Link Privacy in Social Networks

Aleksandra Korolova
Dept. of Computer Science
Stanford University
Stanford, CA, USA
korolova@cs.stanford.edu

Rajeev Motwani
Dept. of Computer Science
Stanford University
Stanford, CA, USA
rajeev@cs.stanford.edu

Shubha U. Nabar
Dept. of Computer Science
Stanford University
Stanford, CA, USA
sunabar@cs.stanford.edu

Ying Xu
Dept. of Computer Science
Stanford University
Stanford, CA, USA
xuying@cs.stanford.edu

ABSTRACT

We consider a privacy threat to a social network in which the goal of an attacker is to obtain knowledge of a significant fraction of the links in the network. We formalize the typical social network interface and the information about links that it provides to its users in terms of lookahead. We consider a particular threat where an attacker subverts user accounts to get information about local neighborhoods in the network and pieces them together in order to get a global picture. We analyze, both experimentally and theoretically, the number of user accounts an attacker would need to subvert for a successful attack, as a function of his strategy for choosing users whose accounts to subvert and a function of lookahead provided by the network. We conclude that such an attack is feasible in practice, and thus any social network that wishes to protect the link privacy of its users should take great care in choosing the lookahead of its interface, limiting it to 1 or 2, whenever possible.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: Analysis of Algorithms and problem complexity—*Nonnumerical Algorithms and Problems*; J.4 [**Computer Applications**]: Social and behavioral sciences; H.2.8 [**Information Systems**]: Database Management—*Database applications*[Data mining]

General Terms

Theory, Experimentation, Security, Design

1. INTRODUCTION

Participation in online communities is becoming ubiquitous. Not only do people keep personal content such as their journals, photos, bookmarks and contacts online, they also increasingly interact online, both socially and professionally. In online communities whose primary goal is social networking, such as MySpace, Facebook, and LinkedIn, each user's set of trusted users is of paramount importance to their activity on the site. For example, in the case of LinkedIn, an online network of professionals, each connection signifies a professional relationship between two individuals, such as having worked together in the past. One's connections, and connections' connections, and so on, form a network that an individual has access to and can tap to foster professional connections or to find potential collaborators, clients, employers and subject experts.

A major part of the value of participating in an online social network or in a web-service with an online community, such as LiveJournal, for a user lies in the ability to leverage the structure of the *social network graph*. However, knowledge of this social graph by parties other than the service provider opens the door for powerful data mining, some of which may not be desirable to the users. For example, an employer might want to look at the network of a potential employee in order to evaluate its size and quality, or to approach random former colleagues of the individual with a request for references. An advertiser might want to look at the profiles and interests of people in the user's network and extended network, in order to more accurately infer the user's demographic and interest information for use in targeted advertisements.

Although some web-communities, such as LiveJournal, allow users to see all the links of any user in the network, the motivation for this paper is networks such as LinkedIn, where relationships between users may be sensitive to privacy concerns, and the link information is a valuable asset to the user and to the network owner. In such networks, a user is typically permitted only limited access to the link structure. For example, a LinkedIn user can only see the profiles and friends lists of his friends and the profiles of friends of friends. On Facebook, each user can specify whether to make their friend list and profile information visible only to their friends or to friends and friends of friends.

The most recent example of the value of link information to social network owners and users is Facebook's move to suspend the access of Google's Friend Connect program to Facebook's social graph [3] in May, 2008. Google's Friend Connect enables website developers to add social features to their website by supplying them with information about the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIKM'08, October 26–30, 2008, Napa Valley, California, USA.
Copyright 2008 ACM 978-1-59593-991-3/08/10 ...$5.00.

visitors' friends from social networks, and Facebook felt that "[that] doesn't respect the privacy standards [their] users have come to expect" [3]. Although a heated debate on Facebook's true motivation for suspending Google's Friend Connect ensued in the blogosphere [2], with some arguing that Facebook's move was motivated by their desire to prevent other entities from obtaining the social graph they worked hard to build rather than by their stated reason of user privacy, there is hardly any doubt that the desire for link privacy, from the perspective of the users or the social network owner, was at the core of Facebook's move.

Even though each user is given access only to a small part of the social network graph, one could imagine a resourceful adversarial entity trying to stitch together local network information of different users in order to gain global information about the social graph. In this paper, we analyze the methods one could employ to obtain information about the link structure of a social network, and the difficulty of doing that depending on the interface of neighborhood access permitted by the social network. We focus on the case in which an attacker, whose goal is to ascertain a significant fraction of the links in a network, obtains access to parts of the network by gaining access to the accounts of some select users. This is done either maliciously by breaking into user accounts or by offering each user a payment or a service in exchange for their permission to view their neighborhood of the social network. Both scenarios are realistic and common in practice. For example, both LiveJournal and Facebook have experiences successful account hijacking attempts recently [16], [14], and the accounts of other users might have been accessed without their knowledge. An example when a user voluntarily grants access to their friends list in exchange for a service is an addition of applications developed by third parties through Facebook Platform or Bebo Platform. More than 95% of Facebook users have used at least one application built on Facebook Platform [1]. We describe both experimental and theoretical results on the success of such an attack in obtaining the link structure of a significant portion of the network, and make recommendations for the type of neighborhood access that a social network should permit to prevent such an attack and protect the privacy of its network and its users.

In Section 2, we discuss related work on privacy in social networks and models of social network graphs. Section 3 lays out a formal model of the kind of attacks we consider and the goal of the attacker. We present experimental results of the success of different attack strategies on both simulated and real world social network graphs in Section 4, and present a rigorous theoretical analysis in Section 5. We conclude in Section 6 with recommendations of actions for web service providers that would preserve user privacy.

2. RELATED WORK

There has been much recent interest in anonymized data releases. Backstrom et. al. [6] consider a framework where a social network owner announces the intention to release an anonymized version of a social network graph, i.e., a copy where true usernames are replaced with random ids but the network structure is unchanged, and the goal of an attacker is to uniquely identify the node that corresponds to a real world entity in this anonymized graph. They show that, if given a chance to create as few as $\Theta(\log(n))$ new accounts in the network prior to its anonymized release, an attacker can efficiently recover the connections between any $\Theta(\log^2(n))$ nodes chosen a-priori. This is achieved by first finding the new accounts that the attacker inserted into the network and working through the connections established between the attacker's accounts and the chosen targets to identify the targets. In [13], the authors experimentally evaluate how much background information about the structure of the neighborhood of an individual would be sufficient for an attacker to uniquely identify the individual in such an anonymized graph. In [25] the emphasis is on protecting the types of links associated with individuals in an anonymized release. Simple edge-deletion and node-merging algorithms are proposed to reduce the risk of sensitive link disclosure. [26] and [19] pursue the question of privacy as it relates to social networks from various other perspectives.

