Preserving Link Privacy in Social Network Based Systems

Prateek Mittal
University of California, Berkeley
pmittal@eecs.berkeley.edu

Charalampos Papamanthou
University of California, Berkeley
cpap@eecs.berkeley.edu

Dawn Song
University of California, Berkeley
dawnsong@cs.berkeley.edu

Abstract

A growing body of research leverages social network based trust relationships to improve the functionality of the system. However, these systems expose users' trust relationships, which is considered sensitive information in today's society, to an adversary.

In this work, we make the following contributions. First, we propose an algorithm that perturbs the structure of a social graph in order to provide link privacy, at the cost of slight reduction in the utility of the social graph. Second we define general metrics for characterizing the utility and privacy of perturbed graphs. Third, we evaluate the utility and privacy of our proposed algorithm using real world social graphs. Finally, we demonstrate the applicability of our perturbation algorithm on a broad range of secure systems, including Sybil defenses and secure routing.

1 Introduction

In recent years, several proposals have been put forward that leverage user's social network trust relationships to improve system security and privacy. Social networks have been used for Sybil defense [36, 35, 6, 23, 31], secure routing [14, 21, 17], secure reputation systems [32], mitigating spam [19], censorship resistance [30], and anonymous communication [5, 25, 20, 22].

A significant barrier to the deployment of these systems is that they do not protect the privacy of user's trusted social contacts. Information about user's trust relationships is considered sensitive in today's society; in fact, existing online social networks such as Facebook, Google+ and Linkedin provide explicit mechanisms to limit access to this information. A recent study by Dey et al. [7] found that more than 52% of Facebook users hide their social contact information.

Most protocols that leverage social networks for system security and privacy either explicitly reveal users' trust relationships to an adversary [6] or allow the adversary to easily perform traffic analysis and infer these trust relationships [35]. Thus the design of these systems is fundamentally in conflict with the current online social network paradigm, and hinders deployment.

In this work, we focus on protecting the privacy of users' trusted contacts (edge/link privacy, not vertex privacy) while still maintaining the utility of higher level systems and applications that leverage the social graph. Our key insight in this work is that for a large class of security applications that leverage social relationships, preserving the exact set of edges in the graph is not as important as preserving the graph-theoretic structural differences between the honest and dishonest users in the system.

This insight motivates a paradigm of *structured graph perturbation*, in which we introduce noise in the social graph (by deleting real edges and introducing fake edges) such that the local structures in the original social graph are preserved. We believe that for many applications, introducing a high level of noise in such a structured fashion does not reduce the overall system utility.

1.1 Contributions

In this work, we make the following contributions.

- First, we propose a mechanism based on random walks for perturbing the structure of the social graph that provides link privacy at the cost of a slight reduction in application utility (Section 4).
- We define a general metric for characterizing the utility of perturbed graphs. Our utility definition considers the change in graph structure from the perspective of a vertex. We formally relate our notion of utility to global properties of social graphs, such as mixing times and second largest eigenvalue

modulus of graphs, and analyze the utility properties of our perturbation mechanism using real world social networks (Section 5).

- We define several metrics for characterizing link privacy, and consider prior information that an adversary may have for de-anonymizing links. We also formalize the relationship between utility and privacy of perturbed graphs, and analyze the privacy properties of our perturbation mechanism using real world social networks (Section 6).

- Finally, we experimentally demonstrate the real world applicability of our perturbation mechanism on a broad range of secure systems, including Sybil defenses and secure routing (Section 7). In fact, we find that for Sybil defenses, our techniques are of interest even outside the context of link privacy.

2 Related Work

Work in this space can be broadly classified into two categories: (a) mechanisms for protecting the privacy of links between labeled vertices, and (b) mechanisms for protecting node/graph privacy when vertices are unlabeled.

For many real world applications relying on social networks, such as reputation systems, recommendation systems, Sybil defenses, secure routing, and anonymous communications, it is critical for a user to know the identities of the vertices in the social graph that it interacts with. Protocols that preserve vertex privacy (unlabeled vertices) would not be appropriate for such applications. On the other hand, these applications could reveal sensitive social contacts to an adversary; thus the focus of this work is on protecting the privacy of links between labeled vertices.

2.1 Link privacy between labeled vertices

There are two main mechanisms for preserving link privacy between labeled vertices. The first approach is to perform clustering of vertices and edges, and aggregate them into super vertices (e.g., [11] and [37]). In this way, information about corresponding sub-graphs can be anonymized. While these clustering approaches permit analysis of some macro-level graph properties, they are not suitable for black-box application of existing social network based applications, such as Sybil defenses. The second class of approaches aim to introduce perturbation in the social graph by adding and deleting edges and vertices. Next, we discuss this line of research in more detail.

Hay et al. [12] propose a perturbation algorithm which applies a sequence of k edge deletions followed by k random edge insertions. Candidates for edge deletion are sampled uniformly at random from the space of existing edges in graph G, while candidates for edge insertion are sampled uniformly at *random* from the space of edges not in G. The key difference between our perturbation mechanism and that of Hay et al. is that we sample edges for insertion based on the *structure* of the original graph (as opposed to random selection). We will compare our approach with that of Hay et al. in Section 6.

Ying and Wu [34] study the impact of Hay et al.'s perturbation algorithms [12] on the spectral properties of graphs, as well as on link privacy. They also propose a new perturbation algorithm that aims to preserve the spectral properties of graphs, but do not analyze its privacy properties.

Korolova et al. [13] show that link privacy of the overall social network can be breached even if information about the local neighborhood of social network nodes is leaked (for example, via a look-ahead feature for friend discovery).

2.2 Anonymizing the vertices

Although the techniques described above reveal the *identity* of the vertices in the social graph but add noise to the *relationships* between them, there have been various works in the literature that aim at anonymizing the identities of the nodes in the social network. This line of research is orthogonal to our goals, but we describe them for completeness.

The straightforward approach of just removing the identifiers of the nodes before publishing the social graph does not always guarantee privacy, as shown by Backstrom et. al. [2]. To deal with this problem, Liu and Terzi [15] propose a systematic framework for identity anonymization on graphs, where they introduce the notion of k-degree anonymity. Their goal is to minimally modify the graph by changing the degrees of specially-chosen nodes so that the identity of each individual involved is protected. An efficient version of their algorithm was recently implemented by Lu et al. [16].

Another notion of graph anonymity in social networks is presented by Pei and Zhou [38]: A graph is k-anonymous if for every node there exist at least $k-1$ other nodes that share isomorphic neighborhoods. This is a stronger definition than the one in [15], where only vertex degrees are considered.

Zhou and Pei [39] recently introduced another notion called l-diversity for social network anonymiza-

tion. In this case, each vertex is associated with some non-sensitive attributes and some sensitive attributes. Maintaining the privacy of the individual in this scenario is based on the adversary not being able (with high probability) to re-identify the sensitive attribute values of the individual.

Finally, Narayanan and Shmatikov [26] show some of the weaknesses of the above anonymization techniques, propose a generic way for modeling the release of anonymized social networks and report on successful de-anonymization attacks on popular networks such as Flickr and Twitter.

2.3 Differential privacy and social networks

Sala et al. [29] use differential privacy (a more elaborate tool of adding noise) to publish social networks with privacy guarantees. Given a social network and a desired level of differential privacy guarantee, they extract a detailed structure into degree correlation statistics, introduce noise into these statistics, and use them to generate a new synthetic social network with differential privacy. However, their approach does not preserve utility of the social graph from the perspective of a vertex (since vertices in their graph are unlabeled), and thus cannot be used for many real world applications such as Sybil defenses.

Also, Rastogi et al. [28] introduce a relaxed notion of differential privacy for data with relationships so that more expressive queries (e.g., joins) can be supported without significant harm to utility.

2.4 Link privacy preserving applications

X-Vine [21] proposes to perform DHT routing using social links, in a manner that preserves the privacy of social links. However, the threat model in X-Vine excludes adversaries that have prior information about the social graph. Thus in real world settings, X-Vine is vulnerable to the Narayanan-Shmatikov attack [26]. Moreover the techniques in X-Vine are specific to DHT routing, and cannot be used to design a general purpose defense mechanism for social network based applications, which is the focus of this work.

3 Basic Theory

Before we introduce our mechanism, we present some necessary notation and background on graph theory.

Let us denote the social graph as $G = (V, E)$, comprising the set of vertices V (wlog assume the vertices have labels $1, \ldots, n$), and the set of edges E, where $|V| = n$ and $|E| = m$. The focus of this paper is on undirected graphs, where the edges are symmetric. Let A_G denote the $n \times n$ *adjacency matrix* corresponding to the graph G, namely if $(i, j) \in E$, then $A_{ij} = 1$, otherwise $A_{ij} = 0$.

A *random walk* on a graph G starting at a vertex v is a sequence of vertices comprising a random neighbor v_1 of v, then a random neighbor v_2 of v_1 and so on. A random walk on a graph can be viewed as a Markov chain. We denote the *transition probability matrix* of the random walk/Markov chain as P, given by:

$$P_{ij} = \begin{cases} \frac{1}{\mathsf{deg}(i)} & \text{if } (i,j) \text{ is an edge in } G\,, \\ 0 & \text{otherwise}\,. \end{cases} \quad (1)$$

where $\mathsf{deg}(i)$ denotes the degree of the vertex i. At any given iteration t of the random walk, let us denote with $\pi(t)$ the probability distribution of the random walk state at that iteration ($\pi(t)$ is a vector of n entries). The state distribution after t iterations is given by $\pi(t) = \pi(0) \cdot P^t$, where $\pi(0)$ is the initial state distribution. The probability of a t-hop random walk starting from i and ending at j is given by $(P^t)_{ij}$. From now on, to simplify the notation, we write $(P^t)_{ij}$ as P^t_{ij} in order to denote the element (i, j) of the matrix P^t.

