De-anonymizing Social Networks and Inferring Private Attributes Using Knowledge Graphs

Jianwei Qian*, Xiang-Yang Li‡†*, Chunhong Zhang§, Linlin Chen*
† School of Software, Tsinghua University
* Department of Computer Science, Illinois Institute of Technology
‡ School of Computer Science and Technology, University of Science and Technology of China
§ School of Information and Communication Engineering, Beijing University of Posts and Telecommunications

***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 strengthen 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 privacy preservation.**

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

I. 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 [10], [27], [37], [43]. 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 [5], [17], [19], [28].

The privacy issue in network data publishing is attracting increasing attention from researchers and social network providers [5], [15], [18], [24]. Various privacy attack and protection techniques have been proposed, including k-anonymity [39] based techniques (*e.g.*, k-degree anonymity [24]), and graph mapping based de-anonymization (*e.g.*, [15]). 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 [24]. Others assume that the attacker possesses a pure topological network (without user profiles) which overlaps with the published network data, such as [15]. 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. **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. The significance of our work lies in providing a better understanding of the attacker's prior knowledge and privacy inference, demonstrating the effectiveness of our method, alerting social network providers to privacy leakage risks, and leaving implications to researchers for designing general privacy protection (*e.g.*, anonymization) techniques.

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 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

978-1-4673-9953-1/16/$31.00 ©2016 IEEE

definition of privacy yet, let alone privacy disclosure's quantification.

In this paper, we will model the attacker's prior knowledge using knowledge graphs [13]. A confidence score is attached to each piece of knowledge to reflect the extent of the attacker's conviction. Then we use variations of the confidence scores to inflect the attacker's privacy inference 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.

2) We utilize this model to depict the privacy inferring process, *i.e.*, how an attacker de-anonymizes and infers users' private attributes, which helps to determine and measure privacy disclosure.

3) 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 perturbated 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.

The rest of this paper is organized as follows. Section II presents the preliminary concepts and the models to be used. We present the attack overview in Section III. In Section IV, we describe the de-anonymization and privacy inference technical details. We present our experiment evaluation results in Section V, discuss some limitations and challenges in Section VI, review the related work in Section VII, and conclude the paper in Section VIII.

II. Preliminaries and Modeling

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

A. Knowledge Graph

A knowledge graph [13] 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.*, $\langle$subject, predicate/relation, object$\rangle$, each representing a link. Note multiple links between two nodes are allowed, since there could be multiple relations between two entities. For example, $\langle \text{Tom Cruise, was born in, the USA} \rangle$, $\langle \text{Tom Cruise, has nationality, the USA} \rangle$. Actually, if the knowledge graph contains the triple $\langle s, r, o \rangle$, it can still contain $\langle s', r, o \rangle$, $\langle s, r', o \rangle$, $\langle s, r, o' \rangle$ [9]. Following Knowledge Vault [9], each link/triple is assigned a confidence score in our methods, which implies the probability of this knowledge being true.

B. 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. Thus, we have $\mathcal{V} = \mathcal{V}^U \cup \mathcal{V}^A$. Links in the knowledge graph convey a wide variety of information, which consist of three types: *user-to-user links*, *user-to-attribute links* and *attribute-to-attribute links*. Attribute-to-attribute links describe the correlations or some properties between attributes. Examples are $\langle \text{Bob, is colleague, Alice} \rangle$, $\langle \text{Bob, has gender, male} \rangle$, $\langle \text{Doctor, has salary, } > 50\text{K} \rangle$. Accordingly, we have $\mathcal{E} = \mathcal{E}^{UU} \cup \mathcal{E}^{UA} \cup \mathcal{E}^{AA}$.

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.

C. 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. 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.

D. 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 learned from her webpage and real life.
- *Network structural information e.g.*, node degrees, link relations, ego networks.

