Privacy-Enhanced Public View for Social Graphs

Hyoungshick Kim
Computer Laboratory,
University of Cambridge, UK
hk331@cl.cam.ac.uk

Joseph Bonneau
Computer Laboratory,
University of Cambridge, UK
jcb82@cl.cam.ac.uk

ABSTRACT

We consider the problem of releasing a limited *public view* of a sensitive graph which reveals at least k edges per node. We are motivated by Facebook's public search listings, which expose user profiles to search engines along with a fixed number of each user's friends. If this public view is produced by uniform random sampling, an adversary can accurately approximate many sensitive features of the original graph, including the degree of individual nodes. We propose several schemes to produce public views which hide degree information. We demonstrate the practicality of our schemes using real data and show that it is possible to mitigate inference of degree while still providing useful public views.

Categories and Subject Descriptors

K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*; E.1 [**Data Structures**]: Graphs and networks; H.2.8 [**Database Applications**]: Data mining; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security, Design, Theory, Experimentation

Keywords

Social networks, Public view, Privacy, Graph obfuscation

1. INTRODUCTION

The growing popularity of online social networks has encouraged new research into privacy protection. Much of the discussion has focused on the protection of users' personal information such as their photographs or beliefs. Simultaneously, data mining on large graphs has been intensively studied for the past two decades, discovering significant topological patterns in various complex networks and designing efficient analysis methods [5, 14, 4, 9]. The two threads of

research are coming together as the privacy of *social graphs* becomes a major concern [9], focusing on protecting network structure as well as personal information.

A social network is usually represented as an undirected graph, where nodes correspond to individuals and edges capture relationships between these individuals. Large-scale aggregation of this information is attractive to many third parties. For example, an advertiser may look at the profiles of a user's friends for use in targeted advertisements [10]. Despite the desire to limit access to data, networks may wish to show a limited *public view* for promotional purposes. In particular, Facebook implements "public search listings" which include a simple profile and a list of 8 friends.

At first glance, public search listings seem to be a compromise between privacy and utility, revealing an equal number of edges for each user. However, Bonneau at al. showed that public search listings leak critical information about the network structure [3]. This is because links associated with a given user are revealed not only by her own public list but also by all of her friends' public lists. An adversary can approximate many properties of the original graph by crawling all users' public search listings, which are freely available to those not logged in to the site.

In light of these challenges, we propose several schemes to produce public views which mitigate the exposure of degree information. We evaluate them on realistic data and find it is possible to improve privacy with a minimal loss of utility.

2. MODEL

We model a *social graph* as an undirected graph $G = (U, E)$, where the nodes U represent the users and the edges E represent social connections between users. In online social networks a node u represents a user profile and an edge e represents a declared friendship, while in an email system a node might represent an email address and an edge represents that a message has been sent between two addresses.

The network operator provides a *public view* of a graph which reveals a constant number of edges per node in the graph for promotional purposes. This is available to an adversary as a function $public\_neighbours_k(u)$, which return up to k neighbours of the u in the network. The adversary may not select k; it is a constant defined by the network operator.

This adversary can collect data about the network by repeatedly querying this interface and building up a *sampled graph* $\widehat{G} = (\widehat{U}, \widehat{E})$. Since some structural information must be included in a public view, there is an inherent trade-off between privacy and utility. We first define the goals of adversary and then discuss the utility of the public view.

Figure 1: An public search listings on Facebook

2.1 Threat Model

Online social networks play an important role as a medium for the spread of information among their members. In this context, Domingos and Richardson evaluated nodes' values in social networks for marketing purposes [13]. The optimisation problem of selecting the most influential nodes is NP-hard [8]. However, heuristic algorithms based on nodes' degree in G have been shown to perform well in practice. Thus, we consider degree information to be of high importance in selecting influential members of G [13]. Our adversary's goal is to select influential members of G as accurately as possible given the sampled graph $\widehat{G}$. To evaluate this goal, we define two metrics: *edge coverage* and *hub identification*.

2.1.1 Edge Coverage

The first metric is the ability to maximise the fraction of edges in the network covered by a *target set* of users $S \subseteq U$. For example, a marketer may wish to advertise their product to a few nodes within a network and maximise the reach of subsequent word-of-mouth promotion. An adversary's effectiveness can be defined in several ways such as the amount of node or edge coverage of the target set [10]. We use the notion of *edge coverage* which measures the number of network edges covered by the target set.

Definition 1. **Edge Coverage**– Given a graph $G = (U, E)$ and a target set of nodes $S \subseteq U$, we denote $E_S \subseteq U$ the set of all edges incident to at least one node in S and define the edge coverage of S as:

$$\text{EdgeCoverage}(S, G) = \frac{|E_S|}{|E|}$$

2.1.2 Hub Identification

The second metric measures the ability of the adversary to explicitly identify the highest-degree nodes in the original graph. We compute the ratio of nodes which are commonly included in the set of n top degree nodes in both of the original graph G and the sampled graph $\widehat{G}$.