For irreducible and aperiodic graphs (which undirected and connected *social graphs* are), the corresponding Markov chain is ergodic, and the state distribution of the random walk $\pi(t)$ converges to a unique stationary distribution denoted by π. The stationary distribution π satisfies $\pi = \pi \cdot P$.

For undirected and connected social graphs, we can see that the probability distribution $\pi_i = \frac{\mathsf{deg}(i)}{2 \cdot m}$ satisfies the equation $\pi = \pi \cdot P$, and is thus the unique stationary distribution of the random walk.

Let us denote the eigenvalues of A as $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$, and the eigenvalues of P as $\nu_1 \geq \nu_2 \geq \ldots \geq \nu_n$. The eigenvalues of both A and P are real. We denote the second largest eigenvalue modulus (SLEM) of the transition matrix P as $\mu = \max(|\nu_2|, |\nu_n|)$. The eigenvalues of matrices A and P are closely related to structural properties of graphs, and are considered utility metrics in the literature.

4 Structured Perturbation

4.1 System model and goals

For the deployment of secure applications that leverage user's trust relationships, we envision a scenario where these applications bootstrap user's trust relationships using existing online social networks (OSNs) such as Facebook or Google+.

However, most applications that leverage this information do not make any attempt to hide it; thus an adversary can exploit protocol messages to learn the entire social graph.

Our vision is that OSNs can support these applications while protecting the link privacy of users by introducing noise in the social graph. Of course the addition of noise must be done in a manner that still preserves application utility. Moreover the mechanism for introducing noise should be computationally efficient, and must not present undue burden to the OSN operator.

We need a mechanism that takes the social graph G as an input, and produces a transformed graph $G' = (V, E')$, such that the vertices in G' remain the same as the original input graph G, but the set of edges is perturbed to protect link privacy. The constraint on the mechanism is that application utility of systems that leverage the perturbed graph should be preserved. Conventional metrics of utility include degree sequence and graph eigenvalues; we will shortly define a general metric for utility of perturbed graphs in the following section. There is a tradeoff between privacy of links in the social graph and the utility derived out of perturbed graphs. As more and more noise is added to the social graph, the link privacy increases, but the corresponding utility decreases.

4.2 Perturbation algorithm

We propose the paradigm of *structured graph perturbation*, in which we introduce noise in the social graph (by deleting real edges and introducing fake edges) such that the local structures in the original social graph are preserved.

Let t be the parameter that governs how much noise we wish to inject in the social graph. We propose that for each node u in graph G, we perturb *all* of u's contacts as follows. Suppose that node v is a social contact of node u. Then we perform a random walk of length $t-1$ starting from node v. Let node z denote the terminus point of the random walk. Our main idea is that instead of the edge (u, v), we will introduce the edge (u, z) in the graph G'. It is possible that the random walk terminates at either node u itself, or that node z is already a social contact of u in the transformed graph G' (due to a previously added edge). To avoid self loops and duplicate edges, we perform another random walk from vertex v until a suitable terminus vertex is found, or we reach a threshold number of tries, denoted by parameter M.

For undirected graphs, the algorithm described so far would double the number of edges in the perturbed graphs: for each edge (u, v) in the original graph, an edge would be added between a vertex u and the terminus point of the random walk from vertex v, as well as between vertex v and the terminus point of the random walk from vertex u. To preserve the degree distribution, we could add an edge between vertex u and vertex z in the transformed graph with probability 0.5. However this could lead to low degree nodes becoming disconnected from the social graph with non-trivial probability. To account for this case, we add the first edge corresponding to the vertex u with probability 1, while the remaining edges are accepted with a reduced probability to preserve the degree distribution. The overall algorithm is depicted in Algorithm 1. The computational complexity of our algorithm is $O(m)$.

Algorithm 1 $\mathsf{Transform}(G, t, M)$: Perturb undirected graph G using perturbation t and maximum loop count M.

```
G' = null;
foreach vertex u in G
  let count = 1;
  foreach neighbor v of vertex u
   let loop = 1;
   do
    perform t − 1 hop random walk from vertex v;
    let z denote the terminal vertex of the random
walk;
    loop + +;
   until (u = z ∨ (u, z) ∈ G') ∧ (loop ≤ M)
   if loop ≤ M
    if count = 1
     add edge (u, z) in G'
    else
     let deg(u) denote degree of u in G;
       add edge (u, z) in G' with probability
(0.5×deg(u)−1)/(deg(u)−1);
  count + +;
return G';
```

4.3 Visual depiction of algorithm

For our evaluation, we consider two real world social network topologies (a) *Facebook friendship graph from the New Orleans regional network [33]*: the dataset comprises 63,392 users that have 816,886 edges amongst them, and (b) *Facebook interaction graph from the New Orleans regional network [33]*: the dataset comprises 43,953 users that have 182,384 edges amongst them. Mohaisen et al. [24] found that preprocessing social graphs to exclude low degree nodes significantly changes the graph theoretic characteris-

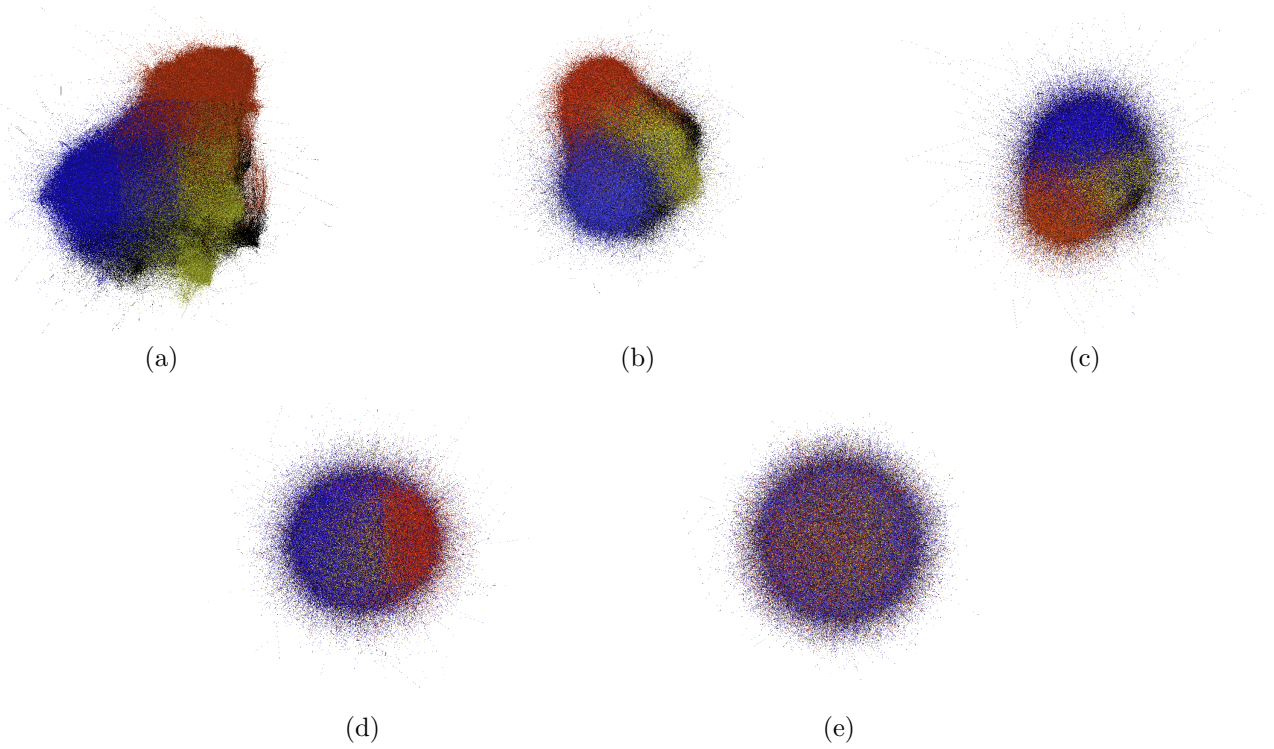


Figure 1. Facebook dataset link topology (a) Original graph (b) Perturbed, $t = 5$ (c) $t = 10$ (d) $t = 15$, and (e) $t = 20$. The color coding in (a) is derived using a modularity based community detection algorithm. For the remaining figures, the color coding of vertices is same as in (a). We can see that short random walks preserve the community structure of the social graph, while introducing a significant amount of noise.

tics. Therefore, we did not pre-process the datasets at all.

Figure 1 depicts the original Facebook friendship graph, and the perturbed graphs generated by our algorithm for varying perturbation parameters, using a force directed algorithm for depicting the graph [3]. The color coding of nodes in the figure was obtained by running a modularity based community detection algorithm [3] on the *original* Facebook friendship graph, which yielded three communities. For the perturbed graphs, we used the same color for the vertices as in the original graph. This representation allows us to visually see the perturbation in the community structure of the social graph. We can see that for small values of the perturbation parameter, the community structure (related to utility) is strongly preserved, even though the edges between vertices are randomized. As the perturbation parameter is increased, the graph looses its community structure, and eventually begins to resemble a random graph.

Figure 2 depicts a similar visualization for the Facebook interaction graph. In this setting, we found two communities using a modularity based community detection algorithm in the original graph. We can see a similar trend in the Facebook interaction graph as well: for small values of perturbation algorithm, the community structure is somewhat preserved, even though significant randomization has been introduced in the links. In the following sections, we formally quantify the utility and privacy properties of our perturbation mechanism.

5 Utility

In this section, we develop formal metrics to characterize the utility of perturbed graphs, and then analyze the utility of our perturbation algorithm.

