisfies

$$\delta(c_p(t), c_q(t)) > \epsilon(t), \tag{1}$$

where δ is the distance function. If there are multiple sensitive attributes, the Kullback-Leibler divergence can be used to measure the distance of prior and posterior distributions. The threshold ϵ can be set to different values for different triple t (different subjects, predicate, or objects).

4 Knowledge Graph Based De-anonymization & Privacy Inference Methods

In this section, we present the de-anonymization and privacy inference methods based on the knowledge graph model. We will first introduce how de-anonymization is transformed into a bipartite matching problem and then design two heuristics for bipartite construction. Finally we would describe how to perform privacy inference on the knowledge graph.

4.1 Node Similarity

Before presenting the de-anonymization algorithm, we first define the node similarity measurement. The similarity $S(i,j)$ between two user nodes i, j includes the attribute similarity and the structural similarity scores.

Attribute Similarity: For computing the attribute similarity score $S_A(p,a)$ between two nodes, $p \in \mathcal{V}_p^U$,$a \in \mathcal{V}_a^U$, we use the following attribute set.

Attribute set: For $p \in \mathcal{V}_p^U$ (resp., $\mathcal{V}_a^U$), its attribute set $\mathcal{A}_p$ is a set of tuples of predicates and objects. For example, the attribute set of Bob in Fig. 2(b) is $\{(has\ age, 60),(has\ occupation, doctor)\}$.

In a knowledge graph, each user node s has some links connecting to attribute nodes. Given a node $p \in G_p$ (and a set of emergent links $E(p) = \{\langle p,r,o\rangle \mid r \in \mathcal{R}, o \in \mathcal{O}\}$ with confidence score c for each triple $\langle s,p,o\rangle$), and a node $a \in G_a$ (with a set of links $E(a) = \{\langle a,r,o\rangle \mid r \in \mathcal{R}, o \in \mathcal{O}\}$), the attribute similarity between p and a is denoted by the probability $Pr(E(a) \mid E(p))$ that these links are observed with the prior knowledge. In this work, we use $I(r,o)$ to denote whether the link

$\langle p, r, o\rangle$ from G_p appeared in G_a as $\langle a, r, o\rangle$. Then the attribute similarity of p and a is defined as,

$$\begin{aligned} S_A(p,a) &= Pr(E(a)|E(p)) \\ &= \prod_{I(r,o)=1} c(\langle p,r,o\rangle) \cdot \prod_{I(r,o)=0} (1 - c(\langle p,r,o\rangle). \end{aligned} \quad (2)$$

Notice that here we assume that the link that appeared with 100% confidence in the G_p will also appear in G_a. If this is not the case (*i.e.*, the link disappears due to the anonymizing operation), we use c_0 to denote the probability that this event happens. Then the attribute similarity can be modified as $S_A(p,a) = \prod_{I(r,o)=1} c(\langle p,r,o\rangle) \cdot \prod_{I(r,o)=0} (1 - c(\langle p,r,o\rangle)c_0)$. For numerical attributes (such as age), the difference of its values in G_p and G_a over the maximum difference can be used as its weight.

Structural Similarity: In addition to attribute similarity, structural similarity between two nodes is also considered. We use the following information to compute the structural similarity score $S_R(p,a)$ between $p \in \mathcal{V}_p^U$ and $a \in \mathcal{V}_a^U$.

Inbound neighborhood $\mathcal{I}_u$: For a user node $u \in \mathcal{V}_p^U$ (resp., $\mathcal{V}_a^U$), its inbound neighborhood $\mathcal{I}_u$ is a set of predicates of relational links that are incident to u.

Outbound neighborhood $\mathcal{O}_u$: This is a set of predicates of relational links that are emergent from a node u.

ℓ-hop neighborhood: We may also consider the general ℓ-hop neighborhood of a node p and its induced subgraph. When $\ell = 1$, it is simply *ego neighborhood*.

The structural similarity between two nodes is

$$S_R(p,a) = w_i S_i(p,a) + w_o S_o(p,a) + w_\ell S_\ell(p,a) \quad (3)$$

where $w_i + w_o + w_\ell = 1$, $w_i, w_o, w_\ell \in [0,1]$, and $S_i(p,a)$, $S_o(p,a)$, and $S_\ell(p,a)$ denote the inbound similarity, outbound similarity, and ℓ-hop neighborhood similarity. We compute these similarities by Jaccard index, *e.g.*, $S_i(p,a) = J(\mathcal{I}_p, \mathcal{I}_a) = \frac{|\mathcal{I}_p \cap \mathcal{I}_a|}{|\mathcal{I}_p \cup \mathcal{I}_a|}$.

Node Similarity: Then we assign weights to S_A, S_R so the node similarity $S(p,a)$ of p and a is computed as

$$S(p,a) = w_A S_A(p,a) + (1 - w_A) S_R(p,a). \quad (4)$$

Therefore, the similarity S is a normalized function.

4.2 De-anonymization Formulation

We assume all the users in the attacker's prior knowledge have corresponding nodes in G_a, that it, $\mathcal{V}_p^U \subseteq \mathcal{V}_a^U$ (Overlapped graphs will be discussed in Section 6). Suppose $\pi : \mathcal{V}_p^U \to \mathcal{V}_a^U$ is an injective mapping from users in G_p to anonymized user nodes in G_a. The goal of de-anonymization is to find such a mapping π that maximizes the similarity between G_p and G_a,

$$\text{argmax} \ \underset{\pi}{\text{Sim}}(G_p, G_a), \quad (5)$$

where Sim is a function that measures the similarity between two graphs G_p and G_a after their user nodes are matched by a function π. We compute Sim by summing up the similarity scores of their *matched* user nodes.

$$\underset{\pi}{\text{Sim}}(G_p, G_a) = \sum_{(i,j)\in\pi} S(i,j), \quad (6)$$

To transform the problem, we introduce a complete weighted bipartite graph $G_B(\mathcal{V}_p^U + \mathcal{V}_a^U, \mathcal{E}_B)$, in which a weight $S(i,j)$ is assigned to each link $e_{ij} \in \mathcal{E}_B$. Any link is a possible candidate match. Thus the de-anonymization problem can be reduced to the maximum weighted bipartite matching problem (also known as the assignment problem), which can be further reduced to the minimum cost maximum flow problem, and thus can be solved by many algorithms [4], [5]. We implement an algorithm based on [2], which has $O(nmf)$ time complexity and $O(n^2)$ space complexity. Herein, $n = n_p + n_a$ is node number in G_B, m is link number, and f is the maximal flow value ($f = O(n_p)$ in our case). Notice that for real world social networks n_a can be very large, even up to millions.

There are three challenges in implementing such method:

1) Constructing this complete bipartite graph has large space and time complexity,
2) Finding the maximum weighted matching also has a large time and space complexity for large scale network data,
3) Selecting proper features from the attributes and structural similarity, and assign a proper weight for every feature has a large impact on the de-anonymization performance. This sometimes is more of an art than science.

Graph G_B has $O(m) = O(n_p n_a)$ links in total. To reduce mapping space and complexity, we can decrease m by keeping only links with largest weights, without computing similarities of all pairs of nodes. Specifically, each user in $\mathcal{V}_p^U$ is linked to top k candidate nodes in $\mathcal{V}_a^U$. Here k is a predefined parameter that balances accuracy and complexity. Since we have assumed $n_p \ll n_a$, there will be many isolated nodes in $\mathcal{V}_a^U$ that can be removed. Accordingly, n_a is reduced to $O(kn_p)$, and thus time and space complexity of solving the assignment problem is lowered to $O(k^2 n_p^3)$ and $O(k^2 n_p^2)$, respectively. However, now the major complexity lies in building such a light-weighted bipartite graph G_B. We will present two heuristic techniques of bipartite construction on the basis of the top-k strategy.