While the privacy attack model of [6] is very interesting and has received substantial research focus, in this paper we study the privacy in social networks from an entirely different angle. We consider a case where no underlying graph is released, and, in fact, the owner of the network would like to keep the entire structure of the graph hidden from any one individual. An attacker we consider does not have access to the entire anonymized structure of the graph, nor is his goal to de-anonymize particular individuals from that graph. In contrast, he aims to compromise the link privacy of as many individuals as possible by determining the link structure of the graph based on the local neighborhood views of the graph from the perspective of several non-anonymous users.

There has been considerable theoretical work in modeling the structure and evolution of the web graph and social networks. In [7] and [17] the preferential attachment model and the copying model are introduced as generative models for the web graph. Many variations and extensions of these models have been proposed, such as [9] and [10]. It has been observed that social networks are subject to the small-world phenomenon [15] and models such as [24] have been proposed to account for it. The model of [18] aims to account for all of the commonly found patters in graphs. The common theme in this research is a search for a random process that models how users establish links to one another. The various models succeed to differing extents in explaining certain properties of the web graph and social networks observed in practice.

In the attack strategies that we consider, the effectiveness of the attack is likely to depend on the underlying social graph and the degree distribution of its nodes, which is commonly known to be close to power law [23, 11]. In our theoretical analysis of the effectiveness of an attack, we use the configuration model of [8] and [5] that guarantees a power law distribution. Unlike the evolutionary models such as preferential attachment, this model does not consider the process by which a network comes to have a power law degree sequence; rather, it takes the power law degree distribution as a given and generates a random graph whose degree distribution follows such a power law (specifics of graph generation according to this model are described in Section 4.1.1). We could also use the preferential attachment or copying models for analysis, but a static model such as [8] or [5] suffices for our purpose and allows for simpler analysis.

As a side note, our theoretical and experimental results also have implications on the power of lookahead in speed-

ing up web crawls studied in [20]. [20] analyzes a particular crawling strategy where the crawler performs a random walk on the graph. Some of the strategies proposed in our paper can be potentially used for web crawling, and would provide larger coverage than a random walk crawler. A similar problem has been studied in [4], but as pointed out by [20], the main result of [4] does not hold.

3. THE MODEL

In this section we formalize the privacy threat drafted in the Introduction. We first define the primary goal of the privacy attack considered in this paper (Section 3.1); then discuss the knowledge of social networks available to users, and thus adversaries (Section 3.2); finally, we list possible attack strategies (Section 3.3).

3.1 Goal of the Attack

We view a social network as an undirected graph $G = (V, E)$, where the nodes V are the users and the edges E represent connections or interactions between users. Even though some online social networks, such as LiveJournal, allow one-directional links, many others, and especially those where the link information is sensitive and subject to privacy considerations, such as LinkedIn and Facebook, require mutual friendship. In those networks links between users are naturally modeled as undirected edges, and thus we consider undirected graphs in our subsequent discussion and analysis.

As was informally discussed in Section 1, the primary goal of the privacy attack is to discover the link structure of the network. Knowledge of the entire network is superior to knowledge of connections of a subset of individual users because it allows seamless application of commonly used graph mining algorithms, such as computation of the shortest path between two people, clustering, or study of diffusion processes. We measure an attack's effectiveness using the notion of *node coverage,* or simply *coverage*, which measures the amount of network graph structure exposed to the attacker.

Definition 1 (Node Coverage). *The fraction of nodes whose entire immediate neighborhood is known. We say that a node is* covered*, if and only if the attacker knows precisely which nodes it is connected to and which nodes it is not connected to.*

One may also consider measuring an attack's effectiveness using a notion of *edge coverage,* defined in one of the two following ways:

1. Edge coverage: the fraction of edges known to the attacker among all edges that exist in the graph. This notion of edge coverage does not account for the attacker's knowledge about non-existing edges, and is therefore not a comprehensive view of an attacker's knowledge.

2. Edge coverage: among all pairs of users, the fraction of pairs between which the attacker knows whether or not an edge exists. As will become clear in the following sections, our definition of node coverage is more sensible for the attack strategies we consider and implies the knowledge of edge coverage under this definition. Thus, throughout the paper we will use node coverage as the primary measure of an attack's effectiveness.

3.2 The Network through a User's Lens

As mentioned in Section 1, LinkedIn allows a user to see all edges incident to oneself, as well as all edges incident to one's friends. An online social network could choose the extent to which links are made visible to its users depending on how sensitive the links are and we quantify such choices using *lookahead*. We say that the social network has lookahead of 0 if a user can see exactly who he links to; it has lookahead 1 if a user can see exactly the friends that he links to as well as the friends that his friends link to. In general, we say that the social network has *lookahead* ℓ if a user can see all of the edges incident to the nodes within distance ℓ from him. Using this definition, LinkedIn has lookahead 1. In terms of node coverage, a lookahead of ℓ means that each node covers all nodes within distance ℓ from it; nodes at distance $\ell + 1$ are *seen* (i.e., their existence is known to the user), but not *covered* (i.e., their connections are not known to the user).

There are other variations on the type of access that a user can have to the social graph structure. For example, some networks allow a user to see the shortest path between himself and any other user, some display the path only if it is relatively short, some only display the length of the shortest path, and others let the user see the common friends he has with any other user. We ignore these additional options in our discussion, while noting that the presence of any of them reduces the difficulty of discovering the entire link structure.

In addition to the connection information, a typical online social network also provides a search interface, where people can search for users by username, name or other identifying information such as email or school affiliation. The search interface returns usernames of all users who satisfy the query, often with the numbers of friends of those users, i.e., the degrees of the nodes corresponding to those users in the social network graph, G. LinkedIn is an example of a social network that allows such queries and provides degree information.

We formalize the various aspects of social network interfaces that may be leveraged by attackers to target specific user accounts below:

- *neighbors(username, password, ℓ)*: Given a username with proper authentication information, return all users within distance ℓ and all edges incident to those users in the graph G;
- *exists(username)*: Given a username, return whether the user exists in the network;
- *degree(username)*: Given a username, return the degree (number of friends) of the user with that username. Note that *degree(username)* implies *exists(username)*;
- *userlist()*: Return a list of all usernames in the network.

In the above, only *neighbors()* requires authentication information, all other functions are publicly available. A social network might expose some or all of these functions to its users. For example, LinkedIn provides *neighbors(username, password, ℓ)* for $\ell = 0$ or 1, but not for $\ell > 1$; it also provides *exists(username)* and *degree(username)*. Most social networks do not expose *userlist()* directly; however, an attacker may be able to generate a near complete user list

through other functionalities provided by the network such as fuzzy name search or public profiles.