5.1 Metrics

One approach to measure utility would be to consider global graph theoretic metrics, such as the second largest eigenvalue modulus of the graph transition matrix P. However, from a user perspective, it may be the

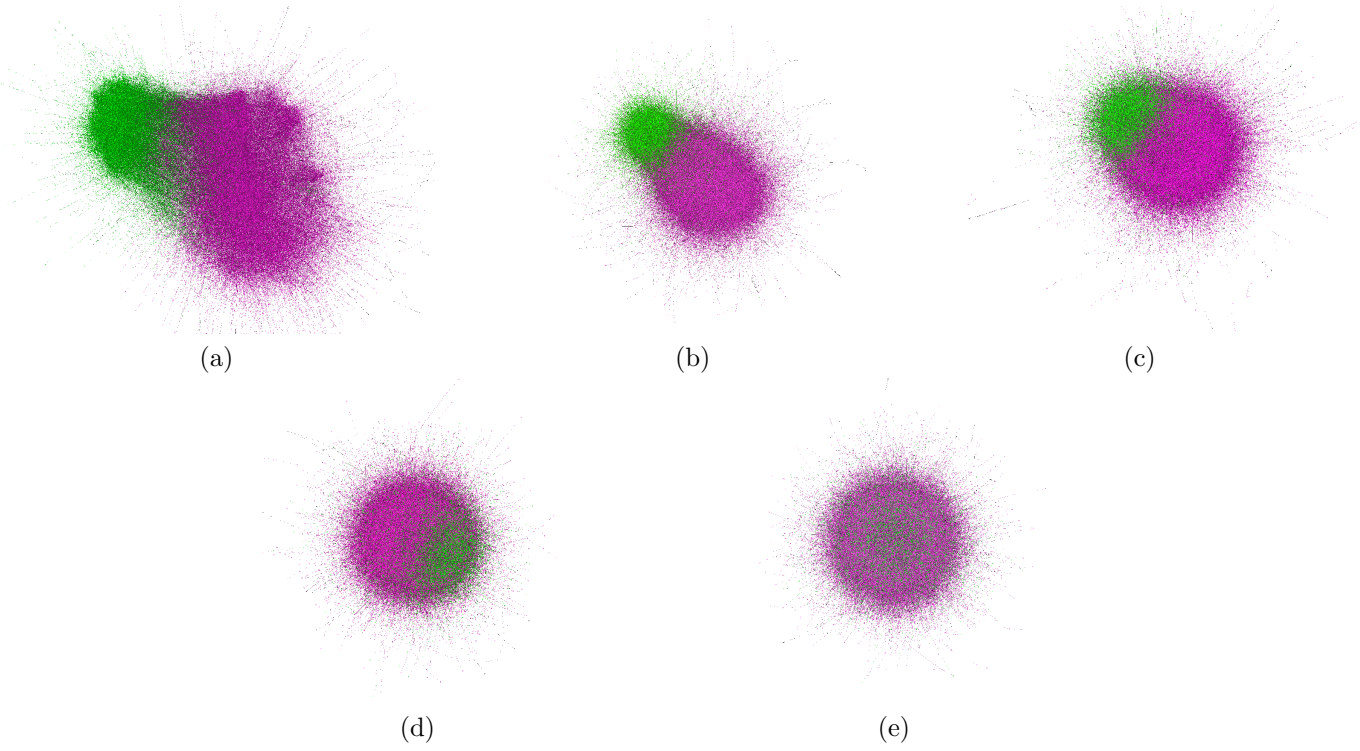


Figure 2. Facebook dataset interaction topology (a) Original graph (b) Perturbed $t = 5$ (c) $t = 10$ (d) $t = 15$, and (e) $t = 20$. We can see that short random walks preserve the community structure of the social graph, while introducing a significant amount of noise.

case that the users' position in the perturbed graph relative to malicious users is much worse, even though the global graph properties remain the same. This motivates our first definition of utility of a perturbed graph from the perspective of a single user.

Definition 1. *The vertex utility of a perturbed graph G' for a vertex v, with respect to the original graph G and an application parameter l, is defined as the statistical distance between the probability distributions induced by l hop random walks starting from v in graphs G and G', i.e.,*

$$\mathsf{VU}(v, G, G', l) = \mathsf{distance}(P_v^l(G), P_v^l(G'))\,,$$

where P_v^l denotes the v-th row of the matrix P^l.

In the above definition, the parameter l is linked to higher level applications that leverage social graphs. For example, Sybil defense mechanisms exploit large scale community structure of social networks, where the application parameter $l \geq 10$. For applications such as recommendation systems, it may be more important to preserve the local community characteristics, where l could be set to a smaller value.

Random walks are intimately linked to the structure of communities and graphs, so it is natural to consider their use when defining utility of perturbed graphs. In fact, a lot of security applications directly exploit the trust relationships in social graphs by performing random walks themselves, such as Sybil defenses and anonymous communication.

There are several ways to define statistical distance between probability distributions P and Q [4] (assume P and Q are vectors of n entries summing up to one). In this work, we consider the following three notions.

The *total variation* distance between two probability distributions is a measure of the maximum difference between the probability distributions for any individual element. Namely

$$\mathsf{distance}_v(P, Q) = ||P - Q||_{tvd} = \sup_i |p_i - q_i|\,.$$

As we will discuss shortly, the total variation distance is closely related to the computation of several graph properties such as mixing time and second largest eigenvalue modulus. However, the total variation distance only considers the maximum difference between probability distributions corresponding to an element,

and not the differences in probabilities corresponding to other elements of the distribution. This motivates the use of *Hellinger* distance, defined as

$$\mathsf{distance}_h(P,Q) = \frac{1}{\sqrt{2}} \cdot \sqrt{\sum_{i=1}^{n} (\sqrt{p_i} - \sqrt{q_i})^2} \,.$$

The Hellinger distance is related to the Euclidean distance between the square root vectors of P and Q. Finally, we also consider the *Jenson-Shannon* distance measure, which takes an information theoretic approach of averaging the Kullback-Leibler divergence between P and Q, and between Q and P (since Kullback-Leibler divergence by itself is not symmetric) and is defined as

$$\mathsf{distance}_j(P,Q) = \frac{1}{2} \cdot \sum_{i=1}^{n} p_i \log\left(\frac{p_i}{q_i}\right) + \frac{1}{2} \cdot \sum_{i=1}^{n} q_i \log\left(\frac{q_i}{p_i}\right) \,.$$

Using these notions, we can compute the utility of the perturbed graph with respect to an individual vertex (vertex utility). Note that a lower value of $\mathsf{VU}(v, G, G', l)$ corresponds to higher utility (we want distance between probability distributions over original graph and perturbed graph to be low). Using the concept of vertex utility, we can define metrics for overall utility of a perturbed graph.

Definition 2. *The overall mean vertex utility of a perturbed graph G' with respect to the original graph G, and an application parameter l is defined as the mean utility for all vertices in G, i.e.,*

$$\mathsf{VU}_{mean}(G, G', l) = \sum_{v \in V} \frac{\mathsf{distance}(P_v^l(G), P_v^l(G'))}{|V|} \,.$$

Similarly the maximum vertex utility (worst case) of a perturbed graph G' is defined by computing the maximum of the utility values over all vertices in G, i.e.,

$$\mathsf{VU}_{max}(G, G', l) = \max_{v \in V} \{\mathsf{distance}(P_v^l(G), P_v^l(G'))\} \,.$$

The notion of maximum vertex utility is particularly interesting, specially in conjunction with the use of total variation distance. This is because of its relationship to global graph metrics such as mixing times and second largest eigenvalue modulus, which we demonstrate next. Our analysis shows the generality of our formal definition for utility.

5.2 Metrics analysis

Towards this end, we first introduce the notion of mixing time of a Markov process. The mixing time of a Markov process is a measure of the minimum number of steps needed to converge to its unique stationary distribution. Formally, the mixing time of a graph $G = (V, E)$ is defined as

$$\tau_G(\epsilon) = \max_{v \in V} \{\min_{t>0} \{t : |P_v^t(G) - \pi| < \epsilon\}\} \,,$$

where π is the stationary distribution defined in Section 3. The following two theorems illustrate the bound on global properties of the perturbed graph, using the global properties of the original graph, and the utility metric. We defer the proofs of these theorems to the Appendix.

Theorem 1. *Let $\mathsf{VU}_{max}(G, G', l)$ be the maximum vertex utility distance between the perturbed graph G' and the original graph G. Then $\tau_{G'}(\epsilon + \mathsf{VU}_{max}(G, G', \tau_G(\epsilon)) \leq \tau_G(\epsilon)$.*

Theorem 1 relates the mixing time of the perturbed graph using the mixing time of the original graph, and the max vertex utility metric, for application parameter $l = \tau_G(\epsilon)$.

Theorem 2. *Let μ_G be the second largest eigenvalue modulus (SLEM) of transition matrix P_G of graph G. We can bound the SLEM $\mu_{G'}$ of a perturbed graph G' using the mixing time $\tau_G(\epsilon)$ of the original graph and the worst case vertex utility distance $\chi = \mathsf{VU}_{max}(G, G', \tau_G(\epsilon))$ between G and G'. Namely we have*

$$\mu_{G'} \leq \frac{2\tau_G(\epsilon)}{2\tau_G(\epsilon) + \log\left(\frac{1}{2\epsilon + 2\chi}\right)} \,.$$

Theorem 2 relates the second largest eigenvalue modulus of the perturbed graph, using the mixing time of the original graph, and the worst case vertex utility metric for application parameter $l = \tau_G(\epsilon)$.

These theorems show the generality of our utility definitions. Mechanisms that provide good utility (have low values of VU_{max}), introduce only a small change in the mixing time and SLEM of perturbed graphs.