Definition 2. **Hub Identification**– Given a graph $G = (U, E)$ and a sampled graph $\widehat{G} = (\widehat{U}, \widehat{E})$, we denote $H_n \subseteq U$ the set of n highest-degree nodes in G, and likewise $\widehat{H}_n \subseteq \widehat{U}$ the set of n highest-degree nodes in $\widehat{G}$. We then define the *hub identification* score as

$$\text{HubIdentification}_n(\widehat{G}, G) = \frac{|H_n \cap \widehat{H}_n|}{|H_n|}$$

2.2 Utility

The utility of a sampled graph is application dependent, but we propose extending the standard information retrieval notions of *precision* and *recall*. For each node $u \in G$, we define $d(u)$ as its degree in the original graph G, $\widehat{d}(u)$ as its degree in the sampled graph $\widehat{G}$, and $\widehat{d}_*(u)$ as the number of its incident edges in the sampled graph which truly exist in the original graph G.[1] We can then define precision and recall formally as:

Definition 3. **Precision**– Given two graphs $G = (U, E)$ and $\widehat{G} = (\widehat{U}, \widehat{E})$, the precision of $\widehat{G}$ is:

$$\text{P}(\widehat{G}, G) = \frac{1}{|\widehat{U}|} \cdot \sum_{u \in \widehat{U}} \frac{\widehat{d}_*(u)}{\widehat{d}(u)}$$

Definition 4. **Recall**– Given two graphs $G = (U, E)$ and $\widehat{G} = (\widehat{U}, \widehat{E})$, the recall of $\widehat{G}$ is:

$$\text{R}(\widehat{G}, G) = \frac{1}{|U|} \cdot \sum_{u \in U} \frac{\widehat{d}_*(u)}{d(u)}$$

Precision is the fraction of revealed edges which are accurate[2] and recall is the fraction of original edges which are revealed. We choose to average the scores for each node because we consider the important metric to be the expected precision and recall a user may get when they examine an individual node in the sampled graph.

If we gain no further utility by revealing more than k of a node's edges, we can define a *normalised recall* measure which is constant after k edges have been shown:

Definition 5. **Recall**$_k$– Given two graphs $G = (U, E)$ and $\widehat{G} = (\widehat{U}, \widehat{E})$, the recall$_k$ of $\widehat{G}$ is:

$$\text{R}_k(\widehat{G}, G) = \frac{1}{|U|} \cdot \sum_{u \in U} \frac{\min(\widehat{d}_*(u), k)}{\min(d(u), k)}$$

In all subgraphs which include no false edges, the precision of the sampled graph is exactly 1. The recall of a sampled graph will be 1 if and only if all edges are revealed. Only G itself as a sampled graph has both a precision and recall of 1. The normalised recall, however, will be 1 if and only if all nodes in the sampled graph have at least k edges (or, for nodes with less than k edges in the original graph, they retain all of their original edges).

3. UNIFORMLY RANDOM PUBLIC VIEW

We first consider the effectiveness of an adversary in our model if the graph is produced by uniform random sampling. That is, given a request for *public_neighbours$_k$(u)*, the network responds with a random selection of k of the u's friends, or returns all of their friends if $d(u) \leq k$. We assume that the network responds deterministically to each request, providing the same set of friends every time *public_neighbours$_k$(u)*

---

[1] $\widehat{d}(u) \neq \widehat{d}_*(u)$ iff the sampled graph includes false edges.

[2] Note that the summation for precision is over $\widehat{U}$ but for recall it is over U. This is because false nodes in $\widehat{G}$ reduce precision but do not affect recall.

is called, to prevent trivial attacks which assemble the entire graph by repeated queries.[3]

An adversary who compiles all such public listings creates the uniformly sampled graph G_k^{uniform} which has a precision of 1 and a normalised recall of 1. As shown previously, degree information is leaked by this sampled graph [3]. Only k friends will be shown in each user's listing, but high-degree users will show up in many other users' listings and thus remain high-degree nodes in the sampled graph $\widehat{G}$.

3.1 Theoretical Analysis

We can quantify this leakage formally by considering the sampled graph to be directed, with all edges learned from $public\_neighbours_k(u)$ originating from u. Let $E(u)$ be the set of u's adjacent edges in the graph G; We will use $E^{\mathrm{out}}(u)$ to denote the set of edges learned from $public\_neighbours_k(u)$, and $E^{\mathrm{in}}(u)$ to denote the set of u's adjacent edges learned from $public\_neighbours_k(v)$ for all $v \in \mathrm{neighbours}(u)$.