A particular network may expose only a subset of the above functions and even if all functions are available, their costs may vary greatly. Therefore, when we discuss attack strategies in the next section we list the functions required by each strategy, and when we evaluate and compare strategies there is a trade-off between the effectiveness of an attack and the complexity of the available interface it requires.

3.3 Possible Attack Strategies

Recall that each time the attacker gains access to a user account, he immediately covers all nodes that are at distance no more than the lookahead distance ℓ enabled by the social network, i.e., he learns about all the edges incident to these nodes. Thus by gaining access to user u's account, an attacker immediately covers all nodes that are within distance ℓ of u. Additionally, he learns about the existence of ("sees") all nodes within distance $\ell + 1$ from u. We call the users to whose accounts the attacker obtains access *bribed* users.

A natural question that arises is how an attack's success or attained node coverage vary depending on the strategy followed for picking the users to bribe. We list the strategies we study in the decreasing order of information needed for the attacker to be able to implement them and study the success of attacks following these strategies both experimentally and theoretically in Sections 4 and 5.

Benchmark-Greedy: From among all users in the social network, pick the next user to bribe as the one whose perspective on the network will give the largest possible amount of new information. More formally, at each step the attacker picks the node covering the maximum number of nodes not yet covered. For $\ell \leq 1$ this can be implemented if the attacker can access the degrees of all users in the network. However, for $\ell > 1$ it requires that for each node the attacker has access to all usernames covered by that node, which is not a primitive that we consider available to the attacker. Thus this strategy serves as a benchmark rather than as an example of a feasible attack – it is the optimal bribing algorithm that is computationally feasible when given access to the entire graph G. Note that by reduction to set cover, finding the optimal bribing set for a given G is NP hard, thus the best polynomial-time (computationally feasible) approximation algorithm is the greedy algorithm described.
Requires: G;

Heuristically Greedy: Pick the next user to bribe as the one who can offer the largest possible amount of new information, according to some heuristic measure. The heuristic measure is chosen so that the attacker does not need to know G to evaluate it. In particular, we consider the following strategy:

- **Degree-Greedy**: Pick the next user to bribe as the one with the maximum "unseen" degree, i.e., its degree according to the *degree(username)* function minus the number of edges incident to it already known by the adversary.
  Requires: *neighbors(username, password, ℓ), degree(username), userlist();*

Highest-Degree: Bribe users in the descending order of their degrees.
Requires: *neighbors(username, password, ℓ), degree(username), userlist();*

Random: Pick the users to bribe at random. Variations could include picking the users uniformly at random, or with probability proportional to their degrees, etc. In particular, we study one strategy in this category:

- **Uniform-Random**: Pick the users to bribe uniformly at random.
  Requires: *neighbors(username, pwd, ℓ), userlist();*

Crawler: This strategy is similar to the Heuristically Greedy strategy, but the attacker chooses the next node to bribe only from the nodes already seen (within distance $\ell+1$ of some bribed node). We consider one such strategy:

- **Degree-Greedy-Crawler**: From among all users already seen, pick the next user to bribe as the one with the maximum unseen degree.
  Requires: *neighbors(username, password, ℓ), degree(username)*;

Note that the **Degree-Greedy-Crawler** and **Uniform-Random** strategies are very easily implementable in practice on most social networks, since they do not require any knowledge of nodes that are not within the neighborhood visible to the attacker. Furthermore, the **Degree-Greedy-Crawler** strategy could also be used by web crawlers to crawl web pages more rapidly when each web page stores information about its lookahead.

4. EXPERIMENTAL RESULTS

We present experimental results from the application of the strategies from Section 3.3 to both synthetic and real world social network data. At a high level, our experiments explore the fraction, f, of nodes that need to be bribed by an attacker using the different bribing strategies in order to achieve $1 - \varepsilon$ node coverage of a social network with lookahead ℓ. Our experimental results show that the number of users an attacker needs to bribe in order to acquire a fixed coverage decreases exponentially with increase in lookahead. In addition, this number is also fairly small from the perspective of practical attack implementation, indicating that several of the attack strategies from Section 3.3 are feasible to implement in practice and will achieve good results.

We implemented and evaluated the following five strategies, ordered in the decreasing order of complexity of the social network interface needed for them to become feasible: **Benchmark-Greedy** (abbreviated as **Benchmark**); **Degree-Greedy** (abbrev. as **Greedy**); **Highest-Degree** (abbrev. as **Highest**); **Uniform-Random** (abbrev. as **Random**); **Degree-Greedy-Crawler** (abbrev. as **Crawler**).

4.1 Results on Synthetic data

4.1.1 Generating Synthetic Graphs

In order to measure the effectiveness of the different attack strategies, we generate random graphs with power-law degree distributions and apply our strategies to them. Following the motivation of Section 2, we use the configuration model in [5] to generate the graphs. The model essentially generates a graph that satisfies a given degree distribution, picking uniformly at random from all such graphs.

More specifically, let n be the total number of nodes in G, α ($2 < \alpha \leq 3$) be the power law parameter; let d_0 and d_{max}

be the minimum and maximum degree of any node in the graph, respectively. First, we generate the degrees of all the nodes $d(v_i), i = 1, \ldots, n$ independently according to the distribution $Pr[d(v_i) = x] = C/x^{\alpha}, d_0 \leq x \leq d_{max}$, where C is the normalizing constant. Second, we consider $D = \sum d(v_i)$ minivertices which correspond to the original vertices in a natural way and generate a random matching over D. Finally, for each edge in the matching, we construct an edge between corresponding vertices in the original graph. As a result, we obtain a random graph with a given power-law degree distribution. The graph is connected almost surely [12]. The graph has a few multi-edges and self-loops that we remove in our experiments, without affecting the power law degree distribution.

Furthermore, following the practice of [20], we cap d_{max}, the maximum number of connections that a user may have at $\sqrt{n}$, reflecting the fact that in a large enough social network, a single person, even a very social one, cannot know a constant fraction of all users.

We denote the fraction of nodes bribed by f, the number of nodes bribed by $k = fn$, and the coverage achieved by $1 - \varepsilon = \frac{number\ of\ nodes\ covered}{n}$.

4.1.2 Comparison of Strategies

We analyze the relative performance of five of the strategies proposed in Section 3.3 on random power-law graphs with $100,000$ nodes, $\alpha = 3$ and $d_{min} = 5$. We run each strategy on 10 power-law graphs generated as described in 4.1.1, with the aim of achieving coverage of 0.5 through 0.99. For each strategy, we average across the runs the fraction of nodes that need to be bribed with that strategy in order to achieve the desired coverage. This gives us f as a function of $1-\varepsilon$ for each strategy. We present the results for lookahead 1 and 2 in Figure 1.