4.3 BFS Based Bipartite Construction

In Algorithm 1 we build a light-weighted bipartite graph with the top-k strategy by traversing G_p. The intuition is that if two nodes match, their neighbors are also very likely to match. An example is given in Fig. 3. Mapping space is thus lowered by utilizing users' connections. We choose to perform breadth first search (BFS) on G_p, which induces less error accumulation than DFS. In the beginning, G_B has no links. The algorithm selects an outstanding initial node from G_p such that it can be mapped with high confidence, which is very critical as it has an impact on mapping other connected nodes. This initial node can be picked with different ways. In our approach, we randomly pick a node, compute the similarities between it and all nodes in G_a, and check if there is a distinct disparity. If the range of the similarities is greater than a threshold r_{min}, then pick this node as the initial node. As an alternative, we can compute the structure/attribute score of a node (such as the degree of the node, the size of the ego network, the size of ℓ-hop network, the set of attributes of a node), and then sort nodes in decreasing order of the structure/attribute score. We match the node in G_p with nodes G_a having the largest score similarities.

Then BFS is performed in G_p where link directions are ignored temporarily. Before mapping each node $p \in G_p$, the algorithm checks whether p has a predecessor/father node (denoted as $pr(p)$) in the BFS. If so, it searches the neighbors of $pr(p)$'s candidate matches, to get the top k similar candidates

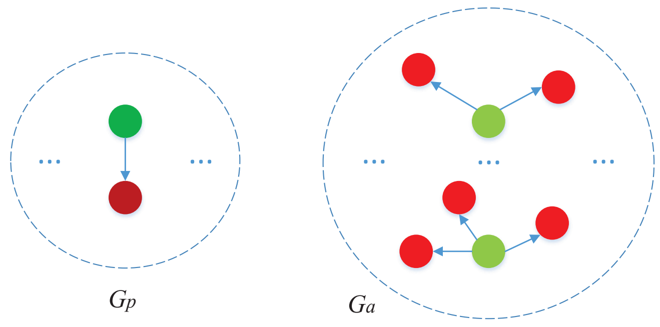


Fig. 3: **An example of BFS-based bipartite construction:** The intuition is that if two nodes match, their neighbors are also very likely to match. Suppose we already know the light green nodes in G_a are candidate matches for the dark green node in G_p, and suppose the dark green node is the predecessor of the dark red node. To find the candidates for the dark green node in G_p, we just need to search among the neighbors of the two light green nodes in G_a.

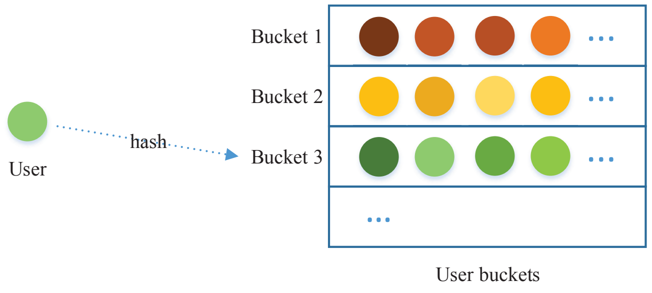


Fig. 4: **An example of LSH-based bipartite construction:** Step 1: Group nodes of G_a into buckets using LSH such that similar nodes are grouped in the same bucket. The buckets is shown on the right. Step 2: Given any user in G_p, use the same hash function to convert the user's profile to an index, then find the corresponding bucket (row in the table). The nodes in this bucket are candidates of the user's match.

for p. (Luckily, the expected number of neighbors of a node is far less than n_a for real social networks *e.g.*, the average number of Facebook friends of a person is about 330 [1].) Otherwise, it searches all the nodes in $\mathcal{V}_a^U$, which is a relatively rare case. Among these top-k potential matching candidates, we remove the candidates whose similarity with the node p is less than a threshold s_{min}. Then k links connecting p with its candidates are added to G_B and the similarities are attached as weights. When BFS is done, all the nodes in G_p are mapped. After removing isolated nodes in $\mathcal{V}_a^U$, we finish constructing the bipartite graph. Compared to building a complete bipartite which costs $O(n_a n_p)$ time, the top-k strategy has much less overhead to build the reduced bipartite.

Obviously, the choice of parameter k, the weight threshold s_{min}, and the weight range threshold r_{min} will have a large impact on the accuracy of the final mapping result. The smaller s_{min} is (or the larger k is), the more candidates a user would have, which results in greater search space for mapping the user's successors. On the other hand, large s_{min} (and small k) could result in a reduced bipartite graph missing some links from the real maximum weighted matching. Later on our experiment will evaluate the impact of these parameters. We found that typically $k = 10$ is enough for achieving high accuracy with small complexity.

4.4 LSH Based Bipartite Construction

The top-k candidate selection method presented above might have the problem of error accumulation, because the selection

Algorithm 1 BFS Based Bipartite Graph Construction

```
Require: Anonymized graph $G_a$, prior attack graph $G_p$, parameter $k$.
Ensure: A bipartite graph $G_B$.
 1: Define $\mathcal{E}_B = \emptyset$, build a bipartite graph $G_B(\mathcal{V}_p^U + \mathcal{V}_a^U, \mathcal{E}_B)$.
 2: Pick an initial node $p_0 \in \mathcal{V}_p^U$.
 3: Perform BFS in $G_p$ starting from $p_0$ (treat $G_p$ as undirected graph).
 4: for each $p \in \mathcal{V}_p^U$, following the order of BFS do
 5:   if $p$ has a predecessor $pr(p)$ then
 6:     Define $\mathcal{N} = \emptyset$.
 7:     for each $a \in \mathcal{C}_{pr(p)}$ do
 8:       for each neighbor $n$ of $a$ do
 9:         if The relation of $n, a$ is the same as that of $p, pr(p)$ then
10:           $\mathcal{N} = \mathcal{N} \cup \{n\}$.
11:     for each $n \in \mathcal{N}$ do
12:       Calculate $S(p, n)$.
13:     Add top $k$ similar nodes to $\mathcal{C}_p$ as $p$'s candidates.
14:   else
15:     for each $a \in \mathcal{V}_a^U$ do
16:       Calculate $S(p, a)$.
17:     Add top $k$ similar nodes in $\mathcal{N}$ to $\mathcal{C}_p$.
18:   for each $a \in \mathcal{C}_p$ do
19:     Attach weight $S(p, a)$ to $e_{pa}$, $\mathcal{E}_B = \mathcal{E}_B \cup \{e_{pa}\}$.
20: Remove isolated nodes in $a \in \mathcal{V}_a^U$.
21: return $G_B$.
```

of candidate matches of a user is dependent on the selection for her predecessor. Now we design a new method to avoid error accumulation by finding candidates for each user independently. The key idea is to group users in G_a into buckets for indexing. Please see Fig. 4 for an example. Below are some notations used in the algorithm. Hereafter, we assume there is only one type of user-to-user relation for simplicity. The formula can be easily extended to the case where there are multiple types of relations like friends, colleagues and classmates.

Node profile: For a user node $a \in \mathcal{V}_a^U$ (resp., $\mathcal{V}_p^U$), its profile $\mathcal{A}_a$ is a set of tuples of predicates and objects. For example, the profile of Bob in Fig. 2(b) is $\{(has\ age, 60), (has\ occupation, doctor)\}$.

Neighborhood profile of a node a refers to the set of profiles of a's neighbors, which is split into two sets, *inbound neighborhood profile* $\mathcal{P}_a^I = \{\mathcal{A}_v | v \in \mathcal{I}_a\}$, and *outbound neighborhood profile* $\mathcal{P}_a^O = \{\mathcal{A}_v | v \in \mathcal{O}_a\}$.