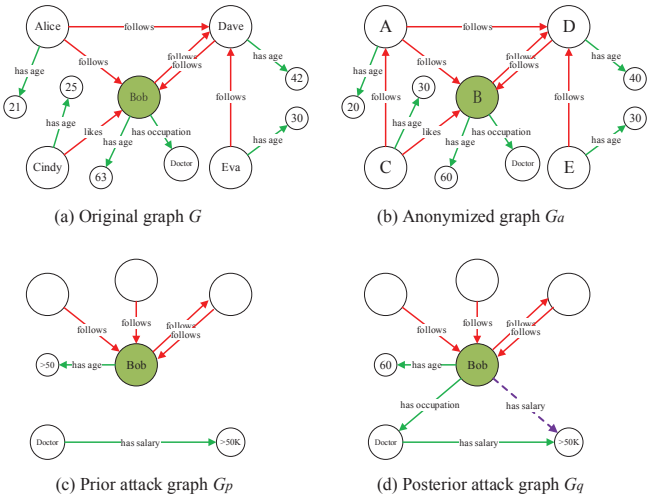
All of these information can be represented in a knowledge graph (including structural information, which can be seen as attributes), denoted by $G_p(\mathcal{V}_p, \mathcal{E}_p)$ (we call it *prior attack graph*). We assume the attacker targets at only a small number n_p of users from the global set $\mathcal{V}_a$ 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 [9], 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, $\langle$Jackie, has gender, female$\rangle$ and $\langle$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, $\langle$Jackie, lives in, New York$\rangle$ is included by $\langle$Jackie, lives in, the USA$\rangle$.
- *Soft correlation*. For instance, the fact $\langle$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*).



(a) Original graph G (b) Anonymized graph G_a

(c) Prior attack graph G_p (d) Posterior attack graph G_q

Fig. 1: An example of de-anonymization and privacy inference

III. 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.

A. 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 [34] 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*.

B. 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. 1 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. 1(a)) is published after anonymization and perturbations (Fig. 1(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. 1(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. 1(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 $\langle \text{Bob, has occupation, doctor} \rangle$ is added to the graph (Fig. 1(d)).

C. 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. 1(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. 1(d)) to her knowledge graph.

D. 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).

IV. 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.

A. 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 attributes similarity and the structural similarity scores.

Attribute Similarity: The attribute similarity score between two nodes $p \in \mathcal{V}_p^U$,$a \in \mathcal{V}_a^U$ is denoted as $S_A(p, a)$. 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,

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

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 $S_A(p, a) = \prod_{I(r,o)=1} c(\langle p, r, o\rangle) \cdot \prod_{I(r,o)=0} (1 - c(\langle p, r, o\rangle) c_0)$.

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.

The structural similarity between two nodes is

$$S_R(p, a) = w_i S_i(p, a) + w_o S_o(p, a), \tag{3}$$

where the weights w_i, w_o are set to 0.5 as default, and $S_i(p, a)$, $S_o(p, a)$ denote the inbound similarity and outbound similarity, computed 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

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

Therefore, the similarity S is a normalized function.

B. 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 VI). 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,

$$\operatorname{argmax}_{\pi} \operatorname{Sim}(G_p, G_a), \tag{5}$$

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.

$$\operatorname{Sim}_{\pi}(G_p, G_a) = \sum_{(i,j)\in\pi} S(i,j), \tag{6}$$

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 [3], [4]. We implement an algorithm based on [1], 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 $n_p << n_a$, 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.

In Algorithm 1 we build a light-weighted bipartite graph with the top-k strategy while traversing G_p. The intuition is that if two nodes match, their neighbors are also very likely to match. 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} (set to 0.8 as default), 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.

Algorithm 1 Bipartite Graph Construction Algorithm

Require: Anonymized graph G_a, prior attack graph G_p, parameter k.
Ensure: A bipartite graph G_B.

```
Define E_B = ∅, build a bipartite graph G_B(V_p^U + V_a^U, E_B).
Pick an initial node p_0 ∈ V_p^U.
Perform BFS in G_p starting from p_0 (treat G_p as undirected graph).
for each p ∈ V_p^U, following the order of BFS do
    if p has a predecessor pr(p) then
        Define N = ∅.
        for each a ∈ C_pr(p) do
            for each neighbor n of a do
                if The relation of n, a is the same as that of p, pr(p) then
                    N = N ∪ {n}.
        for each n ∈ N do
            Calculate S(p, n).
        Add top k similar nodes to C_p as p's candidates.
    else
        for each a ∈ V_a^U do
            Calculate S(p, a).
        Add top k similar nodes to C_p.
    for each a ∈ C_p do
        Attach weight S(p, a) to e_pa, E_B = E_B ∪ {e_pa}.
Remove isolated nodes in a ∈ V_a^U.
return G_B.
```

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 for p. 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.

Obviously, the choice of parameter k and the weight threshold s_{min} will have a large impact on the accuracy of the final mapping result. Small s_{min} (and large k) will result in an overblown bipartite graph with many unnecessary edges for computing the maximum weighted matching. On the other hand, large s_{min} (and small k) will 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 building a bipartite graph containing the optimum maximum weighted matching.