The experimental results show that **Benchmark** has the best performance, i.e., to achieve a fixed coverage of $1 - \varepsilon$, **Benchmark** needs to bribe fewer nodes than any other strategy. However, as mentioned previously, **Benchmark** is not feasible to implement in practice because it requires knowledge of the entire graph structure, and so it can only serve as a benchmark upper bound on how good any given strategy can be.

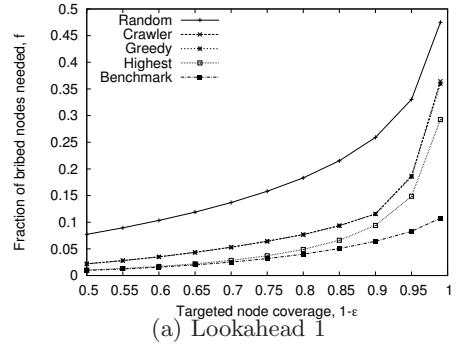
Some of the other observations we make are that **Highest** and **Benchmark** perform almost equally well when the desired coverage is less than 90%. However, the performance of **Highest** deteriorates as the lookahead increases and desired coverage increases.

Somewhat surprisingly, we find that **Greedy** performs worse than **Highest** while **Greedy** and **Crawler** perform equally well. Not surprisingly, **Random** performs the worst out of all the strategies.

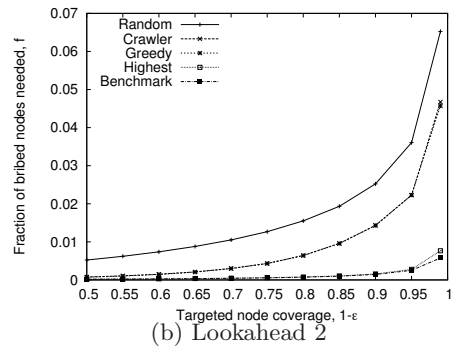
We choose the following three strategies to analyze in more detail and show that they can pose serious threats to link privacy: **Highest** and **Crawler** as a measure of performance of somewhat sophisticated yet still implementable strategies; and **Random** as the most easily implementable attack strategy that can serve as a lower bound on how well other strategies can work.

4.1.3 Dependence on the Number of Users

We analyze how performance of a bribing strategy changes with an increase in the number of nodes in the graph. We observe in Figure 2 that the number of nodes k that need



(a) Lookahead 1



(b) Lookahead 2

Figure 1: Comparison of attack strategies. We plot the fraction of bribed nodes against node coverage on synthetic graphs (with 100,000 nodes), using the five bribing strategies with lookahead 1 and 2. Lines for **Crawler** and **Greedy** are almost overlapping.

to be bribed using the **Highest** strategy in order to achieve a fixed coverage of $1 - \varepsilon$ is linear in the size of the network, for various values of ε and lookahead of 2. The same was observed for other values of lookahead but we omit those graphs for lack of space. Since **Highest** has the best performance among all the suggested realistically implementable strategies, this implies that k is linear in n for other strategies as well. However, it is worth observing that the slope of the linear function is very small, for all ε not very close to 1. As discussed in the next section, this makes all of the strategies a realistic threat at lookaheads greater than 1.

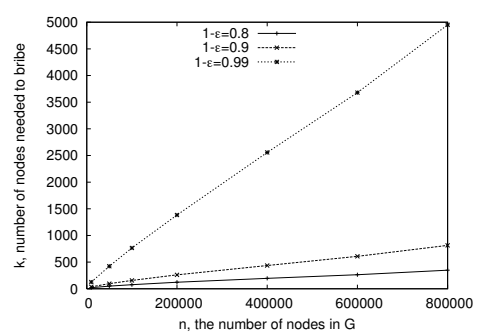


Figure 2: Number of nodes that need to be bribed for graph sizes n using Highest with lookahead 2 for coverage $0.8, 0.9, 0.99$**.**

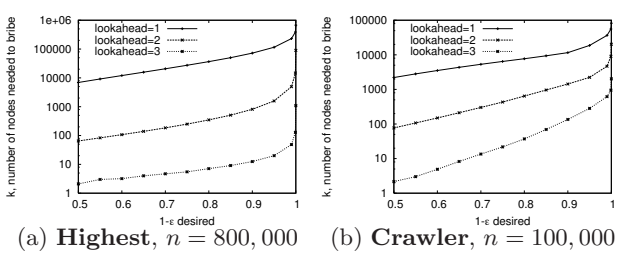
4.1.4 Dependence on Lookahead

The performance of all strategies substantially improves with increase in lookahead. Consider, for example, the performance of the **Highest** strategy, plotted in Figure 3 (a), and also detailed in Table 1.

1-ε	f_1/f_2	f_2/f_3
0.7	112.3	39.3
0.8	105.0	49.1
0.9	88.6	65.1
0.95	73.1	79.0
0.99	46.6	101.7

Table 1: Factors of improvement in performance of Highest with increases in lookaheads.

With each increase in lookahead, the number of nodes k that need to be bribed in order to achieve the same $1-\varepsilon$ coverage decreases by two orders of magnitude. In an 800,000-user social network, **Highest** needs to bribe 36,614 users in order to achieve a 0.8 coverage in a network with lookahead 1, but in the network of the same size with lookahead 2 **Highest** needs to bribe 348 users to achieve the same coverage, and only 7 users, if the lookahead is 3. In other words, the number of nodes that need to be bribed to achieve fixed coverage decreases exponentially in the lookahead, making the **Highest** strategy attack a feasible threat at lookahead 2 in social networks with under 1 million users, and a feasible threat at lookahead 3 in social networks with as many as 100 million users.



(a) **Highest**, $n = 800,000$ (b) **Crawler**, $n = 100,000$

Figure 3: Effect of lookahead. The figures show the number of nodes needed to bribe to achieve $1-\varepsilon$ coverage with various lookaheads, using **Highest** and **Crawler** respectively. Note that y axis is log scale.

We observe a similar exponential decrease with increase in lookahead in the number of nodes that need to be bribed for **Crawler** (Figure 3 (b)) and for **Random** (Figure omitted).