5.3 Algorithm analysis

Our above results show the general relationship between our utility metrics and global graph properties (which hold for any perturbation algorithm). Next, we analyze the properties of our proposed perturbation algorithm.

First, we empirically compute the mean vertex utility of the perturbed graphs (VU_{mean}), for varying perturbation parameters and varying application parameters. Figure 3 depicts the mean vertex utility for the

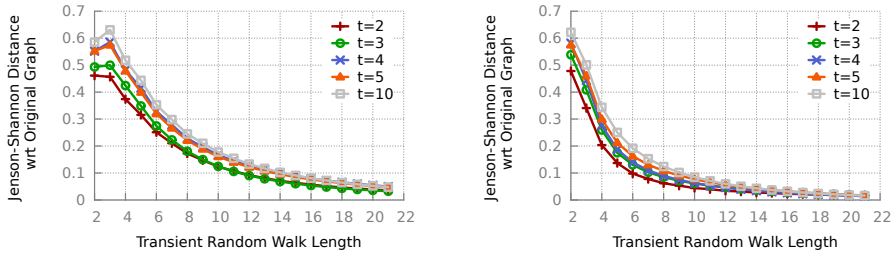


Figure 3. *Jenson-Shannon distance between transient probability distributions for original graph and transformed graph* **using (a) Facebook interaction graph (b) Facebook link graph. We can see that as the original graph is perturbed to a larger degree, the distance between original and transformed transient distributions increases, decreasing application utility.**

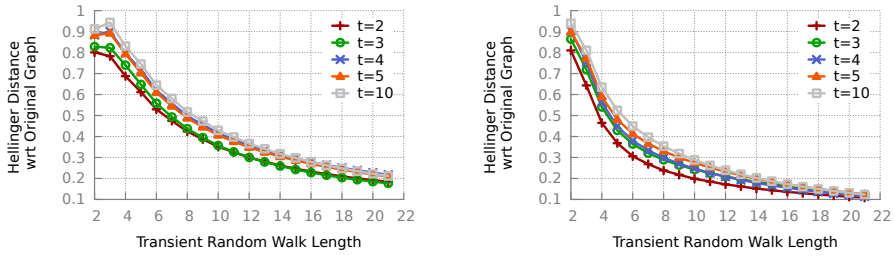


Figure 4. *Hellinger distance between transient probability distributions for original graph and transformed graph* **using (a) Facebook interaction graph (b) Facebook link graph. We can see that even with different notions of distance between probability distributions (Hellinger/Jenson-Shannon), the distance between original graph and perturbed graph monotonically increases depending on the perturbation degree.**

Facebook interaction and friendship graphs using the Jenson-Shannon information theoretic distance metric. We can see that as the perturbation parameter increases, the distance metric increases. This is not surprising, since additional noise will increase the distance between probability distributions computed from original and perturbed graphs. We can also see that as the application parameter l increases, the distance metric decreases. This illustrates that our perturbation algorithm is ideally suited for security applications that rely on local or global community structures, as opposed to applications that require *exact* information about one or two hop neighborhoods. We can see a similar trend when using Hellinger distance to compute the distance between probability distributions, as shown in Figure 4.

Theorem 3. *The expected degree of each node after the perturbation algorithm is the same as in the original graph, i.e., $\forall v \in V$, it holds that $\mathbb{E}(\mathsf{deg}'(v)) = \mathsf{deg}(v)$, where $\mathsf{deg}'(v)$ denotes the degree of vertex v in G'.*

Proof. On an expectation, half the degree of any node v is preserved via outgoing random walks from v in the perturbation process. To prove the theorem, we need to show that each node v is the terminal point of $\mathsf{deg}(v)/2$ random walks in the perturbation mechanisms (on average). From the time reversibility property of the random walks, we have that $P^t_{ij}\pi_i = P^t_{ji}\pi_j$. Thus for any node i, the incoming probability of a random walk starting from node j is $P^t_{ji} = P^t_{ij}\frac{\mathsf{deg}(i)}{\mathsf{deg}(j)}$, i.e., it is proportional to the node degree of i. Thus the expected number of random walks terminating at node i in the perturbation algorithm is given by $\sum_{v\in V}\mathsf{deg}(v)P^{t-1}_{vi}$. This is equivalent to $\sum_{v\in V}P^{t-1}_{iv}\mathsf{deg}(i) = \mathsf{deg}(i)$. Since half of these walks will be added to the graph G' on average, we have that $\mathbb{E}(\mathsf{deg}'(v)) = \mathsf{deg}(v)$. □

Corollary 1. *The expected value of the largest eigenvalue of the transformed graph is bounded as $\max\{d_{avg}, \sqrt{d_{max}}\} \leq \mathbb{E}(\lambda'_1) \leq d_{max}$, where d_{avg} and d_{max} are the average and maximum degrees respectively.*

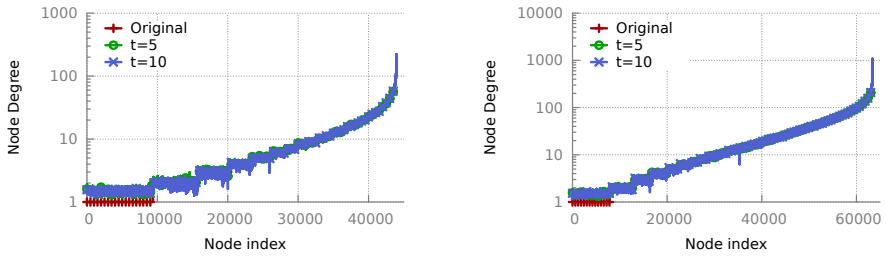


Figure 5. *Degree distribution of nodes* **using (a) Facebook interaction graph (b) Facebook link graph. We can see that the expected degree of each node after the perturbation process remains the same as in the original graph.**

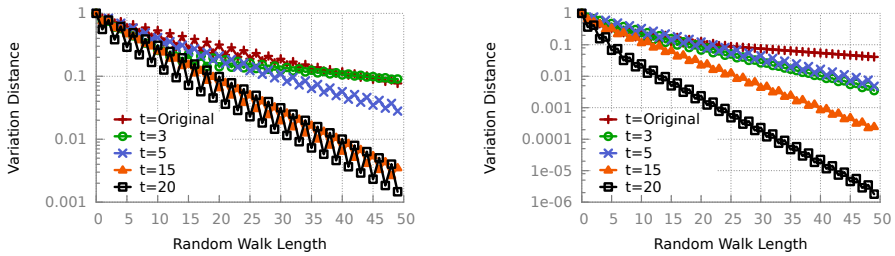


Figure 6. *Total variation distance as a function of random walk length* **using (a) Facebook interaction graph (b) Facebook link graph. We can see that increasing the perturbation parameter of our algorithm reduces the mixing time of the graph.**

From the Perron-Frobenius theorem, we have that the largest eigenvalue of the graph is related to the notion of average graph degree as follows,

$$\max\{d'_{avg}, \sqrt{d'_{max}}\} \leq \lambda'_1 \leq d'_{max}\,.$$

Taking expectation on the above equation, and using the previous theorem yields the corollary.

Next, we show experimental results validating our theorem. Figure 5 depicts the node degrees of the original graphs, and expected node degrees of the perturbed graphs, corresponding to all nodes in the Facebook interaction and friendship graphs. In this figure, the points in a vertical line for different perturbed graphs correspond to the same node index. We can see that the degree distributions are nearly identical, validating our theoretical results.

Theorem 4. *Using our perturbation algorithm, the mixing time of the perturbed graphs is related to the mixing time of the original graph as follows:* $\frac{\tau_G(\epsilon)}{t} \leq \mathbb{E}(\tau_{G'}(\epsilon)) \leq \tau_G(\epsilon)$.

Theorem 4 bounds the mixing time of the perturbed graph using the mixing time of the original graph and the perturbation parameter t. We defer the proof to the Appendix. Finally, we compute the mixing time of the original and perturbed graphs using simulations. Figure 6 depicts the total variation distance between random walks of length x and the stationary distribution, for the original and perturbed graphs.[1] We can see that as the perturbation parameter increases, the total variation distance (and the mixing time) decreases. Moreover, for small values of the perturbation parameter, the difference from the original topology is small. As an aside, it is interesting to note that the variation distance for the Facebook friendship graph is orders of magnitude smaller that the Facebook interaction graph. This is because the Facebook interaction graphs are a lot sparser, resulting in slow mixing.

6 Privacy

We now address the question of understanding link privacy of our perturbation algorithm. We use sev-

[1] Variation distance has a slight oscillating behavior at odd and even steps of the random walk; this phenomenon is also observed in Figure 8.

eral notions for quantifying link privacy, which fall into two categories (a) quantifying exact probabilities of de-anonymizing a link given specific adversarial priors, and (b) quantifying risk of de-anonymizing a link without making specific assumptions about adversarial priors. We also characterize the relationship between utility and privacy of a perturbed graph.

6.1 Bayesian formulation for link privacy

Definition 3 (Link privacy)**.** *Let G be the original graph and L be a link (edge) belonging to G. Let G' be the perturbed graph, as computed by our algorithm. The adversary is given G' and some prior (auxiliary) information H. The privacy of link L is the probability $\Pr[L \in G|G', H]$. Namely, we define the privacy of a link L (or a subgraph) as the probability of existence of the link (or a subgraph) in the original graph G under the assumption that the adversary has access to the perturbed graph G' and prior information H. We write this probability as $\Pr[L|G', H]$.*

Note that low values of link probability $\Pr[L|G', H]$ correspond to high privacy. We cast the problem of computing the link probability as a Bayesian inference problem. Using Bayes theorem, we have that:

$$\Pr[L|G', H] = \frac{\Pr[G'|L, H] \cdot \Pr[L|H]}{\Pr[G'|H]}.$$

In the above expression, $\Pr[L|H]$ is the prior probability of the link. In Bayesian inference, $\Pr[G'|H]$ is a normalization constant that is typically difficult to compute, but this is not an impediment for the analysis since sampling techniques can be used (as long as the numerator of the Bayesian formulation is computable upto a constant factor [18, 10]). Our key theoretical challenge is to compute $\Pr[G'|L, H]$.