Our goal is to analyse the expected value of the degree of the node u, $\widehat{d}(u)$, in the sampled graph $\widehat{G}$:

$$
\begin{aligned}
\mathrm{E}[\widehat{d}(u)] &= \mathrm{E}[|E^{\mathrm{out}}(u) \cup E^{\mathrm{in}}(u)|] \quad (1) \\
&= \mathrm{E}[|E^{\mathrm{in}}(u)|] + \mathrm{E}[|E^{\mathrm{out}}(u) - E^{\mathrm{in}}(u)|] \\
&= \sum_{v \in \mathrm{Neighbours}(u)} \frac{|E^{\mathrm{out}}(v)|}{d(v)} + c_0(u) \\
& \text{where } c_0(u) = \mathrm{E}[|E^{\mathrm{out}}(u) - E^{\mathrm{in}}(u)|]
\end{aligned}
$$

If we assume that most nodes in the network have degree $\geq k$, we can further simplify by replacing $|E^{\mathrm{out}}(v)|$ with k, giving us the following approximation:

$$
\begin{aligned}
\mathrm{E}[\widehat{d}(u)] &\approx \sum_{v \in \mathrm{Neighbours}(u)} \frac{k}{d(v)} + c_0(u) \\
&= k \cdot d(u) \cdot c_1(u) + c_0(u) \\
& \text{where } c_1(u) \text{ is the average value of } \frac{1}{d(v)}
\end{aligned}
$$

If we approximate that all nodes' friend sets have degree distributions which are roughly even,[4] then we can approximate $c_1(u) = \frac{1}{\bar{d}}$ where $\bar{d}$ is the average degree for the graph. This is a constant independent of $d(u)$. Because $c_0(u)$ is at most k, it is dominated by the first term $k \cdot d(u) \cdot c_1(u)$ for high-degree nodes. Thus, we expect $\widehat{d}(u)$ to be roughly linearly proportional to u's true degree $d(u)$.

This explains why $\widehat{d}(u)$ was empirically found to be an effective approximation of $d(u)$ by Bonneau et al. [3]. They also proposed more complex approximations which attempt to individually approximate $c_1(u)$, but we find for our purposes that the simpler approximation is very effective.

3.2 Empirical Analysis

We supplement our theoretical analysis by measuring the accuracy of $\widehat{d}(u)$ in G_k^{uniform} on real social network data.

3.2.1 Experimental Data

We used a crawled dataset of 15,441 nodes and 620,075 edges representing students from the Columbia University sub-network of Facebook. The obtained data has an average degree of $\bar{d} = 80.3$, with minimum and maximum of $d_{\min} = 1$ and $d_{\max} = 1277$, and a median of $d_{\mathrm{med}} = 52$. We found that it fits a power law with $\alpha = 1.47$, consistent with previous larger-scale crawls [15, 12]. We will use this data set throughout our paper to analyse the utility and privacy implications of public views into social graphs.

3.2.2 Attacker Strategies

We evaluated the success of attacks seeking to use the public view of the original graph to select a target set S with maximal *edge coverage*. Given a size constraint $|S| \leq n$, finding the optimal S is NP-hard even if the underlying graph G is completely given [7]. This means that an adversary must use heuristics.

As noted by Korolova et al. [10], two obvious strategies are to pick the highest-degree node in each step (*highest-degree*), or to pick the node incident to the most uncovered edges in each step (*highest-uncovered-degree*). Though the second approach is superior given the complete graph, we found that it often performed worse in sampled graphs. For the remainder of our paper, we assume the attacker always uses the better of the two approaches.[5]
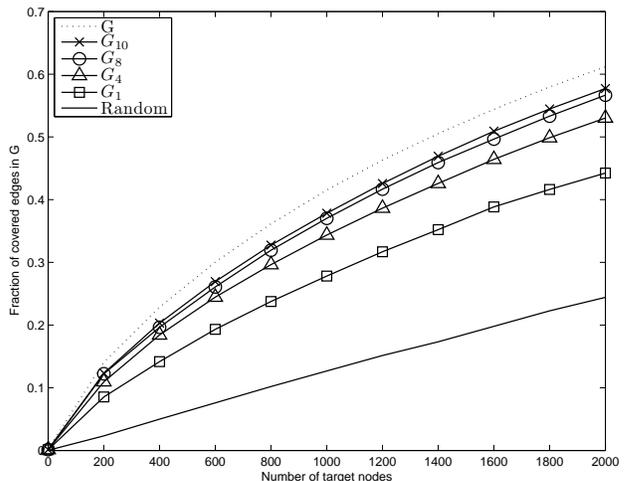
3.2.3 Comparison of Public List Size

Figure 2: Effect of public list size on edge coverage

We examined the effect of k on the success of edge-coverage attacks using G_k^{uniform}. The result is shown in Figure 2. We observe that the attack is still effective even if k is small when the public views are generated by uniform sampling. For the remainder of this paper, we will use $k = 8$, the value used in Facebook's public listings and a realistic value to provide the utility which network operators seek.

We compare the edge-coverage attacks against a strategy of randomly selecting nodes for S; this strategy is a lower

---

[3]In Facebook, for instance, there is a correlation between the set of friends shown and the geographic location of the requester's IP address [3].

[4]In large-scale web crawls, social networks have been found to have positive *assortativity*, meaning higher-degree nodes are more often friends with other high-degree nodes [15].

[5]*Highest-uncovered-degree* has the best performance for the original graph while *Highest-degree* is the best attack strategy for all of of our privacy-enhanced sampled graphs except "Regular subgraph extraction" in Section 4.4.

bound on information leakage since it can be used without any knowledge of the graph structure. Our goal in creating a privacy-enhanced view is to prevent an attacker from doing any better than random selection.