User buckets H is a 2D array of which each row is a bucket of nodes whose node profiles are similar. Specifically, for each node a we compute the hash value $h(\mathcal{A}_a)$ of its node profile using a LSH function h, and then insert the node to the bucket $H[i]$ (the index i equals $h(\mathcal{A}_a)$). The implementation of LSH, *i.e.* the selection of h, is highly dependently on the properties of the data to be bucked. Thanks to the property of LSH, nodes with more similar profiles are more likely to be grouped the same bucket. We will search for possible candidates of each user within the corresponding bucket, which is much more efficient than searching the whole node set.

Neighborhood similarity (denoted as $\hat{S}$) measures the similarity of two nodes according to the likeness of their 1-hop neighborhood, specifically the profiles of their inbound and outbound neighbors. For $p \in \mathcal{V}_p^U$ and $a \in \mathcal{V}_a^U$, their neighborhood similarity is defined as

$$\hat{S}(p, a) = \frac{1}{2}\left(\frac{|\mathcal{P}_p^I \cap \mathcal{P}_a^I| + 1}{|\mathcal{P}_p^I| + 1} + \frac{|\mathcal{P}_p^O \cap \mathcal{P}_a^O| + 1}{|\mathcal{P}_p^O| + 1}\right). \quad (7)$$

Considering the attacker has incomplete knowledge, missing neighbors is acceptable when comparing p's neighbors with a's but redundant neighbors are not allowed. This is why we choose this asymmetric definition instead of Jaccard Index. To make sure the denominator is non-zero, we add 1 to it and to the numerator so it is still normalized.

Algorithm 2 LSH Based Bipartite Graph Construction

```
Require: Anonymized graph $G_a$, prior attack graph $G_p$, parameter $k$.
Ensure: A bipartite graph $G_B$.
 1: Define $\mathcal{E}_B = \emptyset$, build a bipartite graph $G_B(\mathcal{V}_p^U + \mathcal{V}_a^U, \mathcal{E}_B)$.
 2: Create a 2D array $H$ of neighborhood fingerprint.
 3: for each $a \in \mathcal{V}_a^U$ do
 4:     Calculate $h(\mathcal{A}_a)$.
 5:     Add $a$ to the array $H[h(\mathcal{A}_a)]$.
 6: for each $p \in \mathcal{V}_p^U$ do
 7:     $\mathcal{N} = \emptyset$.
 8:     Calculate $h(\mathcal{A}_p)$.
 9:     for each $a$ in the array $H[h(\mathcal{A}_p)]$ do
10:         Calculate neighborhood similarity $\hat{S}(p, a)$
11:         if $\hat{S}(p, a) \geq s_{min}$ then
12:             $\mathcal{N} = \mathcal{N} \cup \{a\}$.
13:     Add top $k$ similar nodes in $\mathcal{N}$ to $\mathcal{C}_p$ as $p$'s candidates.
14:     for each $a \in \mathcal{C}_p$ do
15:         Attach weight $\hat{S}(p, a)$ to $e_{pa}$, $\mathcal{E}_B = \mathcal{E}_B \cup \{e_{pa}\}$.
16: Remove isolated nodes in $a \in \mathcal{V}_a^U$.
17: return $G_B$.
```

As shown in Algorithm 2, the construction of the user buckets H is straigthforward. For each node a in G_a, calculate its hash value $h(\mathcal{A}_a)$, then add it to the bucket $H[h(\mathcal{A}_a)]$. The time complexity is $O(\mathcal{V}_a^U + \mathcal{E}_a^{UA})$. After the construction, we then find top-k candidates for user nodes in G_p by referring to H. For each node p in G_p, calculate its hash value $h(\mathcal{A}_p)$. The bucket $H[h(\mathcal{A}_p)]$ stores nodes who have similar node profiles with p. Then we determine whether those nodes also have similar neighborhood profiles by calculating their neighborhood similarity $\hat{S}(p, a)$, for any $a \in H[h(\mathcal{A}_p)]$. If the similarity is above a pre-specified threshold s_{min}, it suggests that node a is very likely to be a match of p. We pick top k most similar nodes to be the candidates of p. The rest steps are the same as in BFS based bipartite construction.

This algorithm matches users according to their neighborhood similarity and independently without error accumulation so hopefully it would achieve better accuracy compared to the BFS base method. Meanwhile, it has greater time complexity. The main overhead lies in the computation of neighborhood similarities where each attribute of each neighbor will be accessed for comparison. In this case, k has little influence on the time complexity of bipartite construction since searching for candidates of a user does not depend on the candidates of her predecessor any more.

4.5 Path Ranking Based Privacy Inference

We now present out methods for inferring users' private attributes (including relations between users), which is regarded as link prediction in the knowledge graph. One way to infer/predict new links is to utilize the path ranking algorithm (PRA) proposed by Lao *et al.* [29], which was designed to complement existing knowledge bases. Given any two nodes s, o in the knowledge graph (G'_p in our case), PRA finds a set of paths $P_1, P_2, \ldots, P_n$ connecting s and o, which can be interpreted as rules. The paths are combined by fitting a binary classifier. The probability distributions of reaching o from s along the paths are used as features. Based on logistic regression, we can classify whether a triple $\langle s, r, o \rangle$ holds and thus perform the link prediction task.

5 Experiment Evaluations

We conduct de-anonymization and privacy inference experiments on two real world social network datasets and then we present a comprehensive evaluation on our methods, which validates the effectivity of knowledge graphs as a model of attacker's prior knowledge.

Dataset	$\lvert\mathcal{V}^U\rvert$	$\lvert\mathcal{V}^A\rvert$	$\lvert\mathcal{E}^{UU}\rvert$	$\lvert\mathcal{E}^{UA}\rvert$	$\lvert\mathcal{E}_p^{AA}\rvert$
Google+	107,614	15,691	13,673,453	378,880	2,262
Pokec	306,568	576	2,822,492	1,532,840	38

TABLE 2: **Statistics of two datasets:** the numbers of users, attributes, user-to-user links, user-to-attribute links, and attribute-to-attribute links. Here, attribute-to-attribute links are not contained in the original datasets. We generate them by calculating conditional probabilities between attributes.

5.1 Methodology

5.1.1 Datasets

We simulate our methods on two real world datasets, Google+ and Pokec, both from Stanford Network Analysis Project (SNAP) [3]. Google+ is a social layer for Google services operated by Google Inc., and Pokec is the most popular online social network in Slovakia. They contain rich network data and users' profiles. For Google+, meaningless and duplicate user profiles are removed when we preprocess it; for Pokec, users who have incomplete or invalid attributes are removed. Table 2 shows the statistics of the preprocessed datasets. The relations in the two social networks are oriented, and there is only one type of relation between users: "follows". For Pokec, the selected profiles contains 5 attributes: gender, location, age, height, weight, which are in the form of a relational table. Profiles in Google+ contain 6 attributes: gender, institution, job title, last name, place, university, yet they are not tabular as a user may have multiple values for a single attribute, such as multiple job titles. The preprocessed datasets are treated as original graphs G and used as ground truth. The last column of Table 2 is the number of synthetic attribute-to-attribute links for each dataset. Those links are generated only for the prior attack graph G_p. They represent the attacker's knowledge of attribute correlations, which would be utilized to perform privacy inference. We will mention how we generate G_p later.