To compute $\Pr[G'|L, H]$, the adversary has to consider all possible graphs G_p, which have the link L, and are consistent with background information H. Thus, we have that:

$$\Pr[G'|L, H] = \sum_{G_p} \Pr[G'|G_p] \cdot \Pr[G_p|L, H]. \qquad (2)$$

The adversary can compute $\Pr[G'|G_p]$ using the knowledge of the perturbation algorithm; we assume that the adversary knows the full details of our perturbation algorithm, including the perturbation parameter t. Observe that given G_p, edges in G' can be modeled as samples from the probability distribution of t hop random walks from vertices in G_p. Thus we can

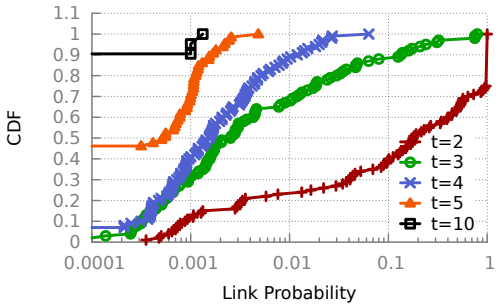


Figure 7. *Cumulative distribution of link probability* $\Pr[L|G', H]$ **(x-axis is logscale) under worst case prior $H = G - L$ using a synthetic scale free topology. Small probabilities offer higher privacy protection.**

compute $\Pr[G'|G_p]$ by using the t hop transition probabilities of vertices in G_p as:

$$\left(\frac{1}{2}\right)^{2m} \cdot \binom{2m}{m'} \cdot \prod_{(i,j)\in E(G')} \frac{P^t_{ij}(G_p) + P^t_{ji}(G_p)}{2}.$$

In general, the number of possible graphs G_p that have L as a link and are consistent with the adversary's background information can be very large, and the computation in Equation 2 then becomes intractable.

For evaluation, we consider a special case of this definition: the adversary's prior is the entire original graph without the link L (which is the link for which we want to quantify privacy). Observe that this is a very powerful adversarial prior; we use this prior to shed light on the *worst-case* link privacy using our perturbation algorithm. Under this prior, we have that:

$$\begin{aligned}\Pr[L|G', G-L] &= \frac{\Pr[G'|L, G-L] \cdot \Pr[L|G-L]}{\Pr[G'|G-L]} \\ &= \frac{\Pr[G'|G] \cdot \Pr[L|G-L]}{\Pr[G'|G-L]} \\ &= \frac{\Pr[G'|G] \cdot \Pr[L|G-L]}{\sum_l \Pr[G'|G-L+l]}. \end{aligned} \qquad (3)$$

Using $G - L$ as the adversarial prior constraints the set of possible G_p to a polynomial number. However, even in this setting, we found that the above computation is computationally expensive ($> O(n^3)$) using our real world social networks. Thus to get an understanding of link privacy in this setting,we generated a 500 node synthetic scale-free topology using the preferential attachment methodology of Nagaraja [25]. The parameters of the scale free topology was set using the average degree in the Facebook interaction

graph. Figure 7 depicts the cumulative distribution for link probability (probability of de-anonymizing a link, $\Pr[L|G', H]$) in this setting (worst case prior) for the synthetic scale-free topology. We can see that there is significant variance in the privacy protection received by links in the topology: for example, using perturbation parameter $t = 2$, 40% of the links have a link probability less than 0.1 (small is better), while 30% of the links have have a probability of 1 (can be fully de-anonymized). Note that this is a worst case analysis, since we assume that an attacker knows the entire original graph except the link in question. Furthermore, even in this setting, as the perturbation parameter t increases, the privacy protection received by links substantially improves: for example, using $t = 5$, all of the links have a link probability less than 0.01, while 70% of the links have a link probability of less than 0.001. Thus we can see that even in this worst case analysis, our perturbation mechanism offers good privacy protection.

We now compare with the work of Hay et al [12], which proposes a perturbation approach where k real edges are deleted and k fake edges are introduced at random. Even considering $k = m/2$ (which would substantially hurt utility), 50% of the edges in the perturbed graph are *real* edges between users; for these edges, $\Pr[L|G'] = 0.5$. Thus we can see that our perturbation mechanism significantly improves privacy compared with the work of Hay et al.

6.2 Relationship between privacy and utility

Intuitively, there is a relationship between link privacy and the utility of the perturbed graphs. Next, we formally quantify this relationship.

Theorem 5. *Let the maximum vertex utility of the graph (over all vertices) corresponding to an application parameter l be $\mathsf{VU}_{max}(G, G', l)$. Then for any two vertices A and B, we have that $\Pr[L_{AB}|G'] \geq f(\delta)$, where $f(\delta)$ denotes the prior probability of two vertices being friends given that they are both contained in a δ hop neighborhood, and δ is computed as $\delta = \min\{k : P^k_{AB}(G') - \mathsf{VU}_{max}(G, G', k) > 0\}$.*

Utility measures the change in graph structure between original and perturbed graphs. If this change is small (high utility), then an adversary can infer information about the original graph given the perturbed graph. For a given level of utility, the above theorem demonstrates a lower bound on link privacy.

We defer the proof of the above theorem to the Appendix. The above theorem is a general theorem that holds for all perturbed graphs. To shed some intuition, we specifically analyze the lower bounds on privacy for our perturbation algorithm (where parameter t governs utility), using the transition probability between A and B in the perturbed graph G' as a feature to assign probabilities to nodes A and B of being friends in the original graph G. We are interested in the quantity $\Pr[L_{AB} \mid P^k_{AB}(G') > x]$, for different values of k. Using Bayes' theorem, we have that the probability $\Pr[L_{AB} \mid P^k_{AB}(G') > x]$ can be written as

$$\frac{\Pr[P^k_{AB}(G') > x \mid L_{AB}] \cdot \Pr[L_{AB}]}{\Pr[P^k_{AB}(G') > x]}. \qquad (4)$$

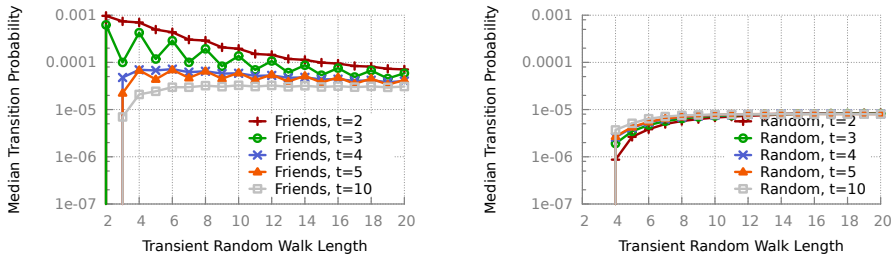
Moreover, we have that

$$\begin{aligned} \Pr[P^k_{AB}(G') > x] &= \Pr[P^k_{AB}(G') > x \mid L_{AB}] \cdot \Pr[L_{AB}] \\ &+ \Pr[P^k_{AB}(G') > x \mid \overline{L}_{AB}] \cdot \Pr[\overline{L}_{AB}], \end{aligned}$$

where $\overline{L}_{AB}$ denotes the event when vertices A and B don't have a link.

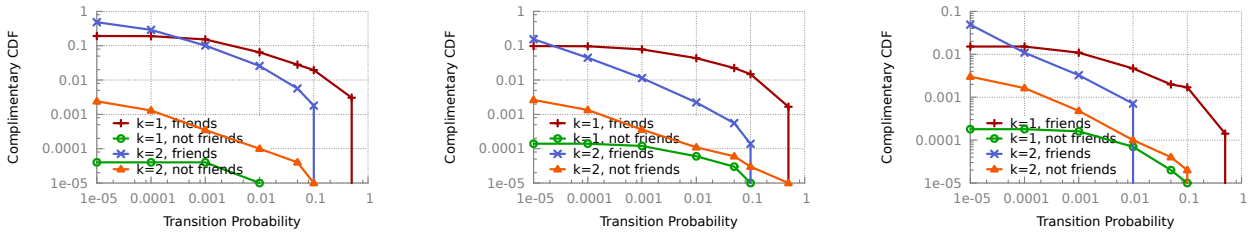
To get some insight, we computed the probability distributions $\Pr[P^k_{AB}(G')|L_{AB}]$ and $\Pr[P^k_{AB}(G')|\overline{L}_{AB}]$ using simulations on our real world social network topologies. Figure 8 depicts the median value of the respective probability distributions, as a function of parameter k, for different perturbation parameters t, using the Facebook interaction graph. We can see that the median transition probabilities are higher for the scenario where two users are originally friends (as opposed to the setting where they are not friends). We can also see that the difference between median transition values in the two scenarios is higher when (a) the perturbation parameter t is small, and (b) the parameter K (random walk length) is small. This difference is related to the privacy of the link—the larger the difference, the greater the loss in privacy. The insight from this figure is that for small perturbation parameters, the closer two nodes are to each other in the perturbed graph, the higher their chances of being friends in the original graph.