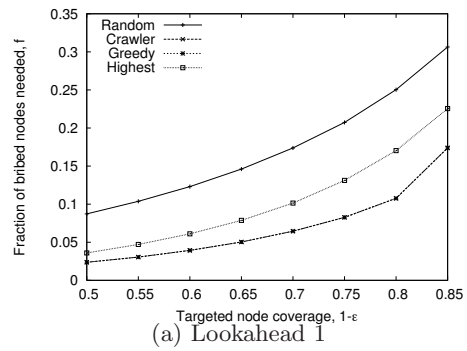
4.2 Results on Real data

As we felt that bribing LinkedIn users with a goal of recovering the network's structure would be inappropriate as a research exercise, we used the LiveJournal friendship graph, whose link structure is readily available, instead as a proxy. We crawled LiveJournal using the friends and friend-of listings to establish connections between users and extracted a connected component of 572,949 users.

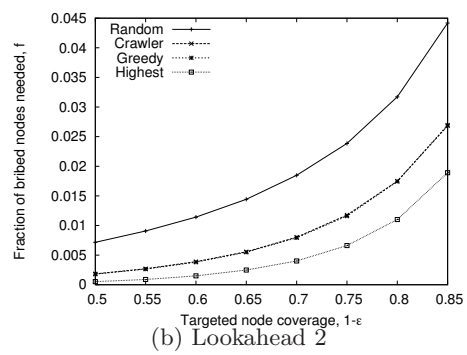
The obtained LiveJournal graph has an average degree of 11.8, $d_{min} = 1, d_{max} = 1974, \alpha = 2.6$.

4.2.1 Comparison of Strategies

Analogous to our discussion in Section 4.1.2 we compare the performance of the different bribing strategies on the LiveJournal graph at lookaheads of 1 and 2 in Figures 4 (a) and (b). The relative performance of the different strategies is the same as on the synthetic data, with the exception of **Highest** performing worse than **Crawler** and **Greedy** at lookahead 1. The **Crawler** and **Greedy** strategies also perform better on real data than on the synthetic data. Our intuition is that these differences are due to the disparities between properties of the graphs generated using the theoretical model and the real social network. The real social network graphs tend to contain a larger number of triangles than the graphs generated using the theoretical model (i.e., in practice, conditioned on edges (a, b) and (b, c), the edge (a, c) is more likely than random), with this local property likely leading to the **Crawler** and **Greedy** strategies being more effective.



(a) Lookahead 1



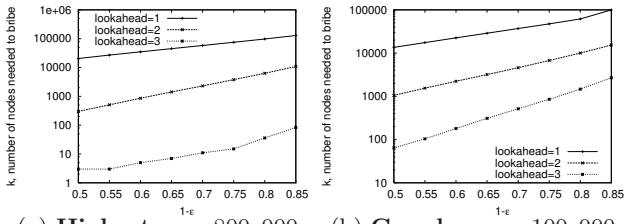
(b) Lookahead 2

Figure 4: Comparison of attack strategies on LiveJournal data. We plot the fraction of bribed nodes against node coverage on LiveJournal graph, using the four bribing strategies with lookahead 1 and 2. The two lines for **Crawler** and **Greedy** are almost overlapping.

4.2.2 Dependence on Lookahead

Furthermore, as on the synthetic data, the number of nodes that need to be bribed in order to achieve fixed coverage of LiveJournal decreases exponentially with an increase in lookahead (see Figure 5).

These experiments also confirm our hypothesis that while none of the strategies are a truly feasible threat at lookahead 1, some of them become feasible at lookahead 2, and all of them become feasible at lookahead 3. For example, in order to obtain 80% coverage of the 572,949-user LiveJournal graph using lookahead 2 **Highest** needs to bribe 6,308



(a) **Highest**, $n = 800,000$ (b) **Crawler**, $n = 100,000$

Figure 5: Effect of lookahead on LiveJournal data. The figures show the number of nodes needed to bribe to achieve $1-\varepsilon$ coverage with different lookaheads, using **Highest** and **Crawler** respectively. Note that y axis is log scale.

users, and to obtain the same coverage using lookahead 3 **Highest** needs to bribe 36 users – a number of users that is sufficiently small given the size of the network, and thus, feasible to bribe in practice.

5. THEORETICAL ANALYSIS FOR RANDOM POWER LAW GRAPHS

In this section we provide a theoretical analysis of the performance of two of the bribing strategies from Section 3: **Uniform-Random** and **Highest-Degree**. We analyze the fraction of nodes an attacker needs to bribe to reach a constant node coverage with high probability for a power law social network graph drawn from the configuration model described in Section 4.1.1. We carry out the analysis for power law graphs; for configuration models with other degree distributions, our analysis technique still applies, but the result depends on the specific degree distribution.

We use the same notation as in Section 4: n is the number of nodes in the network; m is the number of edges; d_0 is the minimum degree of a node; $\sqrt{n}$ is the maximum degree; α is the power law parameter; C is the normalizing constant for the degree distribution so that $\sum_{d=d_0}^{\sqrt{n}} Cd^{-\alpha} = 1$; the target node coverage is $1-\varepsilon$; f is the fraction of bribed nodes and $k = fn$ is the number of bribed nodes.

5.1 Analysis of Lookahead 1

Let us put ourselves in the shoes of an attacker and first answer the following question: if in each trial we cover a node randomly with probability proportional to its degree (all trials being independent), after how many trials will we have covered $(1-\varepsilon)n$ distinct nodes? Once we answer this question, we will come back to estimating the number of nodes to bribe by studying the rate at which different bribing strategies cover nodes. This question is similar to the well-known *coupon collector* problem if all nodes have an equal probability of being covered.

LEMMA 1. *[21] (Coupon Collector) Consider an unlimited supply of coupons of n distinct kinds. At each trial if we collect a coupon uniformly at random and independently of previous trials, then after t trials, the number of distinct coupons collected has the expectation $n(1-e^{-t/n})$ and is sharply concentrated.*

In our problem formulation, each node has a different probability of being covered (collected), thus it can be viewed as an instance of a weighted coupon collector problem. Schelling studied this problem in 1954 [22] when the probability of sampling each coupon is explicitly given. In our problem, not only do we need to consider the random choices of coupon collection, but also the random realization of the graph.

LEMMA 2. *In each trial we cover a node randomly with probability proportional to its degree, independently of previous trials. After $\frac{-\ln \varepsilon_0}{d_0} 2m$ trials, the number of distinct nodes covered is at least $n(1-\varepsilon-o(1))$ with high probability, where $\varepsilon = \sum_{d=d_0}^{\sqrt{n}} \varepsilon_0^{d/d_0} Cd^{-\alpha}$.*

The proof can be found in the Appendix. Both ε and ε_0 are between 0 and 1, and we can show that ε is always smaller than ε_0. Table 2 gives some values of ε and ε_0; for example, when $\alpha = 3$ and $d_0 = 5$, $\varepsilon = 0.4$ gives $\varepsilon_0 = 0.534$.