5.1.2 Anonymized Graph & Prior Attack Graph Generation

Before performing de-anonymization on the datasets, firstly we need to anonymize them in order to generate anonymized graphs G_a. Given an original graph $G(\mathcal{V}^U \cup \mathcal{V}^A, \mathcal{E}^{UU} \cup \mathcal{E}^{UA} \cup \mathcal{E}^{AA})$, users' private data is contained in $\mathcal{E}^{UU} \cup \mathcal{E}^{UA}$, representing relations and profiles correspondingly. To anonymize G, both relations and profiles should be perturbed. For the former, we adopt the sampling method which is commonly used in previous works [22], [24], [36]. (Previous anonymization approaches are not suitable for us because most of them [33], [35], [47], [53], [58] are for undirected graphs while we adopt the directed graph model. We will have a small evaluation on them later in Table 4.) Specifically, links in $\mathcal{E}_a^{UU}$ are randomly sampled from $\mathcal{E}^{UU}$ with a sample ratio sr_a. All the links' confidence scores are set to 1. We also need to perturb users' profiles. For Pokec, we adopt the Flash algorithm [27] that achieves K-anonymity ($K = 10$, use capital K for disambiguation) to generalize the profiles. However, the Google+ profiles are not in the form of relational data because a user could have multiple values for a single attribute like job titles and some attributes have various possible values. Traditional data perturbation algorithms do not apply to such kind of data, and it is challenging to design a new fancy approach to anonymize it. Thus, we perturb it by suppressing a small portion of the attributes, *i.e.* $\mathcal{E}_a^{UA}$ is sampled from $\mathcal{E}^{UA}$ for Google+. In addition, user IDs in $\mathcal{V}^U$ are removed and substituted with pseudonyms. It is assumed that $\mathcal{V}^A$ stays the same when G_a is generated, and $\mathcal{E}^{AA} = \mathcal{E}_a^{AA} = \emptyset$.

Meanwhile, we generate the prior attack graph G_p for the attacker. Since it is very hard to predict what knowledge a real attacker would collect, we simulate it by sampling and calculating statistics of the original graph. First, we select a few users from $\mathcal{V}^U$ to $\mathcal{V}_p^U$ by some means (stated later) as the attacker's target users. Then $\mathcal{E}_p^{UU}$, $\mathcal{E}_p^{UA}$ are randomly sampled from only the relations and profiles relevant to these users, at the sample ratio sr_p. Likewise, $\mathcal{V}_p^A = \mathcal{V}^A$ is assumed. Besides, links for $\mathcal{E}_p^{AA}$ can be synthesized by calculating conditional probabilities between attributes ($Pr(c \mid b) = Pr(b,c)/Pr(b) = n(b,c)/n(b)$, for any $b, c \in \mathcal{V}^A$). Attribute-to-attribute links are used for privacy inference only. We select weight as the target attribute to be inferred for Pokec users, job title for Google+ users. As a result, we only take into account the links from any of other attributes to the target attribute. As shown in Table 2, we have much more attribute-to-attribute links for Google+ because it has a wide range of attribute values for institution, last name, place, and university.

Given a user sampling ratio usr, three sampling methods are used to select target users $\mathcal{V}_p^U$.

1) RS: Randomly sample from $\mathcal{V}^U$ at the ratio usr;
2) EG: Sample n_p users within an ego network of a user ($n_p = n_a \times usr$);
3) RW: Sample n_p users from $\mathcal{V}^U$ based on random walk ($n_p = n_a \times usr$), which reflects how people know friends.

Later on, we will evaluate the influence of different user sampling methods on the attack performance.

5.2 Evaluation on De-anonymization

To evaluate our de-anonymization algorithm, we use accuracy (ratio of correct matches) and run time as metrics to measure utility and complexity.

5.2.1 BFS Based Bipartite Construction

There are several important parameters that have an influence on the performance. Their default parameter settings are listed in Table 3, and RW is chosen as the default user sampling method as it is most practical. The first three parameters usr, sr_a, sr_p controls how noisy the generated graphs G_a, G_p are, *i.e.* the difficulty level of de-anonymizing users in G_p from users in G_a. The rest four k, r_{min}, s_{min}, w_A tune the de-anonymization algorithm (see Section 4.3 for details). Since experiments on Google+ and Pokec are alike, we focus on Google+ unless there is a difference.

Fig. 5 effectively shows how the settings of k, s_{min} influence de-anonymization accuracy and time complexity. As depicted in Fig. 5(a), increasing k (recall that we match a user with the top k users from G_a in building the bipartite graph) can improve accuracy when $k \leq 10$ but has a minor effect when $k > 10$. This is because accuracy is bounded by sample ratio sr_a, sr_p (see details in Fig. 8). Yet run time constantly increases with the growing k (Fig. 5(b)). Therefore, we choose $k = 10$ as default. It is revealed in Fig. 5(c),5(d) that s_{min} has negative correlations with accuracy and run time so it balances the tradeoff between them, which is in accordance with our intuition.

Parameter	Meaning	Default
usr	User sampling rate for G_p	0.005
sr_a	Edge sampling rate for G_a	0.8
sr_p	Edge sampling rate for G_p	0.8
k	For choosing top k similar candidates	10
r_{min}	Minimum range of similarity scores, for picking initial node	0.8
s_{min}	Minimum similarity score required for candidates	0.5
w_A	Weight of attribute similarity	0.5

TABLE 3: Parameters and default settings

Fig. 6 indicates that run time is in proportion to usr but accuracy almost keeps stable, because usr decides the number of target users to be matched, but does not affect the mapping algorithm. Fig. 7 shows that the de-anonymization method has best accuracy when $0.4 \leq w_A \leq 0.9$. Recall $w_A, 1 - w_A$ are the weights assigned to the attribute similarity and relation similarity of two nodes. Thus, it is indicated that both of the two features play an important role in measuring node similarity. But this figure also implies that structural features help less compared with attribute features. This is because nodes of G_p is a small subset of those of G_a, which could results in great structural discrepencies between their nodes.

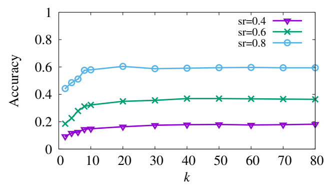
Recall sr_a, sr_p are the link sampling ratios of G_A, G_p, which reflect information fidelity after anonymization and the amount of the attacker's prior knowledge. They intrinsically determine the de-anonymizability of the published graph. As shown in Fig. 8(a), our de-anonymization method has larger accuracy when sr_a, sr_p are closer to 1. As shown in Fig. 8(b), sr_a has a more dominant effect on run time than sr_p, which can be explained by $n_p \ll n_a$. To be practical, we set $sr_a = sr_p = 0.8$.

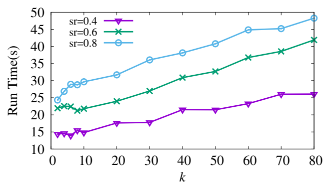
After adjusting these parameters, we make a comparison of three different user sampling methods (Fig. 9). In the left side is run time of 3 phrases of de-anonymization and total time, and the right side compares the methods in terms of accuracy. It is shown that RS has the largest complexity and the worst accuracy, RW has the best accuracy, and EG is the most efficient method. We can also learn from this figure that the main time complexity lies in loading dataset and building bipartite graph.

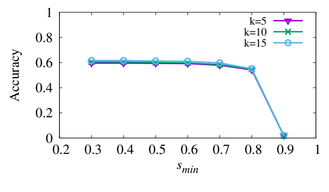
Anonymization approach to be cracked	Accuracy
Randomly sample edges (ratio= 0.8)	0.531
Randomly switch edges [58] (ratio= 0.2)	0.429
k-Degree Anonymity [33] ($k = 50$)	0.382
Bounded t-means clustering [53] ($t = 2000$)	0.347
Differential privacy [47] ($\epsilon = 1$)	0.259
Random walk [35] (#steps=10, max.retry=10)	0.222

TABLE 4: **Accuracy of de-anonymization against existing graph anonymization approaches (BFS, Google+):** Users' relations are perturbed by these approaches while their profiles are still perturbed by sampling as before. Source codes of these methods are from the SecGraph project by Ji *et al.* [23]. Google+ is treated as undirected graph when it is anonymized.