C. 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.* [22], 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.

V. 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.

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 I: Statistics of two datasets

A. Methodology

1) Datasets: We simulate our methods on two real world datasets, Google+ and Pokec, both from Stanford Network Analysis Project (SNAP) [2]. 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 I 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.

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 used in previous works [14], [15], [28]. 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. Since the Google+ profiles are not tabular, there exists few appropriate perturbation policies and algorithms for it. Thus, $\mathcal{E}_a^{UA}$ are also sampled from $\mathcal{E}^{UA}$ for Google+. For Pokec, we adopt the Flash algorithm [20] that achieves K-anonymity ($K = 10$, use capital K for disambiguation) to generalize the profiles. 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 $\mathcal{E}^{AA} = \mathcal{E}_a^{AA} = \emptyset$.

When we generate G_p, the sampling is slightly different. 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, we generate links for $\mathcal{E}_p^{AA}$ by calculating conditional probabilities between attributes ($Pr(c \mid b) = Pr(b, c)/Pr(b) = n(b, c)/n(b)$, for any $b, c \in \mathcal{V}^A$).

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.

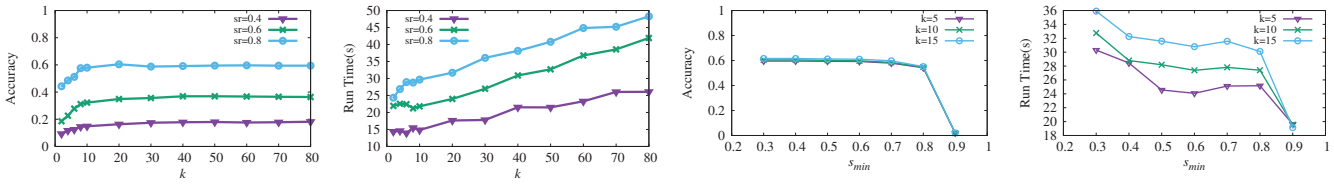
Later on, we will evaluate the influence of different user sampling methods on the attack performance.

B. Evaluation on De-anonymization

To evaluate our de-anonymization algorithm, we utilize accuracy (ratio of correct matches) and run time as metrics to measure utility and complexity. Since experiments on Google+ and Pokec are alike, we focus on Google+ unless there is a difference. The default parameter settings are $k = 10$, $s_{min} = 0.5$, $usr = 0.005$, $sr_a = sr_p = sr$, $sr = 0.8$, $w_A = 0.5$, and RW is chosen as the default user sampling method as it is most practical. We run all programs on Ubuntu 14.04.3 LTS on a server with Intel® Xeon® 2.4GHz 12-core CPU and 32GB memory.

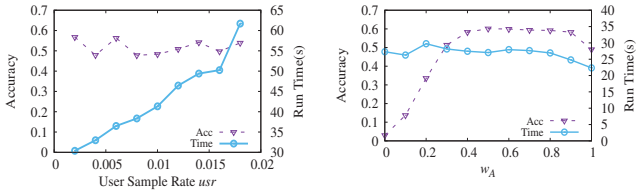
Fig. 2 effectively shows how the settings of k and s_{min} influence de-anonymization accuracy and time complexity. As depicted in Fig. 2(a), increasing k (recall that we match a user with the top k users from G_a in building the bipartite graph) can improve accuracy when $k \leq 10$ but has a minor effect when $k > 10$. This is because accuracy is bounded by sample ratio sr_a, sr_p (see details in Fig. 4). Yet run time constantly increases with the growing k (Fig. 2(b)). Therefore, we choose $k = 10$ as default. It is revealed in Fig. 2(c),2(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.

Fig. 3(a) 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. 3(b) 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



(a) Accuracy-k (b) Run Time-k (c) Accuracy-s_{min} (d) Run Time-s_{min}

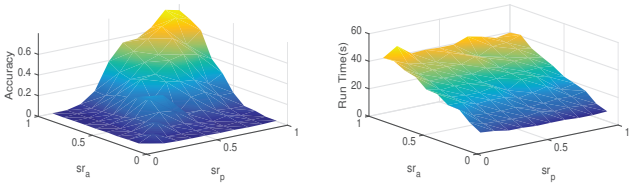
Fig. 2: The impact of parameters k, s_{min} on accuracy and runtime.



(a) Accuracy, Runtime vs usr (b) Accuracy vs w_A

Fig. 3: Impact of parameters on accuracy and time complexity

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.