Next, we consider the full distribution of transition probabilities (as opposed to only the median values discussed above). Figure 9(a) depicts the complimentary cumulative distribution ($P^k_{AB}(G') > x$) using perturbation parameter $t = 2$ for the Facebook interaction graph. Again, we can see that when A and B are friends, they have higher transition probability to each other in the perturbed graph, compared to the scenario when A and B are not friends. Moreover, as the value of k increases, the gap between the distributions becomes smaller. Similarly, as the value of t increases, the gap between the distributions becomes smaller, as depicted in Figures 9(b-c). We can use these simulation results to compute the probabilities in Equation 4.



(a) (b)

Figure 8. *Median transition probability between two vertices in transformed graph* **when (a) two vertices were neighbors (friends) in the original graph and (b) two vertices were not neighbors in the original graph, for the Facebook interaction graph.**



(a) (b) (b)

Figure 9. *Complimentary cumulative distribution for* $P^k_{AB}(G')$ **using (a) perturbation parameter** $t = 2$**, and (b) perturbation parameter** $t = 5$**, and (c) perturbation parameter** $t = 10$ **for the Facebook interaction graph.**

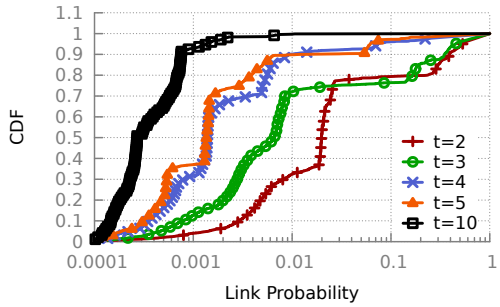


Figure 10. *Cumulative distribution of link probability* $\Pr[L|G', H]$ **for the Facebook interaction graph. Smaller probabilities offer higher privacy protection.**

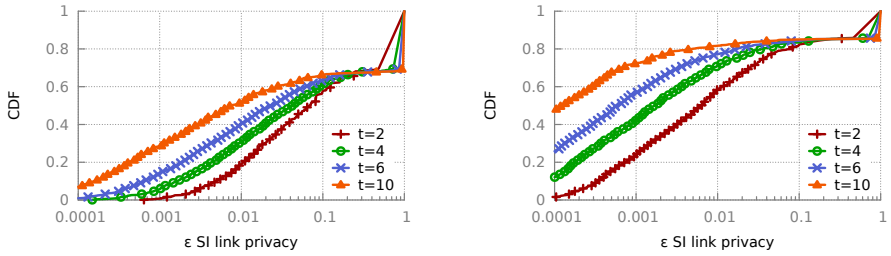
One way to analyzing the lower bound on link privacy would be to choose a uniform prior for vertices being friends in the original graph, i.e., $\Pr[L_{AB}] = \frac{m}{\binom{n}{2}}$. Correspondingly, $\Pr[\overline{L}_{AB}] = 1 - \Pr[L_{AB}]$.

Figure 10 depicts the cumulative distribution of link probability computed using the above methodology, for the Facebook interaction graph. We can see the variance in privacy protection received by links in the topology. Using $t = 2$, 80% of the links have a link probability of less than 0.1. Increasing perturbation parameter t significantly improves performance; using $t = 3$, 95% of the links have a link probability less than 0.1, and 90% of the links have a link probability less than 0.01. In summary, our analysis shows that a given level of utility translates into a lowerbound on privacy offered by the perturbation mechanism.

6.3 Risk based formulation for link privacy

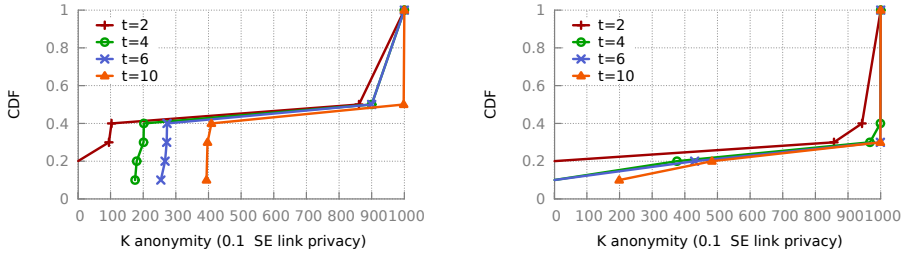
The dependence of the Bayesian inference based privacy definitions on the prior of the adversary motivates the formulation of new metrics that are not specific to adversarial priors. We first illustrate a preliminary definition of link privacy (which is unable to account for links to degree 1 vertices), and then subsequently improve it.

Definition 4 (ϵ-SI link privacy)**.** *We define the structural impact (SI) of a link L in graph G with respect to a perturbation mechanism $\mathcal{M}$, as the statistical distance between probability distributions of the output $\mathcal{M}(G)$ of the perturbation mechanism (i.e., the set of possible perturbed graphs) when using (a) the original graph G as an input to the perturbation mechanism, and (b) the graph $G-L$ as an input to the perturbation mechanism.*



(a) (b)

Figure 11. *Cumulative distribution of ϵ-SI link privacy* **for (a) Facebook interaction graph (b) Facebook link graph. Note that the SI privacy definition is not applicable to links for which either of the vertex has degree of 1, since removing that link disconnects the graph.**



(a) (b)

Figure 12. *Cumulative distribution of K-anonymity set size for $\epsilon = 0.1$ SE link privacy* **using (a) Facebook interaction graph (b) Facebook link graph.**

Let $\Pr[G' = \mathcal{M}(G)]$ *denote the probability distribution of perturbed graphs* G' *using the perturbation mechanism* $\mathcal{M}$ *and input graph* G*. A link has* ϵ*-SI privacy if the statistical distance*

$$||\Pr[G' = \mathcal{M}(G)] - \Pr[G' = \mathcal{M}(G - L)]|| < \epsilon\,.$$

Intuitively, if the SI privacy of a link is high, then the perturbation process leaks more information about that link. On the other hand, if the SI privacy of a link is low, then the perturbed graph G' leaks less information about the link.

As before, we consider the total variation distance as our distance metric between probability distributions. Observe that the links in graphs G' are samples from the probability distribution $P_v^t(G)$, for $v \in V$. So we can bound the difference in probability distributions of perturbed graphs generated from G and $G - L$, by the worst case total variation distance between probability distributions $P_v^t(G)$ and $P_v^t(G-L)$, over all $v \in V$, i.e., by $\mathsf{VU}_{max}(G, G - L, t)$.

Note that our preliminary attempt at defining link privacy above does not accommodate links where either of its endpoints have degree 1, since removal of that link disconnects the graph. This is illustrated in Figure 11, which depicts the cumulative distribution of ϵ-SI link privacy values. We can see that approximately 30% and 20% of links in the Facebook interaction graph and friendship graphs respectively do not receive any privacy protection under this definition (because they are connected to degree 1 vertices). For the remaining links, we can see a similar qualitative trend as before: increasing the perturbation parameter t significantly improves link privacy. To overcome the limitations of this definition, we propose an alternate formulation based on the notion of link equivalence.

Definition 5 (K-anonymous ϵ-SE link privacy)**.** *We define the structural equivalence (SE) between a link* L' *with a link* L *in graph* G *with respect to a perturbation mechanism* $\mathcal{M}$*, as the statistical distance between probability distributions of the output of the perturbation mechanism, when using (a) the original graph* G *as an input to the perturbation mechanism, and (b) the graph* $G - L + L'$ *as the input to the perturbation mechanism. A link* L *has* K*-anonymous* ϵ*-SE privacy, if there exist at least* K *links* L'*, such that*

$$||\Pr[G' = \mathcal{M}(G)] - \Pr[G' = \mathcal{M}(G - L + L')]|| < \epsilon\,.$$

Observe that this definition of privacy is able to account for degree 1 vertices, since they can become connected to the graph via the addition of other alternate links L'. For our experiments, we limited the number

of alternate links explored to 1000 links (for computational tractability). Figure 12 depicts the cumulative distribution of anonymity set sizes for links using $\epsilon = 0.1$ for the Facebook interaction and friendship graphs. For $t = 2$ we see a very similar trend as in the previous definition, where a non-trivial fraction of links do not receive much privacy. Unlike the previous setting however, as we increase the perturbation parameter t, the anonymity set size for even these links improves significantly. Using $t = 10$, 50% and 70% of the links in the interaction and friendship graphs respectively, achieved the maximum tested anonymity set size of 1000 links.

6.4 Relation to differential privacy

There is an interesting relation between our risk-based privacy definitions and differential privacy [9]. A differentially-private mechanism adaptively adds noise to the system to ensure that *all* user records in a database (links in our setting) receive a threshold privacy protection. In our mechanism, we are adding a fixed amount of noise (governed by the perturbation parameter t), and observing the variance in ϵ and anonymity set size values.

7 Applications

In this section, we demonstrate the applicability of our perturbation mechanism to social network based systems.

7.1 Secure routing

Several peer-to-peer systems perform routing over social graph to improve performance and security, such as Sprout [17], Tribler [27], Whanau [14] and X-Vine [21]. Next, we analyze the impact of our perturbation algorithm on the utility of Sprout.

7.1.1 Sprout

Sprout is a routing system that enhances the security of conventional DHT routing by leveraging trusted social links when available. For example, when routing towards a DHT key, if leveraging a social link makes forward progress in the DHT namespace, then the social link is used for routing. The authors of Sprout considered a linear trust decay model, where a user's social contacts are trusted with probability f (set to 0.95 in [17]), and the trust in other users decreases as a linear function of the shortest path distance between the users (a decrement of 0.05 is used in [17]). The decrement is bounded by a value reflecting the probability of a random user in the network being trusted (set to 0.6 in [17]).

Table 1. Path reliability using Sprout for a linear trust decay model.