We also use some of the existing anonymization methods to generate G_a and then perform our de-anonymization algorithm. The results are given in Table 4. We can see that better anonymization is harder to crack. Tests on Pokec have similar results, except that the accuracy is lower (usually < 0.3). This is because 10-anonymity was applied to the original profiles we tested, so it is much harder to differentiate users by attribute features. In addition, the average degree is only 18 which indicates that the graph contains poor structural information. In such case, using more


(a) Accuracy - k


(b) Run time - k


(c) Accuracy - s_{min}

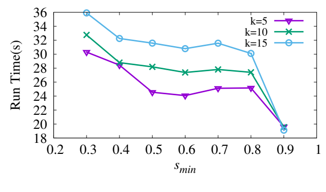

(d) Run Time-s_{min}

Fig. 5: **The impact of k, s_{min} on de-anonymization accuracy and run time (BFS, Google+):** Both k and s_{min} balance de-anonymization accuracy and run time. It is implied that setting k to 10 is enough to achieve high accuracy while incurring small computation overhead.

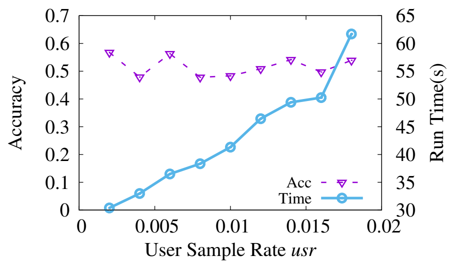


Fig. 6: **The impact of usr on de-anonymization accuracy and run time (BFS, Google+):** Run time is proportional to the user sampling rate usr but accuracy keeps stable.

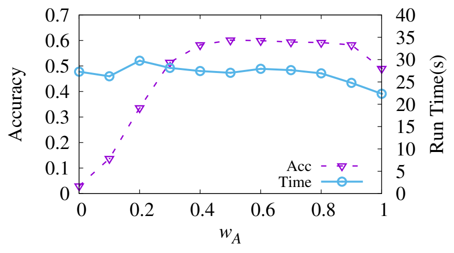
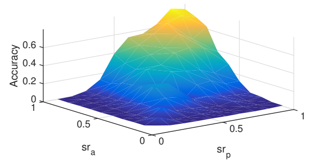


Fig. 7: **The impact of w_A on de-anonymization accuracy and run time (BFS, Google+):** It suggests that both attribute similarity and structural similarity are important to determining if two nodes match.

refined structure-based features will greatly improve the accuracy, such as ℓ-hop neighborhood, closeness centrality, top-k reference distance, landmark reference distance [24].

5.2.2 LSH Based Bipartite Construction

Since the selection of LSH function depends on the properties of the data to be grouped, we choose different LSH implementation for the two data sets. For Pokec, minhashing based method [30] is adopted which groups users into buckets on the basis of Jaccard similarity. However, this method has poor performance in grouping users in Google+ data set, for users' attributes have such a large diversity that most users have very small Jaccard similarity scores, causing almost all users to be binned into one bucket.


(a) Accuracy - sr_a, sr_p

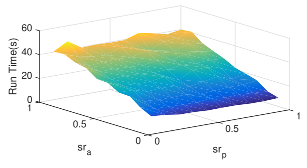

(b) Run time - sr_a, sr_p

Fig. 8: **The impact of sr_a, sr_p on de-anonymization accuracy and run time (BFS, Google+):** De-anonymization would be more accurate with higher sampling rates for generating the anonymized graph and prior attack graph. Besides, sr_a has a dominant effect on the run time.

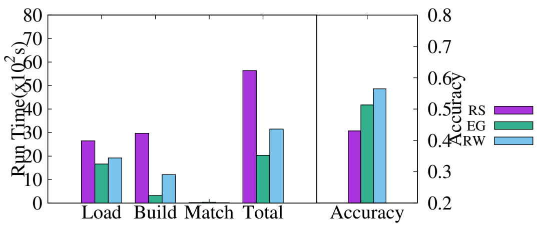
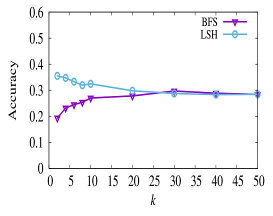


Fig. 9: **The impact of user sampling methods on de-anonymization accuracy and run time (BFS, Google+):** Randomly sampling target users results in the worst complexity and accuracy of de-anonymization. Random-walk sampled users are most prone to de-anonymization because it has the best accuracy, but sampling from an ego network can make the algorithm run efficiently most.

Instead, we group users in Google+ by their attributes which can also be treated as an implementation of LSH. For example, users in the same place are grouped in the same bucket. Last name, place and job title each are selected for grouping because they are more distinguishable than other attributes are. The parameter s_{min} is set to 0.9 for Pokec and 0.3 for Google+ respectively. The values are chosen according to the similarity among users under the neighborhood similarity metric in the data sets.


(a) Accuracy - k

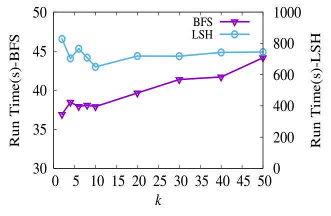

(b) Run time - k

Fig. 10: **The impact of k on de-anonymization accuracy and run time (LSH, Pokec):** This figure shows that the LSH based method achieves higher accuracy and has greater time complexity than the BFS based method.

Fig. 10 shows that the LSH based method achieves higher accuracy than the BFS based for Pokec data set, though it causes much greater time complexity, which is still acceptable for de-anonymization attack. The reasons of higher accuracy are twofold. The BFS based bipartite construction approach avoids error accumulation by mapping users independently, and it utilizes the more precise neighborhood similarity metric to compare users. Fig. 10(a) reveals that greater k leads to worse accuracy, while it has an opposite effect for the BFS case. Since the neighborhood similarity is very precise, a user and her very match are very likely to have a higher similarity score than that of her and another candidate, so choosing top-2 candidates is enough to include the right match. On the contrary, larger k would hardly increase the accuracy but make the bipartite matching more confusing with the interferential edges. Thus, greater k does not bring higher accuracy as in the BFS method. In Fig. 10(b), run time hardly changes with k when the bipartite construction is based on LSH, while it increases proportionally with growing k for the BFS based method. This is because searching for candidates of a user does not depend on the candidates of her predecessor in the LSH based method.

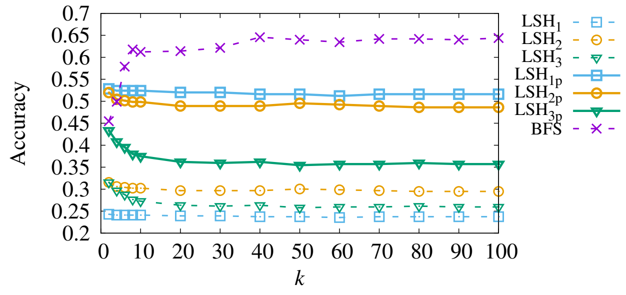


Fig. 11: **The impact of k on de-anonymization accuracy (LSH, Google+):** This figure shows that the LSH based method achieves lower accuracy than the BFS based. Choosing 2 attributes ("LSH$_2$": last name and place) for user binning is the best for LSH. The results are better when we measure de-anonymization accuracy within users whose attributes are not null (denoted as "LSH$_p$").

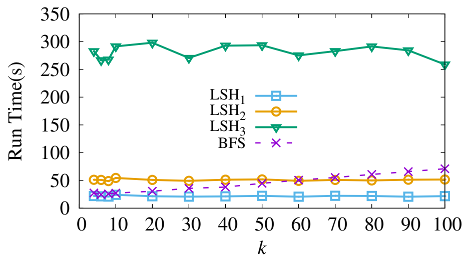


Fig. 12: **The impact of k on de-anonymization run time (LSH, Google+):** When the bipartite construction is based on LSH, run time does not change with k, while it increases proportionally with growing k for the BFS case. The more attributes are used for user binning, the higher run time is induced.