We now come back to the original question: how many nodes do we need to bribe in order to cover a $1-\varepsilon$ fraction of the graph, using different bribing strategies with lookahead 1? Remember that with lookahead 1 we cover a node only if it is a direct neighbor of a bribed node.

Pick a node to bribe using any strategy. Consider one edge of the bribed node, the other endpoint of the edge can be any node v and the probability of it being v is $d(v)/2m$ if we randomize over all graphs with the given degree sequence (this argument can be formalized using the Principle of Deferred Decisions [21]). Therefore, if we bribe a node with degree d and cover all its neighbors, it is equivalent to having made d trials to cover nodes in the graph. And if we bribe nodes $b_1, b_2, \ldots, b_k$ and cover all their neighbors, it is equivalent to having made $D = \sum_{i=1}^{k} d(b_i)$ such trials. However, not every trial covers a node v with the same probability proportional to its degree: if v was already covered in a previous trial, the probability of covering it again decreases, whereas if it was not covered in a previous trial, the probability of covering it with each new trial increases. More formally, the events that a node is covered (collected) in different trials are negatively correlated. This only increases the number of distinct nodes we expect to cover and, therefore, the result in Lemma 2 on the number of distinct nodes collected can still serve as a lower bound. In summary, we have the following Theorem:

THEOREM 3. *Bribe nodes $b_1, b_2, \ldots, b_k$ (all b_is distinct) selected using an arbitrary strategy. Denote the sum of their degrees by $D = \sum_{i=1}^{k} d(b_i)$. If $D = \frac{-\ln \varepsilon_0}{d_0} 2m$, then the node coverage is at least $1-\varepsilon-o(1)$ with high probability under lookahead 1, where $\varepsilon = \sum_{d=d_0}^{\sqrt{n}} \varepsilon_0^{d/d_0} Cd^{-\alpha}$.*

Theorem 3 establishes the connection between the total degree of bribed nodes (regardless of the strategy for choosing nodes to bribe) and the attained node coverage. In order to complete the analysis of particular bribing strategies it remains to analyze the total degree of k nodes bribed by that strategy.

We first analyze the strategy of bribing nodes uniformly at random without replacement. In any graph, a node chosen uniformly at random has expected degree $\bar{d} = 2m/n$, and bribing k nodes yields expected total degree $D = 2mk/n$. Plugging this expected total degree into Theorem 3 we obtain the following Corollary:

COROLLARY 4. *If an attacker bribes $\frac{-\ln \varepsilon_0}{d_0} n$ nodes picked according to the* **Uniform-Random** *strategy, then he covers*

at least $n(1-\varepsilon-o(1))$ nodes with high probability, where $\varepsilon = \sum_{d=d_0}^{\sqrt{n}} \varepsilon_0^{d/d_0} C d^{-\alpha}$.

Next we analyze the **Highest-Degree** strategy. To apply Theorem 3, we compute the expected total degree of the top $k = fn$ nodes, where f is a constant. Let d be such that

$$\sum_{x=d+1}^{\sqrt{n}} Cx^{-\alpha} < f \leq \sum_{x=d}^{\sqrt{n}} Cx^{-\alpha}$$

When n is large we can use integration to approximate the sum and get the equation

$$\int_d^{\sqrt{n}} Cx^{-\alpha} dx = f$$

Recall that C is the normalizing constant satisfying

$$\int_{d_0}^{\sqrt{n}} Cx^{-\alpha} dx = 1.$$

Solving the equation, we get $C \approx (\alpha-1) \cdot d_0^{\alpha-1}$ and $d \approx d_0 k^{1/(1-\alpha)}$. When n is large and f is a constant, the smallest degree of the top fn nodes is sharply concentrated around d; thus, we can roughly assume it is d. Now the top fn nodes have a maximum degree $\sqrt{n}$ and a minimum degree d, and the probability of having degree x is proportional to $x^{-\alpha}$. Therefore, the expected sum of degrees of the top fn nodes is

$$fn \frac{\sum_{x=d}^{\sqrt{n}} xx^{-\alpha}}{\sum_{x=d}^{\sqrt{n}} x^{-\alpha}} \approx nC \int_d^{\sqrt{n}} xx^{-\alpha} dx \approx \frac{\alpha-1}{\alpha-2} d_0 n k^{\frac{\alpha-2}{\alpha-1}}$$

On the other hand, the overall total degree

$$2m = \sum_{x=d_0}^{\sqrt{n}} xCx^{-\alpha} n \approx \frac{\alpha-1}{\alpha-2} d_0 n$$

Therefore, the expected sum of the degrees of the top fn nodes is $D = 2mk^{\frac{\alpha-2}{\alpha-1}}$. When n is large and f is a constant, the smallest degree of the top fn nodes sharply concentrates around d and the above analysis holds with high probability with a lower order error. Note that sharp concentration may not hold when $f = O(1/n)$, hence the assumption that f is a constant. We omit the detailed proof for lack of space.

COROLLARY 5. *If an attacker bribes $(\frac{-\ln \varepsilon_0}{d_0})^{\frac{\alpha-2}{\alpha-1}} n$ nodes picked according to the* **Highest-Degree** *strategy, then he covers at least $n(1-\varepsilon-o(1))$ nodes with high probability, where $\varepsilon = \sum_{d=d_0}^{\sqrt{n}} \varepsilon_0^{d/d_0} C d^{-\alpha}$.*

Even though Corollaries 4 and 5 only give lower bounds on the attained node coverage, our simulation results in Section 5.3 indicate that the analysis is close to being tight.

Compare the two strategies: to cover a certain fraction of the nodes, an attacker needs to bribe much fewer nodes when using the **Highest-Degree** bribing strategy than when using the **Uniform-Random** bribing strategy. For example, when $\alpha = 3$, if an attacker bribes an f fraction of the nodes with the **Uniform-Random** strategy, then he only needs to bribe an f^2 fraction of the nodes using **Highest-Degree** strategy to attain the same coverage. On the other hand, the bad news for an attacker targeting a social network that provides only lookahead of 1 is that even if he has the power to choose the highest degree nodes for an attack, a linear number of nodes will need to be bribed in order to cover a constant fraction of the whole graph (since the number of nodes needed to bribe is linear in n in both Corollaries).

5.2 Heuristic Analysis of Lookahead $\ell > 1$

Finally, we consider a social network with lookahead $\ell > 1$. As before, we analyze the fraction of nodes f that need to be bribed in order for the attacker to get a constant $(1-\varepsilon)$ coverage.

Our heuristic analysis shows that using the **Uniform-Random** strategy, f needs to be approximately $\frac{-\ln \varepsilon_0}{d_0 b^\ell}$ to attain $1-\varepsilon$ coverage, where ε and ε_0 satisfy the equation in Lemma 2 and b is of the order $\ln n$. When using the **Highest-Degree** strategy, the attacker needs to bribe approximately $f = (\frac{-\ln \varepsilon_0}{d_0 b^\ell})^2$, if $\alpha = 3$, fraction of users. The detailed heuristic analysis is included in the Appendix.