4. PRIVACY-ENHANCED PUBLIC VIEW

We now propose several methods for sampling the graph which limit the effectiveness of $\widehat{d}(u)$ as an approximations for $d(u)$.

4.1 Adding Noise

The simplest approach is to add noise to the public view by means of dummy edges between users who are not friends in the original graph. Since a user's friends can identify acquaintances (true edges) from strangers (dummy edges), while an adversary cannot, such added noises may not greatly diminish the utility of public search listings.

We denote as $G_k^{\text{dummy-}d}$ the sampled graph produced by showing k true edges and d dummy edges to each query of $public\_neighbours_k(u)$. $G_k^{\text{dummy-}0}$ is equivalent to G_k^{uniform}. Intuitively, this approach might seem to reduce the accuracy of inference. However, in examining Equation 1 derived in Section 3.1, we see that the dummy nodes in a given user's public list only add an additional constant term to the expected value of d_k, which will not affect edge coverage attacks which only consider the ranking of nodes by $\widehat{d}(u)$.

Useful noise is only produced in that nodes will appear as dummies in a random number of other users' public lists. However, this noise term is Gaussian and is based on a large number of samples ($\sim |U|$), giving it a very low variance and making this an ineffectual approach for slowing down inference of high-degree nodes.

4.2 Node Deletion

Another possibility is to remove some percentage p of nodes from the sampled graph, simply returning nothing for $public\_neighbours_k(u)$ if u is removed. We call such a graph $G_k^{\text{deleted-}p}$. $G_k^{\text{deleted-}0}$ is equivalent to G_k^{uniform}.

This approach is different from other methods in that the network operator may not be directly in control. It is certainly possible that the network itself could determine the p fraction of nodes which are uninteresting for marketing purposes and remove them centrally. More likely, however, is that the network operator may give users the ability to opt out of appearing in public listings.[6] It is possible that p could be increased by education efforts of the network operator or privacy awareness organisations.

4.3 Weighted Random Sampling

We observe that a high-degree node's degree is maintained in G_k^{uniform} because u is exposed uniformly at random in her friends' public search listings in proportion to $d(u)$. In order to offset $d(u)$, we can use weighted random sampling instead of uniform random sampling to break this proportionality. We call the sampled graph resulting from weighted random sampling G_k^{weighted}.

Under uniform random sampling, u is selected with the probability $\frac{k}{d(v)}$ for all $v \in \text{Neighbours}(u)$. In our weighted

random sampling, u is selected with the probability:

$$p_v(u) = \frac{k}{d(u) \cdot \sum_{w \in \text{Neighbours}(v)} \frac{1}{d(w)}}$$

Intuitively, u's low-degree friends will be selected with higher probability compared to her other high-degree friends to hide the disclosure of high-degree nodes. In this case, $\widehat{d}(u)$ is expected to be:

$$
\begin{aligned}
\mathrm{E}[\widehat{d}(u)] &\approx \sum_{v \in \text{Neighbours}(u)} p_v(u) + c_0(u) \\
&= \sum_{v \in \text{Neighbours}(u)} \frac{k}{d(u) \cdot d(v) \cdot c_2(u)} + c_0(u) \\
&= \frac{k}{d(u) \cdot c_2(u)} \cdot \sum_{v \in \text{Neighbours}(u)} \frac{1}{d(v)} + c_0(u) \\
&= \frac{k \cdot c_1(u)}{c_2(u)} + c_0(u)
\end{aligned}
$$

We define c_1 as in Section 3.1, and define a new term c_2 which is the expected value of $\frac{1}{d(w)}$ accross all of u's friends.

$$\mathrm{E}[\widehat{d}(u)] \approx \frac{k \cdot c_1(u)}{c_2(u)} + c_0(u)$$

If we assume $c_1(u)$ and $c_2(u)$ are roughly constant, then $\widehat{d}(u)$ is no longer linear in $d(u)$.[7] This weighted random sampling can be simply implemented in a distributed manner since it does not require the global network structure. Computing $public\_neighbours_k(u)$ only requires degree information for u's neighbours.

4.4 Regular Subgraph Extraction

The use of public listings is based on the (faulty) assumption that bounding the out-degree of a node prevents leaking $d(u)$ for the original graph. However, as we have shown, it is not effective to bound the in-degree of a node. Bounding the in-degree of a node is more difficult than bounding the out-degree since the edges for the in-degree may come from all of the node's neighbours where every node $u \in G$ has the same degree k. Since most nodes will have out-degree k, our best hope is to make this their total degree in the sampled graph by showing all revealed edges symmetrically in both users' public listings. To do so, we must produce a (nearly) k-regular subgraph from G by deleting or adding edges until all nodes have degree k and then publishing the resulting graph as a public listing.