The results for Google+ are depicted in Figs. 11 and 12. Three cases of user bucketing are considered, indicated by subscripts 1, 2, 3. "LSH$_1$" refers to using last name for bucketing, "LSH$_2$" for last name and place, and "LSH$_3$" for last name, place and job title. Fig. 11 shows that the LSH based method does not achieve better de-anonymization accuracy, which is due to two reasons. For one thing, over 40% users in G_a have null values for all of last name, place and job title due to our lossy sampling, which directly results in failure to group them and find their similar counterparts. For another thing, grouping users by their attributes might not be a perfect LSH scheme for the Google+ data set. After removing users whose attributes are null and re-computing the accuracy (referred to as *partial accuracy* and indicated by a subscript 'p'), the results are much better, and even better than the BFS base method when $k < 5$. It is shown in Fig. 12 that k has little influence on the time complexity for the LSH method and using more attributes for bucketing incurs higher computation overhead. According to Figs. 11 and 12, selecting last name and place for bucketing has the highest accuracy (LSH$_2$), the second highest partial accuracy (LSH$_{2p}$) and little run time. On the contrary, selecting three attributes results in much higher time complexity but not better accuracy. Selecting only last name has the highest partial accuracy (LSH$_{1p}$) and the lowest overhead. However, it only applies to a small portion of users as the last names of most users in the data set are missing, which explains why it has the lowest accuracy (LSH$_1$) . We can therefore conclude that selecting last name and place is the best out of the three cases for user bucketing in Google+. In addition, like in the test on Pokec, greater k leads to worse accuracy until $k = 20$.

5.3 Evaluation on Privacy Inference

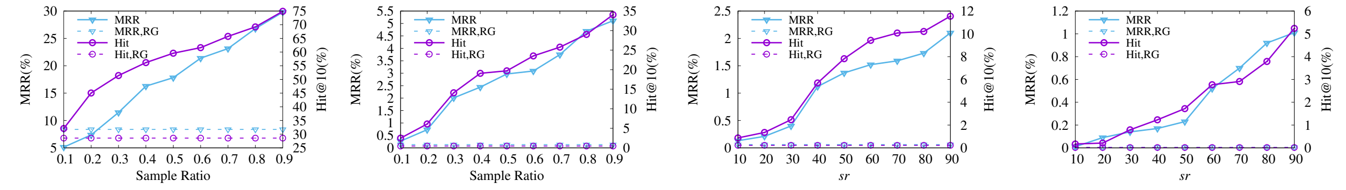
To validate our privacy inference method, we run PRA algorithm on the two datasets. Since the PRA code can easily take up upwards of 20GB of RAM for large scale graphs, we did our simulation on a sampled graph (3000 users for Pokec, 540 users for Google+). The profiles and relations are stored in the form of triples. For Pokec, the inference is performed separately without the involvement of de-anonymization. We randomly sample some triples as the training set at a sample ratio, which simulates the attacker's incomplete knowledge, and the rest triples act as ground truth. We also generate a few links for $\mathcal{E}_p^{AA}$ (attribute correlations) and add them to the train set. Given a query $\langle s, r\rangle$ (subject and relation), PRA outputs a waiting list of candidates for objects. *Mean reciprocal rank* (MRR), a criterion used in information retrieval, is adopted to measure the inference efficacy. We also modify *precision@k* and re-defined a metric *hit@k*. It refers to the ratio of queries who have a hit in the top k candidates, which better applies to our scenario since most relations in Pokec are functional. For Google+, we conduct privacy inference based on G'_p (the updated prior attack graph after de-anonymization). Performance is evaluated by varying the link sampling ratio sr.

For either dataset, an attributive and a relational link types are selected as secrecy. As shown in Fig. 13, our method performs better than random guess (RG) and is proportional to sample ratio (or sr), which indicates that the more prior knowledge the attacker possesses, the stronger inference ability she has. And the running time of our testing is only a few seconds. However, the performance on Google+ is undesirable because of two reasons: 1) G'_p contains false information due to imperfect de-anonymization conducted on the data set; 2) PRA relies strongly on topological information of the knowledge graph, but our graph sampling might have caused damage to that. However, the results are still much better than random guess, which verifies the feasibility of inferring privacy using knowledge graphs.

6 Discussion

Overlapped Graphs: So far we assumed that every user in G_p has a match in G_a, that is, $\mathcal{V}_p^U \subseteq \mathcal{V}_a^U$. But in reality, it is possible that a user in the attacker's prior knowledge does not exist in the original dataset, and thus not in G_a. In this case, Algorithm 1 can be adjusted slightly. Suppose $\pi : \mathcal{V}_p^U \to \mathcal{V}_a^U \cup \{\perp\}$ is an "injective" mapping from users in G_p to anonymized user nodes in G_a plus a no-match indicator $\perp$ (there can be multiple users in G_p mapped to $\perp$). When constructing the bipartite graph G_B, n_p fake nodes are added as candidates such that each node in $\mathcal{V}_p^U$ is linked to a different fake node with a weight w_0. One user would be mapped to a fake node if the weights of her links to other candidates are lower than w_0, *i.e.*, it is very likely she has no match in G_a. The key here is to select a proper weight threshold w_0.

Reducing Complexity: Our extensive evaluations show that the major time cost lies in the construction of the bipartite graph



(a) Pokec, “has weight” (b) Pokec, “follows” (c) Google+, “has job title” (d) Google+, “follows”

Fig. 13: **MRR and Hit@10 of privacy inference on Google+ and Pokec:** We infer users’ weights, job titles, and following. It is suggested that our method have better performance than random guess and is in proportion to the samping ratio.

G_B, *i.e.*, the set of links and the corresponding link weights. We proposed to use only top-k possible matching nodes for each node in the prior attack graph. The challenge is how to efficiently find the k candidates for each of nodes in G_p among n_a nodes from G_a without incurring large amount of computation. Besides, there is a linear time $1/2$-approximation algorithm for maximum weighted matching for general graphs [40], which can be utilized to further reduce the matching time.

Extending to Any RDF Graph: This paper models social network data using RDF triples and designs algorithms based on that. In fact, those formulations and algorithms also apply to any kind of RDF graph. Another work of us [46] presents how the attackers can use RDF graph to model their knowledge and perform privacy inference.

7 Related Work

The last decade witnesses large quantities of research works on privacy issues in social network data publishing. Various attack and protection techniques have been proposed. Most of them employ privacy models derived from k-anonymity [50], by assuming the attacker’s possession of specific limited prior knowledge. Unfortunately, their anonymization techniques are vulnerable to attackers with stronger background knowledge than what is assumed. For instance, k-degree anonymity [33] was proposed to prevent the attacker, who knows the number of neighbors of an individual, from identifying her from the published graph based on vertex degree. However, it cannot defend against community re-identification [52]. Similar anonymization techniques proposed in succession include k-neighborhood anonymity [54], [60] (against 1-neighborhood attack and 1^*-neighborhood attack, respectively), k-candidate anonymity [19], k-automorphism [62], k-isomorphism [10], k-structural diversity [52] (derived from l-diversity [34]), k^2-degree anonymity [51], and t-closeness [11]. Unfortunately, most of these proposals lack a complete or realistic attacker model. Zhou and Pei [60] showed that a person can be identified if the attacker has knowledge of the person’s neighbors and their attributes. Zou *et al.* [62] pointed out that the attacker may exploit multiple types of structural information (not just 1-hop neighborhood) to de-anonymize a person, so they proposed k-automorphism. By contrast, we assume the attacker aims at more than one victims, and she may know any profile and structural information about them. In addition, there are some anonymization techniques based on clustering/aggregation [7], [18], [32], [53], differential privacy [41], [44], [45], [47], [55], [59], and random walk [35]. However, they either are vulnerable to existing attacks or do not preserve data utility well.