The heuristic analysis shows that the number of nodes needed to bribe decreases exponentially with increase in lookahead ℓ. For example, with lookahead $\ell = \ln \ln n$, bribing a constant number of nodes is sufficient to attain coverage of almost the entire graph, making the link privacy attacks on social networks with lookahead greater than 1 truly feasible.

5.3 Validating Theoretical Analysis With Simulation

We validate our theoretical analysis by simulation.

When the lookahead is 1, our theoretical analysis shows that in order to achieve a certain fixed node coverage, the number of nodes needed to bribe is linear in the total number of nodes in the social network, i.e., f is a constant with varying n. This matches and confirms our simulation results from Section 4.1.3.

Next we check whether the f values predicted by Corollaries 4 and 5 match simulation results (see Table 2). We observe that the f values obtained through simulation are smaller than those predicted in Corollaries 4 and 5. This is because Theorem 3, on which Corollaries 4 and 5 rely, gives **a lower bound** on the number of covered nodes. There are two factors responsible for the underestimation of the coverage attained in our theoretical analysis: (1) the different trials cover uncovered nodes with higher probability; (2) we did not count the bribed nodes as covered. The second factor responsible for the underestimation is more severe when the number of bribed nodes is not negligible in comparison to the number of covered nodes, which is especially true in the case of the **Uniform-Random** strategy. We can remedy this by taking into consideration the bribed nodes and refining our analysis. Using the same parameters as in Table 2, for $\varepsilon = 0.4, 0.2, 0.1$, the refined predicted fs for the **Uniform-Random** bribing strategy are $0.110, 0.204, 0.305$ respectively, which are closer to the simulation results, indicating that our theoretical analysis is fairly tight.

For lookahead $\ell > 1$, both the theoretical analysis and simulation results indicate that f decreases exponentially with the increase of lookahead ℓ. The predicted values are not too far from the actual results, although not as close as in case of lookahead 1. For example, for **Uniform-Random** with lookahead 2, to get 0.8-coverage ($\varepsilon = 0.2$), we predict $f = 0.0092$, while the simulation result is $f = 0.0145$.

		Uniform-Random		Highest-Degree	
ε	ε_0	f_p	f_s	f_p	f_s
0.4	0.534	0.125	0.103	0.016	0.015
0.2	0.309	0.235	0.183	0.055	0.045
0.1	0.173	0.350	0.259	0.123	0.090

Table 2: Predicted values vs simulation results. We compute f for varying ε, with two bribing strategies. We compute f : (1) by solving the equation in Corollary 4 and 5, shown in the column "f_p"; (2) by simulation, shown in the column "f_s". We use $\alpha = 3$ and $d_0 = 5$ in the table.

6. CONCLUSIONS

In this paper we provided a theoretical and experimental analysis of the vulnerability of a social network such as LinkedIn to a certain kind of privacy attack, namely, the link privacy attack. We proposed several strategies for carrying out such attacks, and analyzed their potential for success as a function of the lookahead permitted by the social network's interface. We have shown that the number of user accounts that an attacker needs to subvert in order to obtain a fixed portion of the link structure of the network decreases exponentially with increase in lookahead provided by the network owner. We conclude that social networks interested in protecting their users' link privacy ought to carefully balance the trade-off between the social utility offered by a large lookahead and the threat that such a lookahead poses to link privacy. We showed that as a general rule, the social network owners should refrain from permitting a lookahead higher than 2. Social networks may also want to decrease their vulnerability by not displaying the exact number of connections that each users has, or by varying the lookahead available to users depending on their trustworthiness.

7. ACKNOWLEDGEMENTS

This work was supported in part by NSF Grant ITR-0331640, TRUST (NSF award number CCF-0424422), and grants from Cisco, Google, KAUST, Lightspeed, and Microsoft.

8. REFERENCES

[1] *Facebook Press Release*. http://www.facebook.com/press/info.php?statistics, 2008.

[2] *TechCrunch*. http://www.techcrunch.com/2008/05/17/facebooks-glass-jaw/, 2008.

[3] *Technology Review*. http://www.technologyreview.com/Wire/20825/, 2008.

[4] L. Adamic, R. Lukose, A. Puniyani, and B. Huberman. Search in power-law networks. *Phys. Rev. E*, 2001.

[5] W. Aiello, F. Chung, and L. Liu. A random graph model for power law graphs. *IEEE Symposium on Foundations of Computer Science*, 2000.

[6] 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. ACM Press.

[7] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, (509), 1999.

[8] E. A. Bender and E. R. Canfield. The asymptotic number of labeled graphs with given degree sequences. *Journal of Combinatorial Theory, Series A 24, pp. 296-307*, 1978.

[9] B. Bollobas, C. Borgs, T. Chayes, and O. Riordan. Directed scale-free graphs. *ACM-SIAM Sypmosium on Discrete Algorithms*, 2003.

[10] C. Cooper and A. Frieze. A general model of web graphs. *Random Structures and Algorithms, 22(3), pp.311-335*, 2003.

[11] G. Csanyi and B. Szendroi. Structure of a large social network. *Physical Review E 69*, 2004.

[12] C. Gkantsidis, M. Mihail, and A. Saberi. Conductance and congestion in power law graphs. *Sigmetrics*, 2003.

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

[14] M. Kelly. *Facebook Security: Fighting the good fight*. Facebook blog, http://blog.new.facebook.com/blog.php?post=25844207130, August 7, 2008.

[15] J. Kleinberg. Navigation in a small world. *Nature*, (845), 2000.

[16] B. Krebs. *Account Hijackings Force LiveJournal Changes*. Washington Post, January 20, 2006.

[17] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. *IEEE Symposium on Foundations of Computer Science*, 2000.

[18] J. Leskovec, D. Chakrabarti, J. M. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In *PKDD*, pages 133–145, 2005.

[19] K. Liu, K. Das, T. Grandison, and H. Kargupta. Privacy-preserving data analysis on graphs and social networks. In H. Kargupta, J. Han, P. Yu, R. Motwani, and V. Kumar, editors, *Next Generation Data Mining*. CRC Press, 2008.

[20] M. Mihail, A. Saberi, and P. Tetali. Random walks with lookahead in power law random graphs. *Internet Mathematics*.

[21] R. Motwani and P. Raghavan. Cambridge University Press New York, NY, USA, 1995.

[22] H. von Schelling. Coupon collecting for unequal probabilities. *Am. Math. Monthly*, 61:306–311, 1954.