(a) Accuracy (b) Run Time

Fig. 4: Accuracy and Run Time at Different sr_a, sr_p

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. 4(a), our de-anonymization method has larger accuracy when sr_a, sr_p are closer to 1. As shown in Fig. 4(b), sr_a has a more dominant effect on run time than sr_p, which can be explained by $n_p << n_a$. To be practical, we set $sr_a = sr_p = 0.8$.

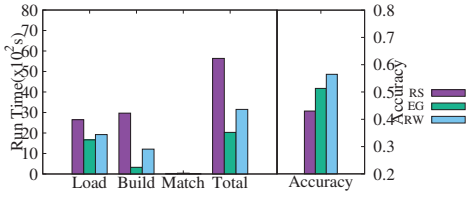


Fig. 5: Comparison of User Sampling Methods

After adjusting these parameters, we make a comparison of three different user sampling methods (Fig. 5). 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.

Tests on Pokec have similar results, except that the accuracy is lower (< 0.2). 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 such case, using more refined structure-based features will greatly improve the accuracy, such as ℓ-hop neighborhood, closeness centrality, Top-K reference distance, landmark reference distance [15]. We will incorporate them into our method as a future work.

C. 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 (3K 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. 6, 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. Besides, 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.

VI. 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 $\pi : \mathcal{V}_p^U \rightarrow \mathcal{V}_a^U \cup \{\perp\}$ 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 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 [32], which can be utilized to further reduce the matching time.

VII. 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 [39], 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 [24] 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 [41]. Similar anonymization techniques proposed in succession include k-neighborhood anonymity [42], [46] (against 1-neighborhood attack and 1^*-neighborhood attack, respectively), k-candidate anonymity [12], k-automorphism [47], k-isomorphism [6], k-structural diversity [41] (derived from l-diversity [25]), k^2-degree anonymity [40], and t-closeness [7]. Unfortunately, none of these proposals are built on a complete or realistic attacker model. In addition, there are some anonymization techniques based on clustering/aggregation [11], [23], differential privacy [33], [35], [36], [45]and random walk [26]. 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 [5], [8], [16], [21], [28], [29], [31], [38], [44]. 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 [15], [30], which are based on Bayesian model and optimization respectively. None of the above works consider the scenarios where some of the attacker's background knowledge might be probabilistic.

There are also some methods that try to construct an attacker model. Hay *et al.* [11] 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.* [28] 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.

VIII. Conclusion

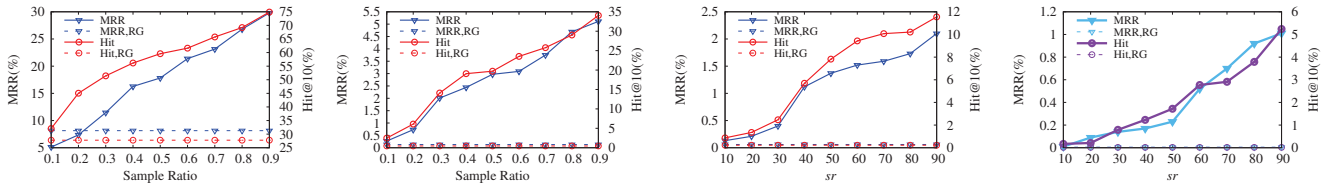
In this work, we build a realistic and comprehensive model of the attacker's background information with knowledge graphs, for better expressing attack process and quantifying privacy disclosure, which would provide a foundation for a generic anonymization technique. Our evaluations on two real-life 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.

Acknowledgment

This work is supported by NSF ECCS-1247944, NSF CMMI 1436786, NSF CNS 1526638, National Natural Science Foundation of China under Grant No. 61520106007. The opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the government.

References

[1] Minimum cost flow algorithm, https://github.com/nikhilgarg28/libalgo/blob/master/MinCostFlow.java.
[2] Stanford large network dataset collection, https://snap.stanford.edu/data/.
[3] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. Network flows. Tech. rep., DTIC Document, 1988.
[4] Ahuja, R. K., Orlin, J. B., Stein, C., and Tarjan, R. E. Improved algorithms for bipartite network flow. *SICOMP 23*, 5 (1994), 906–933.
[5] 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.



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

Fig. 6: MRR and Hit@10 of inference on Google+ and Pokec

[6] 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.

[7] CHESTER, S., KAPRON, B. M., SRIVASTAVA, G., AND VENKATESH, S. Complexity of social network anonymization. *SNAM 3*, 2 (2013), 151–166.