Unfortunately, the general problem of finding the minimum number of edges whose deletion results in a subgraph with maximum degree k is NP-hard for any fixed $k \geq 2$ [16]. Thus, our goal will be to develop efficient heuristics which decrease the information leaked by a public view of social networks. We have developed a simple greedy algorithm which performs quite well on our sample data. We define the algorithm at three successive levels 0, 1, and 2, and denote the approximately k-regular graph produced as $G_k^{\text{regular-}i}$.

---

[6] Very few users exercise their right to opt-out of public-search listings: Bonneau et al. estimated it at $< 1\%$ [3].

[7] These assumptions fail in practice due to the assortativity of node's degrees, which causes $c_1(u)$ to correlate weakly with $d(u)$.

Algorithm 1 Greedy edge deletion

1: Intialise a priority queue $Q \leftarrow \emptyset$.
2: **for all** $e_{u,v} \in E$ **do**
3:    $priority[e_{u,v}] \leftarrow \min(d(u), d(v))$.
4:    Insert $e_{u,v}$ into Q.
5: **end for**
6: **while** Q is not empty **do**
7:    $e_{u,v} \leftarrow Q$.dequeueMax().
8:    **if** $d(u) > k$ **and** $d(v) > k$ **then**
9:       Delete $e_{u,v}$ in E.
10:      **for all** $e_{u,w} \in E$ **do**
11:         $priority[e_{u,w}] \leftarrow \min(d(u) - 1, d(w))$.
12:         Update $e_{u,w}$ in Q.
13:      **end for**
14:      **for all** $e_{w,v} \in E$ **do**
15:         $priority[e_{w,v}] \leftarrow \min(d(w), d(v) - 1)$.
16:         Update $e_{w,v}$ in Q.
17:      **end for**
18:    **end if**
19: **end while**

4.4.1 Level-0 Extraction

Our first pass will only delete edges from the graph, and furthermore will never reduce any node's degree below k. We do this by repeatedly deleting "safe" edges. A safe edge is one which connects two nodes each which have remaining degree greater than k, described in Algorithm 1.

To accomplish this, all edges are placed into a priority queue Q with priority equal to the minimum degree between two endpoints of the edge. The intuition is that removing an edge from a high-degree node doesn't reduce flexibility much, since they have many edges which can be removed. However, removing an edge where even one endpoint is low-order reduces flexibility, since all of that node's edges may be "frozen" from that point onwards if its degree drops to k.

Since this algorithm always produces a subgraph with at least k edges for all nodes with at least k in the original graph, it yields a precision and normalised recall of 1. It is a cautious algorithm in that it produces a public listing which is at least as useful as random sampling of edges.

4.4.2 Level-1 Extraction

After Level-0 extraction, some nodes will be left with more than k edges because all of their edges connect to nodes with remaining degree $\leq k$. If we remove this restriction, we can delete more edges and force all nodes in the sampled graph to have degree at most k. We call this "Level-1 Extraction", and it is achieved simply by changing the "safe" edge condition in Algorithm 1 as follows: An edge $e_{u,v}$ is deleted when either of the end points of $e_{u,v}$ has degree greater than k (i.e. line 8 in Algorithm 1 is modified as $d(u) \geq k$ or $d(v) \geq k$).

This approach will now be showing less than k edges in the sampled graph for some nodes which did have k edges in the original graph, thus there is a loss of recall.

4.4.3 Level-2 Extraction

Finally, to deal with the problem of low-degree nodes, either nodes which had degree less than k in the original graph, or nodes which had edges deleted during Level-1 Extraction, we add dummy edges to bring their degree up to k. This is achieved by a post-processing stage, Algorithm 2, which is run after finishing Level-1 Extraction. Algorithm 2 is a

simple randomised algorithm to do this. Let H be the nodes with remaining degree less than k. We add random dummy edges inside H until all nodes in H have exactly degree k.[8] After generating a new edge between two nodes u and v in H, if u or v has remaining degree equal to k the node is removed from H.

This approach will guarantee all nodes have exactly degree k in the sampled graph, at the cost of losing both precision and recall.

Algorithm 2 Random edge addition

1: Initialise $H \leftarrow \emptyset$.
2: **for all** $u \in U$ **do**
3:    **if** $d(u) < k$ **then**
4:       Insert u into H.
5:    **end if**
6: **end for**
7: **while** H is not empty **do**
8:    Randomly choose u and $v \in H$.
9:    **if** $e_{u,v} \notin E$ **then**
10:      Insert $e_{u,v}$ into E.
11:    **end if**
12:    **if** $d(u) = k$ **then**
13:      Delete u in H.
14:    **end if**
15:    **if** $d(v) = k$ **then**
16:      Delete v in H.
17:    **end if**
18: **end while**

---

5. EXPERIMENTAL RESULTS

We analyse the performance of each approach proposed in Section 4 on our real-world dataset. We used simulations to measure edge coverage (Def. 1) of a set of target nodes chosen by the adversary using the sampled graph, shown in Figure 3. For comparison, we plotted each approach against G_8^{uniform} as an upper bound on leakage of degree information, G_1^{uniform} as a lower bound on uniform random sampling, and random selection as an absolute lower bound.