Other previous methods focus on graph mapping attacks (also called structure-based de-anonymization), in which the attacker attempts to de-anonymize/re-identify users in the network, with only structural/toplogical information as background knowledge. Most of these attacks are seed-based, including [6], [12], [25], [28], [36], [37], [39], [49], [57]. They usually consist of two phases: seed identification and mapping propagation. In other words, a few specific users are identified somehow as seeds, and mapping users and nodes is iteratively conducted from the seed users based on structural characteristics of the data graph. There are also works that do not need seed users, such as [24], [38], which are based on Bayesian model and optimization respectively. These works do not consider the scenarios where some of the attacker’s background knowledge might be probabilistic. They attack the network data based on solely structural information, and their goal is to de-anonymize nodes in the network. On the contrary, our scheme utilizes both node profiles and topological information to infer users’ private attributes that are not contained in the anonymized network data.

There are also some methods that try to construct an attacker model. Hay *et al.* [18] classified adversary knowledge into three variants: vertex refinement queries, subgraph queries, and hub fingerprint queries. However, they either do not model the real capabilities of the attacker or express little adversarial knowledge. Narayanan *et al.* [36] assumed that the attacker has an auxiliary user network with both probabilistic aggregates and individual information, yet this model cannot capture some types of background information, like uncertain user relations. Chen *et al.* [31] considered three types of prior knowledge: positive associations, negative associations, and same-value families, but did not include relationships between individuals. Li *et al.* [32] modeled background knowledge as a graph with semantic edges and correlation edges. The graph is created by the data publisher based on her assumptions on the capability of the attacker, and it is used for designing anonymization methods. In contrast, our goal is to perform de-anonymization and the prior knowledge graph is created by the attacker.

8 Conclusion

In this work, we build a realistic and comprehensive model of the attacker’s background information with knowledge graphs, for better expressing attack process and quantifying privacy disclosure, which would provide a foundation for a generic anonymization technique. Our evaluations on two real-life social network datasets demonstrate its powerfulness and efficiency. There are a number of challenges left for future research. The first is to design an efficient method for constructing the bipartite graph for de-anonymization purpose. Second, we will include more features in matching nodes from the prior knowledge graph and nodes from the anonymized graph. The third is to subtly utilize knowledge correlations and probability distributions in the privacy inference process.

References

[1] 6 new facts about facebook, http://www.pewresearch.org/fact-tank/2014/02/03/6-new-facts-about-facebook/.

[2] Minimum cost flow algorithm, https://github.com/nikhilgarg28/libalgo/blob/master/MinCostFlow.java.

[3] Stanford large network dataset collection, https://snap.stanford.edu/data/.

[4] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. Network flows. Tech. rep., DTIC Document, 1988.

[5] Ahuja, R. K., Orlin, J. B., Stein, C., and Tarjan, R. E. Improved algorithms for bipartite network flow. *SICOMP 23*, 5 (1994), 906–933.

[6] Backstrom, L., Dwork, C., and Kleinberg, J. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *ACM WWW* (2007), pp. 181–190.

[7] Bhagat, S., Cormode, G., Krishnamurthy, B., and Srivastava, D. Class-based graph anonymization for social network data. *PVLDB 2*, 1 (2009), 766–777.

[8] Broder, A. Z. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings* (1997), IEEE, pp. 21–29.

[9] Charikar, M. S. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing* (2002), ACM, pp. 380–388.

[10] Cheng, J., Fu, A. W.-c., and Liu, J. K-isomorphism: privacy preserving network publication against structural attacks. In *SIGMOD* (2010), ACM, pp. 459–470.

[11] Chester, S., Kapron, B. M., Srivastava, G., and Venkatesh, S. Complexity of social network anonymization. *SNAM 3*, 2 (2013), 151–166.

[12] Chiasserini, C., Garetto, M., and Leonardi, E. De-anonymizing scale-free social networks by percolation graph matching. *arXiv* (2015).

[13] Clark Jr, G. C., and Cain, J. B. *Error-correction coding for digital communications*. Springer Science & Business Media, 2013.

[14] Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry* (2004), ACM, pp. 253–262.

[15] Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD* (2014), ACM, pp. 601–610.

[16] Gross, R., and Acquisti, A. Information revelation and privacy in online social networks. In *WPES* (2005), ACM, pp. 71–80.

[17] Hagenauer, J., and Lutz, E. Forward error correction coding for fading compensation in mobile satellite channels. *Selected Areas in Communications, IEEE Journal on 5*, 2 (1987), 215–225.

[18] Hay, M., Miklau, G., Jensen, D., Towsley, D., and Weis, P. Resisting structural re-identification in anonymized social networks. *PVLDB 1*, 1 (2008), 102–114.

[19] Hay, M., Miklau, G., Jensen, D., Weis, P., and Srivastava, S. Anonymizing social networks. *Computer Science Department Faculty Publication Series* (2007), 180.

[20] Indyk, P., and Motwani, R. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing* (1998), ACM, pp. 604–613.

[21] James, P. Knowledge graphs. *Order 501* (1992), 6439.

[22] Ji, S., Li, W., Gong, N. Z., Mittal, P., and Beyah, R. On your social network de-anonymizablity: Quantification and large scale evaluation with seed knowledge. *NDSS* (2015).

[23] Ji, S., Li, W., Mittal, P., Hu, X., and Beyah, R. Secgraph: A uniform and open-source evaluation system for graph data anonymization and de-anonymization. In *USENIX Security* (2015), USENIX Association.

[24] Ji, S., Li, W., Srivatsa, M., and Beyah, R. Structural data de-anonymization: Quantification, practice, and implications. In *ACM CCS* (2014), pp. 1040–1053.

[25] Ji, S., Li, W., Srivatsa, M., He, J. S., and Beyah, R. Structure based data de-anonymization of social networks and mobility traces. In *Information Security*. Springer, 2014, pp. 237–254.

[26] Jung, T., Li, X.-Y., Huang, W., Qian, J., Chen, L., Han, J., Hou, J., and Su, C. AccountTrade: Accountable protocols for big data trading against dishonest consumers. In *INFOCOM* (2017), IEEE.

[27] Kohlmayer, F., Prasser, F., Eckert, C., Kemper, A., and Kuhn, K. A. Flash: efficient, stable and optimal k-anonymity. In *IEEE PASSAT* (2012), pp. 708–717.

[28] Korula, N., and Lattanzi, S. An efficient reconciliation algorithm for social networks. *arXiv* (2013).

[29] Lao, N., Mitchell, T., and Cohen, W. W. Random walk inference and learning in a large scale knowledge base. In *EMNLP* (2011), Association for Computational Linguistics, pp. 529–539.

[30] Leskovec, J., Rajaraman, A., and Ullman, J. D. *Mining of massive datasets*. Cambridge University Press, 2014.

[31] Li, T., Li, N., and Zhang, J. Modeling and integrating background knowledge in data anonymization. In *ICDE* (2009), IEEE, pp. 6–17.

[32] Li, X.-Y., Zhang, C., Jung, T., Qian, J., and Chen, L. Graph-based privacy-preserving data publication. In *INFOCOM* (2016), IEEE.

[33] Liu, K., and Terzi, E. Towards identity anonymization on graphs. In *SIGMOD* (2008), ACM, pp. 93–106.

[34] Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M. l-diversity: Privacy beyond k-anonymity. *ACM TKDD 1*, 1 (2007), 3.

[35] Mittal, P., Papamanthou, C., and Song, D. Preserving link privacy in social network based systems. In *NDSS* (2013), ISOC.

[36] Narayanan, A., and Shmatikov, V. De-anonymizing social networks. In *IEEE S&P* (2009), pp. 173–187.

[37] Nilizadeh, S., Kapadia, A., and Ahn, Y.-Y. Community-enhanced de-anonymization of online social networks. In *ACM CCS* (2014), pp. 537–548.