[8] CHIASSERINI, C., GARETTO, M., AND LEONARDI, E. De-anonymizing scale-free social networks by percolation graph matching. *arXiv* (2015).

[9] 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.

[10] GROSS, R., AND ACQUISTI, A. Information revelation and privacy in online social networks. In *WPES* (2005), ACM, pp. 71–80.

[11] 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.

[12] HAY, M., MIKLAU, G., JENSEN, D., WEIS, P., AND SRIVASTAVA, S. Anonymizing social networks. *Computer Science Department Faculty Publication Series* (2007), 180.

[13] JAMES, P. Knowledge graphs. *Order 501* (1992), 6439.

[14] 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).

[15] JI, S., LI, W., SRIVATSA, M., AND BEYAH, R. Structural data de-anonymization: Quantification, practice, and implications. In *ACM CCS* (2014), pp. 1040–1053.

[16] 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.

[17] JUNG, T., LI, X.-Y., AND WAN, M. Collusion-tolerable privacy-preserving sum and product calculation without secure channel. *TDSC 12*, 1 (2015), 45–57.

[18] JUNG, T., LI, X.-Y., WAN, Z., AND WAN, M. Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption. *TIFS 10*, 1 (2015), 190–199.

[19] JUNG, T., MAO, X., LI, X.-Y., TANG, S.-J., GONG, W., AND ZHANG, L. Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation. In *INFOCOM* (2013), IEEE, pp. 2634–2642.

[20] 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.

[21] KORULA, N., AND LATTANZI, S. An efficient reconciliation algorithm for social networks. *arXiv* (2013).

[22] 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.

[23] LI, X.-Y., ZHANG, C., JUNG, T., QIAN, J., AND CHEN, L. Graph-based privacy-preserving data publication. In *INFOCOM* (2015), IEEE.

[24] LIU, K., AND TERZI, E. Towards identity anonymization on graphs. In *SIGMOD* (2008), ACM, pp. 93–106.

[25] MACHANAVAJJHALA, A., KIFER, D., GEHRKE, J., AND VENKITASUBRAMANIAM, M. l-diversity: Privacy beyond k-anonymity. *ACM TKDD 1*, 1 (2007), 3.

[26] MITTAL, P., PAPAMANTHOU, C., AND SONG, D. Preserving link privacy in social network based systems. In *NDSS* (2013), ISOC.

[27] MOTAHARI, S., JUNG, T., ZANG, H., JANAKIRAMAN, K., LI, X.-Y., AND HOO, K. S. Predicting the influencers on wireless subscriber churn. In *WCNC* (2014), IEEE, pp. 3402–3407.

[28] NARAYANAN, A., AND SHMATIKOV, V. De-anonymizing social networks. In *IEEE S&P* (2009), pp. 173–187.

[29] NILIZADEH, S., KAPADIA, A., AND AHN, Y.-Y. Community-enhanced de-anonymization of online social networks. In *ACM CCS* (2014), pp. 537–548.

[30] PEDARSANI, P., FIGUEIREDO, D. R., AND GROSSGLAUSER, M. A bayesian method for matching two similar graphs without seeds. In *Allerton* (2013), pp. 1598–1607.

[31] PENG, W., LI, F., ZOU, X., AND WU, J. A two-stage deanonymization attack against anonymized social networks. *TC 63*, 2 (2014), 290–303.

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

[33] PROSERPIO, D., GOLDBERG, S., AND MCSHERRY, F. Calibrating data to sensitivity in private data analysis. *arXiv* (2014).

[34] PUJARA, J., MIAO, H., GETOOR, L., AND COHEN, W. Knowledge graph identification. In *The Semantic Web–ISWC 2013*. Springer, 2013, pp. 542–557.

[35] 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.

[36] 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.

[37] SHAH, C., CAPRA, R., AND HANSEN, P. Collaborative information seeking: Guest editors' introduction. *IEEE TC 47*, 3 (2014), 22–25.

[38] SRIVATSA, M., AND HICKS, M. Deanonymizing mobility traces: Using social network as a side-channel. In *ACM CCS* (2012), pp. 628–637.

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

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] YARTSEVA, L., AND GROSSGLAUSER, M. On the performance of percolation graph matching. In *COSN* (2013), ACM, pp. 119–130.

[45] ZHAO, J., JUNG, T., WANG, Y., AND LI, X.-Y. Achieving differential privacy of data disclosure in the smart grid. In *INFOCOM* (2014), IEEE, pp. 504–512.

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

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