We analysed the efficiency of the hub identification attack in Table 1. For each public view, we calculated the hub identification score (Def. 2) for multiple values of n, and compared them with G_8^{uniform}, G_1^{uniform}, and random selection.

We also list the resulting degree distribution for each public view in Table 2, to make possible quantitative analysis of the effects on the graph of each sampling method.

Finally, we compare the utilities of the public views in Table 3, again comparing with G_8^{uniform} and random selection.

Our goal was to produce a public view with significantly enhanced privacy without seriously diminishing the utility of the feature. Based on this criteria, adding dummy edges is not a practical solution. Showing an equal number of dummy edges and true edges produced almost no decrease in edge coverage or hub identification achievable by the attacker, while causing an unacceptable loss of precision. Even adding 5 times as many dummy edges as true edges, the

---

[8]In the case that k is odd and $|U|$ is odd, there will remain a single node whose degree is $k - 1$ because the sum of all nodes' degrees must be even.

(a) Dummy edges

(b) Node deletion

(c) Weighted sampling
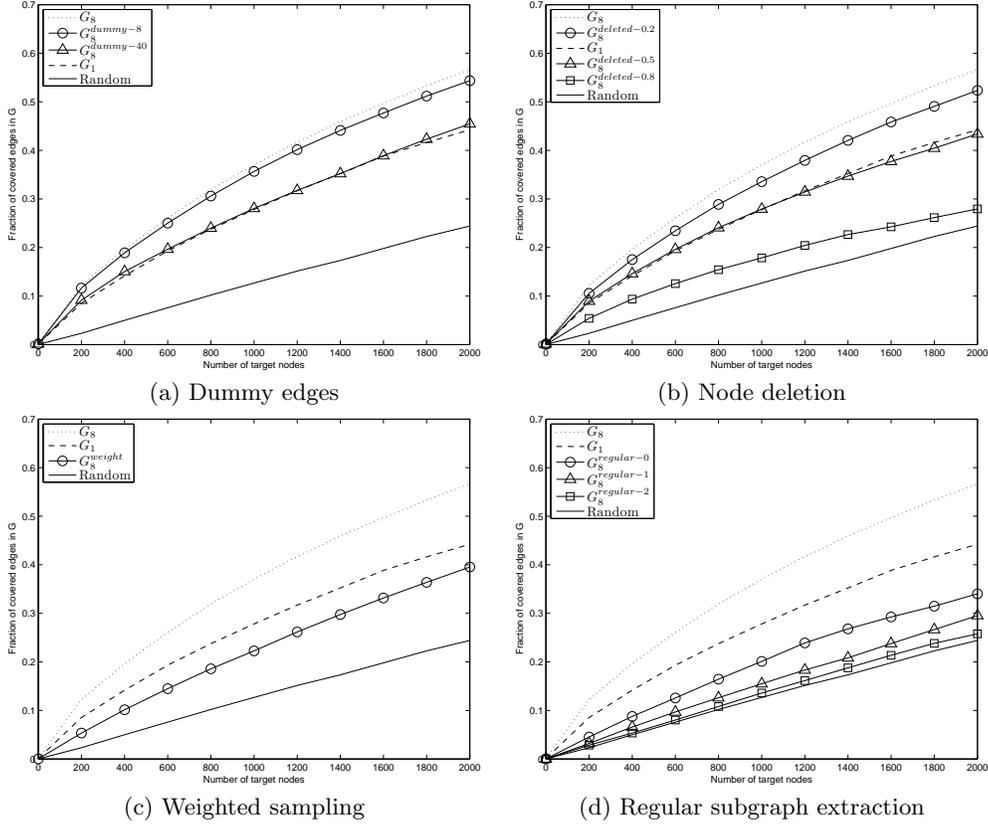
(d) Regular subgraph extraction

Figure 3: Leakage of public views against edge coverage attacks

constructed sampled graph includes significant information about node degree distribution.

We similarly reject node deletion. Deleting 20% of nodes provided little decrease in edge coverage or hub identification achievable by the attacker, but caused a large decrease in recall. The approach only begins to have a significant effect at unrealistically high values such as p=80%.

Weighted random sampling performed much better, with G_8^{weighted} leaking less information than G_1^{uniform} without any loss of utility. By examining the degree distribution of G_8^{weighted} in Table 2, we can see that the vast majority of nodes have a $\widehat{d}(u) \leq 20$, unlike G_8^{uniform} which still contains many high-degree nodes. G_8^{weighted} is in this sense a weak approximation of regular-graph extraction. Given that it does not reduce utility and comes at almost zero computational cost, this method should be the minimum step taken to reduce information leakage in public listings.