[38] Pedarsani, P., Figueiredo, D. R., and Grossglauser, M. A bayesian method for matching two similar graphs without seeds. In *Allerton* (2013), pp. 1598–1607.

[39] Peng, W., Li, F., Zou, X., and Wu, J. A two-stage deanonymization attack against anonymized social networks. *TC 63*, 2 (2014), 290–303.

[40] Preis, R. Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs. In *STACS* (1999), Springer, pp. 259–269.

[41] Proserpio, D., Goldberg, S., and McSherry, F. Calibrating data to sensitivity in private data analysis. *arXiv* (2014).

[42] Pujara, J., Miao, H., Getoor, L., and Cohen, W. Knowledge graph identification. In *The Semantic Web–ISWC 2013*. Springer, 2013, pp. 542–557.

[43] Qian, J., Li, X.-Y., Zhang, C., and Chen, L. De-anonymizing social networks and inferring private attributes using knowledge graphs. In *INFOCOM* (2016), IEEE.

[44] Qian, J., Qiu, F., Wu, F., Ruan, N., Chen, G., and Tang, S. A differentially private selective aggregation scheme for online user behavior analysis. In *GLOBECOM* (2015), IEEE.

[45] Qian, J., Qiu, F., Wu, F., Ruan, N., Chen, G., and Tang, S. Privacy-preserving selective aggregation of online user behavior data. *IEEE Transactions on Computers 66*, 2 (2017), 326–338.

[46] Qian, J., Tang, S., Liu, H., Jung, T., and Li, X.-Y. Privacy inference on knowledge graphs: Hardness and approximation. In *MSN* (2017), IEEE.

[47] Sala, A., Zhao, X., Wilson, C., Zheng, H., and Zhao, B. Y. Sharing graphs using differentially private graph models. In *ACM Internet Measurement Conference* (2011), pp. 81–98.

[48] Shah, C., Capra, R., and Hansen, P. Collaborative information seeking: Guest editors' introduction. *IEEE TC 47*, 3 (2014), 22–25.

[49] Srivatsa, M., and Hicks, M. Deanonymizing mobility traces: Using social network as a side-channel. In *ACM CCS* (2012), pp. 628–637.

[50] Sweeney, L. k-anonymity: A model for protecting privacy. *IJUFKS 10*, 05 (2002), 557–570.

[51] Tai, C.-H., Yu, P. S., Yang, D.-N., and Chen, M.-S. Privacy-preserving social network publication against friendship attacks. In *SIGKDD* (2011), ACM, pp. 1262–1270.

[52] Tai, C.-H., Yu, P. S., Yang, D.-N., and Chen, M.-S. Structural diversity for privacy in publishing social networks. In *SDM* (2011), SIAM, pp. 35–46.

[53] Thompson, B., and Yao, D. The union-split algorithm and cluster-based anonymization of social networks. In *ACM ASIACCS* (2009), pp. 218–227.

[54] Wang, G., Liu, Q., Li, F., Yang, S., and Wu, J. Outsourcing privacy-preserving social networks to a cloud. In *IEEE INFOCOM* (2013), pp. 2886–2894.

[55] Xiao, Q., Chen, R., and Tan, K.-L. Differentially private network data release via structural inference. In *ACM SIGKDD* (2014), pp. 911–920.

[56] Xu, Z., Ramanathan, J., and Ramnath, R. Identifying knowledge brokers and their role in enterprise research through social media. *IEEE TC 47*, 3 (2014), 26–31.

[57] Yartseva, L., and Grossglauser, M. On the performance of percolation graph matching. In *COSN* (2013), ACM, pp. 119–130.

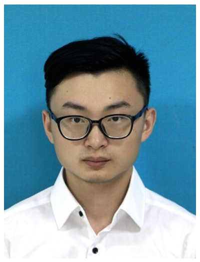
[58] YING, X., AND WU, X. Randomizing social networks: a spectrum preserving approach. In *Proceedings of the 2008 SIAM International Conference on Data Mining* (2008), SIAM, pp. 739–750.

[59] ZHANG, J., CORMODE, G., PROCOPIUC, C. M., SRIVASTAVA, D., AND XIAO, X. Privbayes: Private data release via bayesian networks. In *ACM SIGMOD* (2014), pp. 1423–1434.

[60] ZHOU, B., AND PEI, J. Preserving privacy in social networks against neighborhood attacks. In *IEEE ICDE* (2008), pp. 506–515.

[61] ZHOU, B., PEI, J., AND LUK, W. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explorations Newsletter 10*, 2 (2008), 12–22.

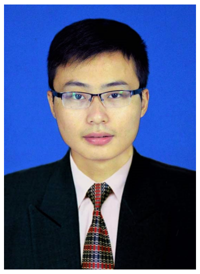
[62] ZOU, L., CHEN, L., AND ÖZSU, M. T. K-automorphism: A general framework for privacy preserving network publication. *PVLDB 2*, 1 (2009), 946–957.



Jianwei Qian is a Ph.D. student in the Department of Computer Science, Illinois Institute of Technology, USA. He received his B.E. degree in Computer Science and Technology from Shanghai Jiao Tong University, P.R. China, in 2015. His research interests include privacy and security issues in big data and social networking.

Dr. Xiang-Yang Li is a professor in School of Computer Science and Technology, University of Science and Technology of China. He was previously a professor at Computer Science Department at the Illinois Institute of Technology. Dr. Li received MS (2000) and PhD (2001) degree at Department of Computer Science from University of Illinois at Urbana-Champaign, a Bachelor degree at Department of Computer Science and a Bachelor degree at Department of Business Management from Tsinghua University, P.R. China, both in 1995. His research interests include wireless networking, mobile computing, security and privacy, cyber physical systems, and algorithms. He and his students won five best paper awards (IEEE GlobeCom 2015, IEEE HPCCC 2014, ACM MobiCom 2014, COCOON 2001, IEEE HICSS 2001), one best demo award (ACM MobiCom 2012). He published a monograph "Wireless Ad Hoc and Sensor Networks: Theory and Applications". He co-edited several books, including, "Encyclopedia of Algorithms". Dr. Li is an editor of several journals, including IEEE Transaction on Mobile Computing, and IEEE/ACM Transaction on Networking. He has served many international conferences in various capacities, including ACM MobiCom, ACM MobiHoc, IEEE MASS. He is an IEEE Fellow and an ACM Distinguished Scientist.

Dr. Chunhong Zhang is a lecturer of School of Information and Communication Engineering at Beijing University of Posts and Telecommunications. She received her PhD degree(2013) in Computer Science, Master degree(1996) in Information Technology, and Bachelor degree(1993) in Telecommunication Engineering from Beijing University of Posts and Telecommunications, P.R. China,. She was a visiting scholar at Illinois Institute of Technology in 2015. Her research interest includes data mining and privacy, ubiquitous computing, and distributed system. Her research has be supported by Ministry of Science and Technology of the People's Republic of China and National Natural Science Foundation of China. She has published 3 books and over 50 papers, and served for several international conferences.



Linlin Chen is a Ph.D student in the Department of Computer Science, Illinois Institute of Technology, USA. He received his B.E degree at Department of Computer Science and Techonlogy from the University of Science and Technology of China, P.R. China, in 2015. His research interests include privacy and security issues in big data, machine learning.

Taeho Jung received the B.E degree in Computer Software from Tsinghua University, Beijing, in 2011, and he is working toward the Ph.D degree in Computer Science at Illinois Institute of Technology while supervised by Dr. Xiang-Yang Li. His research area, in general, includes privacy & security issues in big data analysis and networking applications. Specifically, he is currently working on the privacy-preserving computation in various applications and scenarios where big data is involved.

Junze Han is a Ph.D. student in Computer Science at Illinois Institute of Technology since 2011. He received the B.E. degree from the Department of Computer Science at Nankai University, Tianjin, in 2011. His research interests include data privacy, mobile computing and wireless networking.