Facebook interaction graph		Facebook friendship graph	
Mechanism	Reliability	Mechanism	Reliability
Original	0.110	Original	0.140
$t = 3$	0.101	$t = 3$	0.126
$t = 5$	0.101	$t = 5$	0.121
$t = 10$	0.096	$t = 10$	0.118
Chord	0.075	Chord	0.072

The reliability of a DHT lookup in Sprout is defined as the probability of all users in the path being trusted. Table 1 depicts the reliability of routing using a single DHT lookup in Sprout, for the original and the perturbed topologies. We used Chord as the underlying DHT system. For each perturbation parameter, our results were averaged over 100 perturbed topologies. We can see that as the perturbation parameter increases, the utility of application decreases. For example, using the original Facebook interaction topology, the reliability of a single DHT path in sprout is 0.11, which drops to 0.10 and 0.096 when using perturbed topologies with parameters $t = 5$ and $t = 10$ respectively. However, even when using $t = 10$, the performance is better as compared with the scenario where social links are not used for routing (Chord's baseline performance of 0.075). We can see similar results for the Facebook friendship graph as well.

7.2 Sybil detection

In a Sybil attack [8], a single user or an entity emulates the behavior of multiple identities in a system. Many systems are built on the assumption that there is a bound on the fraction of malicious nodes in the systems. By being able to insert a large number of malicious identities, an attacker can compromise the security properties of such systems. Sybil attacks are a powerful threat against both centralized as well as distributed systems, such as reputation systems, consensus and leader election protocols, peer-to-peer systems, anonymity systems, and recommendation systems.

A wide body of recent work has proposed to leverage trust relationships embedded in social networks for detecting Sybil attacks [36, 35, 6, 23, 31]. However, in all of these mechanisms, an adversary can learn the trust relationships in the social network. Next, we show that our perturbation algorithm preserves the ability of above mechanisms to detect Sybils, while protecting the privacy of the social network trust relationships.

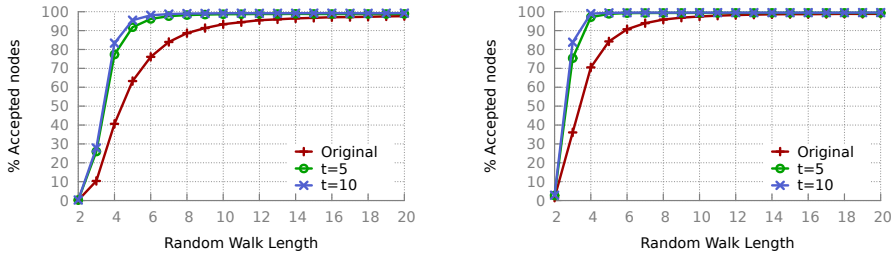


Figure 13. *SybilLimit % validated honest nodes* **as a function of SybilLimit random route length for (a) Facebook interaction graph (b) Facebook link graph. We can see that for a false positive rate of** $1-2\%$**, the required random route length for our perturbed topologies is a factor of** $2-3$ **smaller as compared with the original topology. Random route length is directly proportional to number of Sybil identities that can be inserted in the system.**

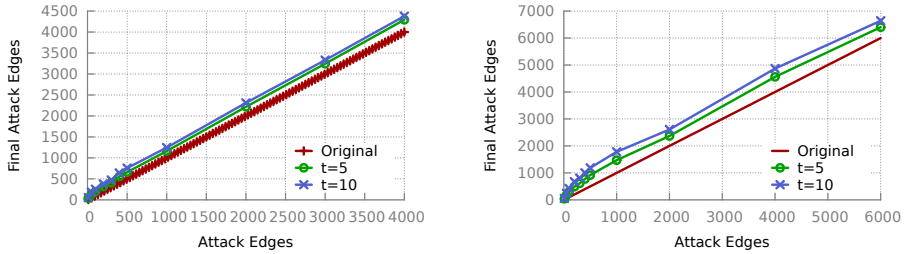


Figure 14. *Attack edges in perturbed topologies* **as a function of attack edges in the original topology. We can see that there is a marginal increase in the number of attack edges in the perturbed topologies. The attack edges are directly proportional to the number of Sybil identities that can be inserted in the system.**

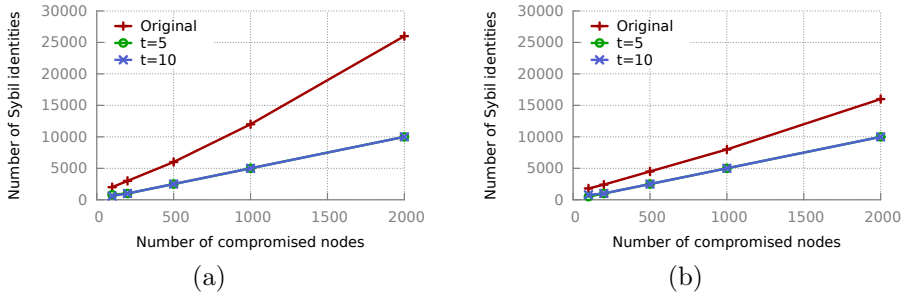


Figure 15. *Number of Sybil identities accepted by SybilInfer* **as a function of number of compromised nodes in the original topology. We can see that there is a significant decline in the number of Sybil identities using our perturbation algorithms.**

7.2.1 SybilLimit

We use SybilLimit [35] as a representative protocol for Sybil detection, since it is the most popular and well understood mechanism in the literature. SybilLimit is a probabilistic defense, and has both false positives (honest users misclassified as Sybils) and false negatives (Sybils misclassified) as honest users.

We compared the performance of running SybilLimit on original graph, and on the transformed graph, for varying perturbation parameters. For each perturbation parameter, we averaged the results over 100 perturbed topologies. Figure 13 depicts the percentage of honest users validated by SybilLimit using the original graph and perturbed graphs, as a function of the SybilLimit random route length (application parameter w). For any value of the SybilLimit random route length, the percentage of honest nodes accepted by SybilLimit is higher when using perturbed graphs. This is because our perturbation algorithms reduce the mixing time of the graph. In fact, for a false positive percentage of 1-2% (99-98% accepted nodes), the required length of the SybilLimit random routes is a factor of 2-3 smaller as compared to the original topology. SybilLimit random routes are directly proportional to the number of Sybil identities that can be inserted in the system.

This improvement in the number of accepted honest nodes (reduction in false positives) comes at the cost of increase in the number of attack edges between honest users and the attacker. Figure 14 depicts the number of attack edges in the perturbed topologies, for varying values of attack edges in the original graph. We can see that as expected, there is a marginal increase in the number of attack edges in the perturbed topologies.

Remark: The number of Sybil identities that an adversary can insert is given by $S = g' \cdot w'$. We note that the marginal increase in the number of attack edges g' is offset by the reduced length of the SybilLimit random route parameter w (for any desired false positive rate), thus achieving comparable performance with the original social graph. In fact, for perturbed topologies, since the required random route length in SybilLimit is halved for a false positive rate of 1-2%, and the increase in the number of attack edges is less than a factor of two, the Sybil defense performance has *improved* using our perturbation mechanism. Thus for Sybil defenses, our perturbation mechanism is of independent interest, even without considering the benefit of link privacy. We further validate this conclusion using another state-of-art detection mechanism called SybilInfer.

7.2.2 SybilInfer

We compared the performance of running SybilInfer on real and perturbed topologies. Figure 15 depicts the optimal number of Sybil identities that an adversary can insert before being detected by SybilInfer, as a function of real compromised and colluding users. Again, we can see that the performance of perturbed graphs is better than using original graphs. This is due to the interplay between mixing time of graphs and the number of attack edges in the Sybil defense application. Our perturbation mechanism significantly reduces the mixing time of the graphs, while suffering only a marginal increase in the number of attack edges. It is interesting to see that the advantage of using our perturbation mechanism is less in Figure 15(b), as compared to Figure 15(a). This is because the mixing time of the Facebook friendship graph is much better (as compared with the the mixing time of the Facebook interaction graph). Thus we conclude that our perturbation mechanism improves the overall Sybil detection performance of existing approaches, especially for interaction based topologies that exhibit relatively poor mixing characteristics.

8 Conclusion and Future Work

In this work, we proposed a random walk based perturbation algorithm that anonymizes *links* in a social graph while preserving the graph theoretic properties of the original graph. We provided formal definitions for utility of a perturbed graph from the perspective of vertices, related our definitions to global graph properties, and empirically analyzed the properties of our perturbation algorithm using real world social networks. Furthermore, we analyzed the privacy of our perturbation mechanism from several perspectives (a) a Bayesian viewpoint that takes into consideration specific adversarial prior, and (b) a risk based view point that is independent of the adversary's prior. We also formalized the relationship between utility and privacy of perturbed graphs. Finally, we experimentally demonstrated the applicability of our techniques on applications such as Sybil defenses and secure routing. For Sybil defenses, we found that our techniques are of independent interest.

Our work opens several directions for future research, including (a) investigating the applicability of our techniques on directed graphs (b) modeling closed form expressions for computing link privacy using the Bayesian framework (c) investigating tighter bounds on ϵ for computing link privacy in the risk-based framework, and (d) modeling temporal dynamics of social

networks in quantifying link privacy.

By protecting the privacy of trust relationships, we believe that our perturbation mechanism can act as a key enabler for real world deployment of secure systems that leverage social links.

Acknowledgments

We are very grateful to Satish Rao for helpful discussions on graph theory, including insights on utility metrics for perturbed graphs, and relating mixing times of original and perturbed graphs. We would also like to thank Ling Huang, Adrian Mettler, Nguyen Tran, and Mario Frank for helpful discussions on analyzing link privacy, as well as their feedback on early versions of this work. Our analysis of SybilLimit is based on extending a simulator written by Bimal Vishwanath. This work was supported by Intel through the ISTC for Secure Computing, by the National Science Foundation under grant numbers CCF-0424422, 0842695 and 0831501 CT-L, by the Air Force Office of Scientific Research (AFOSR) under MURI award FA9550-09-1-0539 and grant number FA9550-08-1-0352 and by the Office of Naval Research under MURI grant number N000140911081.