The regular subgraph extraction algorithm provides the best results. Even using $G_8^{\text{regular-0}}$, which does not reduce precision or recall, both edge coverage and hub identification are significantly less efficient than against G_1^{uniform} or G_8^{weighted}, roughly halving the gap between these public views and the lower bound of random selection. $G_8^{\text{regular-1}}$ and $G_8^{\text{regular-2}}$ each roughly halve this gap again, with $G_8^{\text{regular-2}}$ providing near-perfect resistance to hub identification and doing slightly worse than random against edge coverage.[9]

We can see from the degree distribution that $G_8^{\text{regular-2}}$ is

---

[9]In this case, the attacker does better than random only by using the *highest-uncovered-degree* strategy.

a perfectly 8-regular subgraph. To achieve this, precision drops to 90%. This is largely unavoidable as in the original graph there are a significant number of low-degree nodes, and edges must be added if these are to be covered. We also observe that $G_8^{\text{regular-0}}$ is a good approximation, as the vast majority, more than 99.9% of nodes, are of degree 9 or less.

However, despite the success of this algorithm, it may not be universally applicable. Computing it requires central knowledge of the complete graph, and the running time of $O(|E|\log|E|)$ may be impractical for very large graphs, especially because it must be re-computed after every node-addition or removal. In contrast, weighted sampling can be efficiently computed as needed.

6. LIMITATIONS

Our work is primarily intended to demonstrate that it is possible to reveal a public view of a graph in a more privacy-sensitive manner than uniform random sampling. We note that we have only considered resistance to inferring nodes' degrees. There are many other graph properties which attackers may be interested in such as computing small dominating sets or clustering the group into communities. Our initial experiments show for example that even $G_8^{\text{regular-2}}$ badly fails to prevent calculation of a small dominating set in the original graph. We also only consider simplistic attacks using the apparent degree $\widehat{d}(u)$ as an approximation for $d(u)$. It has already been shown that better methods exist which utilise more properties of the sampled graph [3], and sophisticated attack strategies may exist which extract

	G_8^{uniform}	G_1^{uniform}	G_8^{weight}	$G_8^{\text{dummy-8}}$	$G_8^{\text{deleted-0.2}}$	$G_8^{\text{regular-0}}$	$G_8^{\text{regular-1}}$	$G_8^{\text{regular-2}}$	Random
$n = 200$	0.53	0.28	0.08	0.46	0.39	0.06	0.00	0.01	0.01
$n = 400$	0.53	0.25	0.08	0.50	0.36	0.08	0.02	0.02	0.01
$n = 600$	0.53	0.25	0.09	0.48	0.41	0.07	0.03	0.03	0.01
$n = 800$	0.51	0.27	0.10	0.49	0.42	0.08	0.04	0.03	0.03
$n = 1000$	0.54	0.28	0.09	0.49	0.42	0.09	0.04	0.05	0.04
$n = 1200$	0.54	0.29	0.11	0.50	0.44	0.11	0.06	0.05	0.05
$n = 1400$	0.56	0.29	0.13	0.50	0.46	0.13	0.06	0.06	0.05
$n = 1600$	0.57	0.30	0.14	0.49	0.47	0.13	0.07	0.07	0.06
$n = 1800$	0.58	0.30	0.14	0.50	0.48	0.15	0.08	0.08	0.06
$n = 2000$	0.58	0.31	0.15	0.51	0.47	0.16	0.09	0.08	0.07

Table 1: Hub identification comparison of public views

	G	G_8^{uniform}	G_8^{weight}	$G_8^{\text{dummy-8}}$	$G_8^{\text{deleted-0.2}}$	$G_8^{\text{regular-0}}$	$G_8^{\text{regular-1}}$	$G_8^{\text{regular-2}}$
$\#(u) : d(u) = 1$	558	522	558	0	536	558	558	0
$\#(u) : d(u) = 2$	415	395	415	0	364	415	420	0
$\#(u) : d(u) = 3$	366	368	366	0	344	366	368	0
$\#(u) : d(u) = 4$	321	308	321	0	338	321	319	0
$\#(u) : d(u) = 5$	275	285	275	0	446	275	286	0
$\#(u) : d(u) = 6$	248	420	248	0	632	248	265	0
$\#(u) : d(u) = 7$	261	689	261	0	856	261	1482	0
$\#(u) : d(u) = 8$	253	1013	592	0	1039	11765	11743	15441
$\#(u) : d(u) = 9$	245	1003	664	1	1031	1208	0	0
$\#(u) : d(u) = 10$	213	1042	782	2	977	10	0	0
$\#(u) : d(u) = 11$	193	1041	1072	15	908	3	0	0
$\#(u) : d(u) = 12$	199	1057	1417	29	816	3	0	0
$\#(u) : d(u) = 13$	178	993	1542	78	691	2	0	0
$\#(u) : d(u) = 14$	177	936	1689	103	635	1	0	0
$\#(u) : d(u) = 15$	147	891	1464	158	577	0	0	0
$\#(u) : d(u) = 16$	162	768	1291	213	411	1	0	0
$\#(u) : d(u) = 17$	134	657	905	230	382	1	0	0
$\#(u) : d(u) = 18$	125	620	698	284	272	2	0	0
$\#(u) : d(u) = 19$	136	556	384	359	215	0	0	0
$\#(u) : d(u) = 20$	132	428	192	371	175	0	0	0
$\#(u) : d(u) \geq 21$	10559	2057	305	13598	610	1	0	0

Table 2: Degree distribution comparison of public views

	G	G_8^{uniform}	G_8^{weight}	$G_8^{\text{dummy-8}}$	$G_8^{\text{deleted-0.2}}$	$G_8^{\text{regular-0}}$	$G_8^{\text{regular-1}}$	$G_8^{\text{regular-2}}$
Precision	1	1	1	0.43	1	1	1	0.90
Recall	1	0.42	0.42	0.42	0.34	0.34	0.34	0.34
Recall$_8$	1	1	1	1	0.94	1	0.99	0.99

Table 3: Utility comparison of public views

more degree information based on latent patterns in our regular graph $G_8^{\text{regular-2}}$. We consider extending our work to other threat models and improving inference algorithms as important lines for future work.