[23] S. Wasserman and K. Faust. Cambridge University Press, Cambridge, USA, 1994.

[24] D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature 393 440-442*, 1998.

[25] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *Proceedings of the International Workshop on Privacy, Security and Trust in KDD*, 2007.

[26] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 506–515, 2008.

APPENDIX

A. DETAILED PROOFS

A.1 Proof of Lemma 2

First, consider all nodes with degree d_0 in the graph. Let c_0 be the fraction of such nodes; $c_0 = \Theta(1)$ with high probability (see, for example, [20]). In each trial, the probability of covering a node with degree d_0 is $c_0 n d_0/2m$ (since the total sum of degrees is $2m$). In $\frac{-\ln \varepsilon_0}{d_0} 2m$ trials, in expectation, there are $-c_0 n \ln \varepsilon_0$ trials choosing nodes with degree d_0, and by Chernoff bound [21], there are at least $-(c_0 - o(1))n \ln \varepsilon_0$ such trials with high probability. All nodes with degree d_0 have an equal probability of being covered, so it is a classic coupon collector problem if constrained on such trials. By Lemma 1, the expected number of nodes with degree d_0 collected is at least

$$c_0 n(1 - e^{(c_0 - o(1))n \ln \varepsilon_0 / c_0 n}) = c_0 n(1 - \varepsilon_0 - o(1))$$

and by sharp concentration, the number of such nodes collected is at least $c_0 n(1 - \varepsilon_0 - o(1))$ with high probability.

Now consider nodes with degree $d_i = \Theta(1)$. Let c_i be the fraction of such nodes and again $c_i = \Theta(1)$ with high probability. By an argument similar to the above, there are at least $-(c_i - o(1))\frac{d_i}{d_0} n \ln \varepsilon_0$ trials choosing nodes with degree d_i, and the number of such nodes collected is at least $c_i n(1 - \varepsilon_0^{d_i/d_0} - o(1))$.

Finally, for all the remaining nodes with degree $\omega(1)$, the total number of such nodes is $o(n)$, so we miss at most $o(n)$ such nodes.

In total, with high probability we miss at most $\sum_{d_i} c_i n \varepsilon_0^{d_i/d_0} + o(n)$ nodes after $\frac{-\ln \varepsilon_0}{d_0} 2m$ trials. In the power law random graph model, $c_i = C d_i^{-\alpha} + o(1)$ with high probability, therefore, we miss at most $\sum_{d=d_0}^{\sqrt{n}} C d^{-\alpha} n \varepsilon_0^{d/d_0} + o(n)$, i.e., we collect at least $n(1 - \varepsilon - o(1))$ nodes. □

A.2 Heuristic Analysis with Lookahead $\ell > 1$

For simplicity, we use $\alpha = 3$; the analysis can be generalized to any $\alpha > 2$.

Denote by B the set of bribed nodes; by $N_\ell(B)$ the set of nodes whose shortest distance to B is exactly ℓ. Our goal is to estimate the number of nodes within distance ℓ, denoted by $D_\ell(B) = |\bigcup_{0 \le i \le \ell} N_i(B)|$ – then we have $f = |B|/n$, where $D_\ell(B) = (1 - \varepsilon)n$.

Let us first assume $N_\ell(B)$ is small enough such that there is no loop, i.e., $\bigcup_{0 \le i \le \ell+1} N_i(B)$ is a forest rooted at B. In reality there may exist a few loops, but it does not introduce too much error in estimating $D_\ell(B)$ when $N_\ell(B)$ is very small. Under this assumption, $|N_\ell(B)|$ is much larger than all $|N_i(B)|$s ($i < \ell$), so we can use $|N_\ell(B)|$ as an approximation to $D_\ell(B)$. To compute $|N_\ell(B)|$, we first study the expansion rate from N_ℓ to $N_{\ell+1}$, denoted by $b(\ell) = |N_{\ell+1}(B)|/|N_\ell(B)|$. Under the no-loop assumption, $b(\ell)$ equals to the average degree of nodes in $N_\ell(B)$ minus 1 (we need minus 1 to exclude the edges coming from $N_{\ell-1}(B)$). Note that nodes in $N_\ell(B)$ are not chosen uniformly at random; rather, they are chosen with probability proportional to their degrees because of the random realization of the graph. Therefore, the probability that such a node has degree x is proportional to xCx^{-3}, and consequently the expected degree of such node is

$$\frac{\sum_{x=d_0}^{\sqrt{n}} xxCx^{-3}}{\sum_{x=d_0}^{\sqrt{n}} xCx^{-3}} \approx d_0 \ln \frac{\sqrt{n}}{d_0}$$

Thus we have the expansion rate $b = d_0 \ln \frac{\sqrt{n}}{d_0} - 1$, independent of ℓ. It follows that $d_\ell(B) \approx |N_\ell(B)| \approx b|N_{\ell-1}(B)| \approx b^{\ell-1}|N_1(B)|$.

When $b|N_\ell(B)|$ is large, we can no longer use the above assumption to estimate $|N_{\ell+1}(B)|$: we still have $b|N_\ell(B)|$ edges incident to $N_{\ell+1}(B)$ but now some of the edges may share the same endpoints. This is the same as the weighted coupon collector problem in Lemma 2, so we can apply the result: if $b|N_\ell(B)| = \frac{-\ln \varepsilon_0}{d_0} 2m$, then $|N_{\ell+1}(B)| \approx n(1 - \varepsilon)$.

Now we compute the fraction of bribed nodes for $1 - \varepsilon$ node coverage, i.e., compute $f = |B|/n$, where B satisfies $D_\ell(B) = n(1 - \varepsilon)$. We need $b|N_{\ell-1}(B)| = \frac{-\ln \varepsilon_0}{d_0} 2m$ by Lemma 2, or $|N_{\ell-1}(B)| = \frac{-\ln \varepsilon_0}{d_0} 2m/b$. For large n, $b = \Theta(\ln n)$ is also large, so $|N_{\ell-1}(B)|$ is already small and we use the approximation $|N_{\ell-1}(B)| = b^{\ell-2}|N_1(B)|$. Thus we have $|N_1(B)| = \frac{-\ln \varepsilon_0}{d_0} 2m/b^{l-1}$. For the strategy of **Uniform-Random**, $|N_1(B)| = \bar{d} f n$, so approximately we need $f = \frac{-\ln \varepsilon_0}{d_0 b^\ell}$. For the strategy of **Highest-Degree**, $|N_1(B)| = 2\sqrt{f} d_0 n$ (given $\alpha = 3$), so we need $f = (\frac{-\ln \varepsilon_0}{d_0 b^\ell})^2$.