References

[1] D. Aldous and J. Fill. *Reversible Markov Chains and Random Walks on Graphs*. http://www.stat.berkeley.edu/ aldous/RWG/book.html.

[2] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW*, 2007.

[3] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks. In *International AAAI Conference on Weblogs and Social Media*, 2009.

[4] S.-H. Cha. Comprehensive survey of distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4), 2007.

[5] G. Danezis, C. Diaz, C. Troncoso, and B. Laurie. Drac: an architecture for anonymous low-volume communications. In *PETS*, 2010.

[6] G. Danezis and P. Mittal. Sybilinfer: Detecting Sybil nodes using social networks. In *NDSS*, 2009.

[7] R. Dey, Z. Jelveh, and K. Ross. Facebook users have become much more private: a large scale study. Technical report, New York University, 2011.

[8] J. Douceur. The Sybil Attack. In *IPTPS*, 2002.

[9] C. Dwork. Differential privacy: a survey of results. In *TAMC*, 2008.

[10] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1), 1970.

[11] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. *Proc. VLDB Endow.*, 1(1), 2008.

[12] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing social networks. Technical Report 07-09, University of Massachusetts, Amherst, 2007.

[13] A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu. Link privacy in social networks. In *CIKM*, 2008.

[14] C. Lesniewski-Laas and M. F. Kaashoek. Whanaungatanga: A Sybil-proof distributed hash table. In *NSDI*, 2010.

[15] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *SIGMOD*, 2008.

[16] X. Lu, Y. Song, and S. Bressan. Fast identity anonymization on graphs. In *DEXA*, 2012.

[17] S. Marti, P. Ganesan, and H. Garcia-Molina. SPROUT: P2P routing with social networks. In *P2P&DB*, 2004.

[18] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6), 1953.

[19] A. Mislove, A. Post, P. Druschel, and K. P. Gummadi. Ostra: leveraging trust to thwart unwanted communication. In *NSDI*, 2008.

[20] P. Mittal, N. Borisov, C. Troncoso, and A. Rial. Scalable anonymous communication with provable security. In *USENIX HotSec*, 2010.

[21] P. Mittal, M. Caesar, and N. Borisov. X-vine: Secure and pseudonymous routing using social networks. In *NDSS*, 2012.

[22] P. Mittal, M. Wright, and N. Borisov. Pisces: Anonymous communication using social networks. In *NDSS*, 2013.

[23] A. Mohaisen, N. Hopper, and Y. Kim. Keep your friends close: Incorporating trust into social network-based sybil defenses. In *INFOCOM*, 2011.

[24] A. Mohaisen, A. Yun, and Y. Kim. Measuring the mixing time of social graphs. In *IMC*, 2010.

[25] S. Nagaraja. Anonymity in the wild: mixes on unstructured networks. In *PETS*, 2007.

[26] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE SSP*, 2009.

[27] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips. Tribler: a social-based peer-to-peer system: Research articles. *Concurr. Comput. : Pract. Exper.*, 20(2), 2008.

[28] V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: output perturbation for queries with joins. In *PODS*, 2009.

[29] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao. Sharing graphs using differentially private graph models. In *IMC*, 2011.
[30] Y. Sovran, J. Li, and L. Subramanian. Unblocking the internet: Social networks foil censors. Technical report, NYU, 2008.
[31] N. Tran, J. Li, L. Subramanian, and S. Chow. Optimal sybil-resilient node admission control. In *INFOCOM*, 2011.
[32] N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. In *NSDI*, 2009.
[33] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in Facebook. In *WOSN*, 2009.
[34] X. Ying and X. Wu. Randomizing social networks: a spectrum preserving approach. In *SDM*, 2008.
[35] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against Sybil attacks. In *IEEE S&P*, 2008.
[36] H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman. SybilGuard: Defending against Sybil attacks via social networks. In *SIGCOMM*, 2006.
[37] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *PinKDD*, 2008.
[38] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, 2008.
[39] B. Zhou and J. Pei. The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowl. Inf. Syst.*, 28(1), 2011.

A Proof of Theorem 1: Relating vertex utility and mixing time.

We now sketch the proof of the above theorem. From the definition of total variation distance, we can see that:

$$||P_v^t(G')-\pi||_{tvd} \leq ||P_v^t(G')-P_v^t(G)||_{tvd}+||P_v^t(G)-\pi||_{tvd}\,.$$

From the definition of mixing time, we have that for all $t \geq \tau_G(\epsilon)$:

$$||P_v^t(G') - \pi||_{tvd} \leq ||P_v^t(G') - P_v^t(G)||_{tvd} + \epsilon\,.$$

Substituting $t = \tau_G(\epsilon)$ in the above equation, and taking the maximum over all vertices, we have that:

$$\max_v ||P_v^{\tau_G(\epsilon)}(G') - \pi||_{tvd} \leq$$
$$\max_v ||P_v^{\tau_G(\epsilon)}(G') - P_v^{\tau_G(\epsilon)}(G)||_{tvd} + \epsilon \leq$$
$$\mathsf{VU}_{max}(G, G', \tau_G(\epsilon)) + \epsilon\,.$$

Finally, we have that

$$\tau_{G'}(\epsilon + \mathsf{VU}_{max}(G, G', \tau_G(\epsilon)) \leq \tau_G(\epsilon)\,.$$

B Proof of Theorem 2: Relating vertex utility and SLEM.

We sketch the proof of the theorem. It is known that for undirected graphs, the second largest eigenvalue modulus is related to the mixing time of the graph as follows [1]:

$$\frac{\mu_{G'}}{2(1-\mu_{G'})}\log\left(\frac{1}{2\epsilon}\right) \leq \tau_{G'}(\epsilon) \leq \frac{\log n + \log\left(\frac{1}{\epsilon}\right)}{1-\mu_{G'}}\,.$$

From the above equation, we can bound the SLEM in terms of the mixing time as follows:

$$1 - \frac{\log n + \log(\frac{1}{\epsilon})}{\tau_{G'}(\epsilon)} \leq \mu_{G'} \leq \frac{2\tau_{G'}(\epsilon)}{2\tau_{G'}(\epsilon) + \log(\frac{1}{2\epsilon})}\,.$$

Set $\chi = \mathsf{VU}_{max}(G, G', \tau_G(\epsilon))$. Replacing ϵ with $\epsilon + \chi$, we have that

$$1 - \frac{\log n + \log(\frac{1}{\epsilon+\chi})}{\tau_{G'}(\epsilon+\chi)} \leq \mu_{G'} \leq \frac{2\tau_{G'}(\epsilon+\chi)}{2\tau_{G'}(\epsilon+\chi) + \log(\frac{1}{2\epsilon+2\chi})}\,.$$

Finally, from Theorem 1, we leverage $\tau_{G'}(\epsilon+\chi) \leq \tau_G(\epsilon)$ in the above equation to get

$$\mu_{G'} \leq \frac{2\tau_G(\epsilon)}{2\tau_G(\epsilon) + \log(\frac{1}{2\epsilon+2\chi})}\,.$$

C Proof of Theorem 4: Bounding mixing time.

Observe that the edges in graph G' can be modeled as samples from the t-hop probability distribution of random walks starting from vertices in G. We will prove the lower bound on the mixing time of the perturbed graph G' by contradiction: let us suppose that the mixing time of the graph G' is $k < \frac{\tau_G(\epsilon)}{t}$. Then in the original graph G, a user could have performed random walks of length $k \cdot t$ and achieve a variation distance less than ϵ. But $k \cdot t < \tau_G(\epsilon)$, which is a contradiction. Thus, we have that $\frac{\tau_G(\epsilon)}{t} \leq \tau_{G'}(\epsilon)$.

We prove an upper bound on mixing time of the perturbed graph using the notion of graph conductance. Let us denote the number of edges across the bottleneck cut (say S) of the original topology as g. Observe that the t-hop conductance between the sets S and $\overline{S}$ is strictly larger than the corresponding one hop conductance (since S is the bottleneck cut in the original topology). Thus, $E(G') \geq g$. Hence the expected graph conductance is an increasing function of the perturbation parameter t, and thus $\mathbb{E}(\tau_{G'}(\epsilon)) \leq \tau_G(\epsilon)$.

D Proof of Theorem 5: Relating utility and privacy.

From the definition of maximum vertex utility, we have that $|P^l_{AB}(G) - P^l_{AB}(G')| \leq \mathsf{VU}_{max}(G, G', l)$. Thus, we can bound $P^l_{AB}(G)$ as follows:

$$P^l_{AB}(G') - \chi \leq P^l_{AB}(G) \leq P^l_{AB}(G') + \chi\,,$$

where $\chi = \mathsf{VU}_{max}(G, G', l)$. Thus for any value of k, if $P^k_{AB}(G') - \mathsf{VU}_{max}(G, G', k) > 0$, then we have that the lower bound on the probability $P^k_{AB}(G) > 0$, which reveals the information that A and B are within an k hop neighborhood of each other. Thus the maximum information is revealed when the value of k is minimized, while maintaining $P^k_{AB}(G') - \mathsf{VU}_{max}(G, G', k) > 0$, i.e., $k = \delta$. This gives us a lower bound on the probability of A and B being friends in the original graph: the prior probability that two vertices in a δ hop neighborhood are friends: $f(\delta)$. Let m_δ denote the average number of links in a δ hop neighborhood, and let n_δ denote the average number of vertices in a δ hop neighborhood. In the special case of a null prior, we have that $f(\delta) = m_\delta / \binom{n_\delta}{2}$.