7. RELATED WORK

Social graph privacy has seen increasing interest from the data mining and theory communities. Korolova et al. [10] discuss the problem of an adversary who wants to derive the link structure of the network by collecting neighbourhood information of some users who are bribed by the adversary. Their analysis shows that the number of compromised users needed to cover a constant fraction of the network drops exponentially as amount of graph information available to each node increases. Their analysis was followed by Bonneau et al. who demonstrated the practicality of recovering large portions of the Facebook graph [2]. In a separate work, Bonneau at al. [3] focused on Facebook's public listing feature as a means of graph inference, showing experimentally that several sensitive properties of a social graph can be inferred.

Several research efforts have focused on the difficulty of anonymising social graphs by replacing true node identities with pseudonyms. Backstrom et al. [1] introduced de-anonymisation attacks against an anonymised graph where true node identities are replaced with pseudonyms. To prevent these attacks, algorithms based on graph modification such as addition or deletion of edges are proposed to reduce the risk of sensitive link disclosure [6].

This work is also related to the theory of graph sampling. Previous research, however, has focused on sampling from original graphs so that some graph properties such as nodes' degree distribution are preserved with a small sample [11], which is opposite to our goal of obfuscating some important network structure within a sampled graph.

8. CONCLUSIONS

In this paper, we proposed several schemes to increase the privacy of public views into a social graph. Public lists generated by uniform random sampling leak degree information strongly since they cannot control the incoming edges of each node. We proposed several schemes for hiding degree information in the public view while still being useful for promotional purposes.

We found that the intuitive solutions of adding noise by inserting dummy edges or deleting nodes are ineffective. Instead, we found that the best solution is carefully selecting listings to make the public view resemble a regular graph. This can be approximated cheaply by weighted random sampling, or computed centrally by extracting a nearly-regular subgraph. Both solutions significantly reduce leakage while providing similar utility.

9. REFERENCES

[1] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore Art Thou R3579x?: Anonymized Social Networks, Hidden Patterns, and Structural Steganography. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 181–190, New York, NY, USA, 2007.

[2] J. Bonneau, J. Anderson, and G. Danezis. Prying Data out of a Social Network. *ASONAM 2009 : Advances in Social Networks Analysis and Mining*, 2009.

[3] J. Bonneau, J. Anderson, F. Stajano, and R. Anderson. Eight Friends are Enough: Social Graph Approximation via Public Listings. In *the 2nd ACM Workshop on Social Network Systems*, 2009.

[4] L. da F. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas. Characterization of Complex Networks: A Survey of Measurements. *Advances In Physics*, 56:167, 2007.

[5] M. Girvan and M. E. Newman. Community Structure in Social and Biological Networks. *Proc Natl Acad Sci USA*, 99(12):7821–7826, June 2002.

[6] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis. Resisting Structural Re-identification in Anonymized Social Networks. *Proc. VLDB Endow.*, 1(1):102–114, 2008.

[7] R. M. Karp. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[8] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the Spread of Influence through a Social Network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, New York, NY, USA, 2003.

[9] J. M. Kleinberg. Challenges in Mining Social Network Data: Processes, Privacy, and Paradoxes. In *KDD '07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 4–5, New York, NY, USA, 2007.

[10] A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu. Link Privacy in Social Networks. In *CIKM '08: Proceeding of the 17th ACM Conference on Information and Knowledge Management*, pages 289–298, New York, NY, USA, 2008.

[11] J. Leskovec and C. Faloutsos. Sampling from Large Graphs. In *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 631–636, New York, NY, USA, 2006.

[12] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and Analysis of Online Social Networks. In *IMC '07: Proceedings of the 7th ACM SIGCOMM Conference on Internet measurement*, pages 29–42, 2007.

[13] Matthew Richardson and Pedro Domingos. Mining Knowledge-sharing Sites for Viral Marketing. In *KDD '02: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 61–70, New York, NY, USA, 2002.

[14] S. H. Strogatz. Exploring Complex Networks. *Nature*, 410(6825):268–276, March 2001.

[15] C. Wilson, B. Boe, A. Sala, K. P. N. Puttaswamy, and B. Y. Zhao. User Interactions in Social Networks and their Implications. *EuroSys 2009 : Proceedings of the ACM EuroSys Conference*, 2009.

[16] M. Yannakakis. Node-and Edge-deletion NP-complete Problems. In *STOC '78: Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 253–264, New York, NY, USA, 1978.