Wherefore Art Thou R3579X? Anonymized Social Networks, Hidden Patterns, and Structural Steganography

Lars Backstrom
Dept. of Computer Science
Cornell University, Ithaca NY
lars@cs.cornell.edu

Cynthia Dwork
Microsoft Research
dwork@microsoft.com

Jon Kleinberg
Dept. of Computer Science
Cornell University, Ithaca NY
kleinber@cs.cornell.edu

ABSTRACT

In a social network, nodes correspond to people or other social entities, and edges correspond to social links between them. In an effort to preserve privacy, the practice of anonymization replaces names with meaningless unique identifiers. We describe a family of attacks such that even from a single anonymized copy of a social network, it is possible for an adversary to learn whether edges exist or not between specific targeted pairs of nodes.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Theory, Measurement

Keywords

social networks, anonymization, privacy in data mining

1. INTRODUCTION

Anonymized Social Networks. Digital traces of human social interactions can now be found in a wide variety of on-line settings, and this has made them rich sources of data for large-scale studies of social networks. While a number of these on-line data sources are based on publicly crawlable blogging and social networking sites [6, 20, 22], where users have explicitly chosen to publish their links to others, many of the most promising opportunities for the study of social networks are emerging from data on domains where users have strong expectations of privacy — these include e-mail and messaging networks, as well as the link structure of closed (i.e. "members-only") on-line communities [1, 2, 17, 19, 21]. As a useful working example, consider a "communication graph," in which nodes are e-mail addresses, and there is a directed edge (u, v) if u has sent at least a certain number of e-mail messages or instant messages to v, or if v is included in u's address book. Here we will be considering the "purest" form of social network data, in which there are simply nodes corresponding to individuals and edges indicating social interaction, without any further annotation such as time-stamps or textual data.

In designing studies of such systems, one needs to set up the data to protect the privacy of individual users while preserving the global network properties. This is typically done through anonymization, a simple procedure in which each individual's "name" – e.g., e-mail address, phone number, or actual name – is replaced by a random user ID, but the connections between the (now anonymized) people – encoding who spoke together on the phone, who corresponded with whom, or who instant-messaged whom – are revealed. The motivation behind anonymizing is roughly as follows: while the social network labeled with actual names is sensitive and cannot be released, there may be considerable value in allowing researchers to study its structure. For such studies, including those cited above, researchers are not specifically interested in "who" corresponds to each node, but in the properties of the graph, such as its connectivity, node-to-node distances, frequencies of small subgraphs, or the extent to which it can be clustered. Anonymization is thus intended to exactly preserve the pure unannotated structure of the graph while suppressing the "who" information.

Can this work? The hope is that being handed an anonymized picture of a social network — just a graph with a random identifier attached to each node — is roughly akin to being given the complete social network of Mars, with the true Martian names attached to the nodes. Intuitively, the names are meaningless to earthdwellers: we do not "know" the Martians, and it is completely irrelevant to us whether a given node in the graph is labeled "Groark" or "Zoark". The difficulty with this metaphor, of course, is that anonymous social network data almost never exists in the absence of outside context, and an adversary can potentially combine this knowledge with the observed structure to begin compromising privacy, de-anonymizing nodes and even learning the edge relations between explicitly named (de-anonymized) individuals in the system. Moreover, such an adversary may in fact be a user (or set of users) of the system that is being anonymized.

For distinguishing among ways in which an adversary might take advantage of context, it is useful to consider an analogy to the distinction between passive attacks and active attacks in cryptanalysis — that is, between attacks in which an adversary simply observes data as it is presented, and those in which the adversary actively tries to affect the data to make it easier to decipher. In the case of anonymized social networks, passive attacks are carried out by individuals who try to learn the identities of nodes only after the anonymized network has been released. In contrast, an adversary in an active attack tries to compromise privacy by strategically creating new user accounts and links before the anonymized network

This work has been supported in part by NSF grants CCF-0325453, IIS-0329064, CNS-0403340, and BCS-0537606, by the Institute for the Social Sciences at Cornell, and by the John D. and Catherine T. MacArthur Foundation.

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

is released, so that these new nodes and edges will then be present in the anonymized network.

The present work: Attacks on anonymized social networks. In this paper we present both active and passive attacks on anonymized social networks, showing that both types of attacks can be used to reveal the true identities of targeted users, even from just a single anonymized copy of the network, and with a surprisingly small investment of effort by the attacker.

We describe active attacks in which an adversary chooses an arbitrary set of users whose privacy it wishes to violate, creates a small number of new user accounts with edges to these targeted users, and creates a pattern of links among the new accounts with the goal of making it stand out in the anonymized graph structure. The adversary then efficiently finds these new accounts together with the targeted users in the anonymized network that is released. At a theoretical level, the creation of $O(\sqrt{\log n})$ nodes by the attacker in an n-node network can begin compromising the privacy of arbitrary targeted nodes, with high probability for any network; in experiments, we find that on a 4.4-million-node social network, the creation of 7 nodes by an attacker (with degrees comparable to those of typical nodes in the network) can compromise the privacy of roughly 2400 edge relations on average. Moreover, experimental evidence suggests that it may be very difficult to determine whether a social network has been compromised by such an active attack.

We also consider passive attacks, in which users of the system do not create any new nodes or edges — they simply try to find themselves in the released network, and from this to discover the existence of edges among users to whom they are linked. In the same 4.4-million-node social network dataset, we find that for the vast majority of users, it is possible for them to exchange structural information with a small coalition of their friends, and subsequently uniquely identify the subgraph on this coalition in the ambient network. Using this, the coalition can then compromise the privacy of edges among pairs of neighboring nodes.

There are some obvious trade-offs between the active and passive attacks. The active attacks have more potent effects, in that they are guaranteed to work with high probability in any network (they don't force users to rely on the chance that they can uniquely find themselves after the network is released), and the attacker can choose any users it wants to target. On the other hand, while the passive attack can only compromise the privacy of users linked to the attacker, it has the striking feature that this attacker can simply be a user of the system who sees the anonymized network and indulges his or her curiosity; there is no observable "wrongdoing" to be detected. Moreover, since we find in practice that the passive attack will succeed for the majority of the population, it says in effect that most people in a large social network have laid the groundwork for a privacy-breaching attack simply through their everyday actions, without even realizing it.

These trade-offs naturally suggest the design of hybrid "semi-passive" attacks, in which a user of the system creates no new accounts, but simply creates a few additional out-links to targeted users before the anonymized network is released. As we show later, this can lead to privacy breaches on a scale approaching that of the active attack, without requiring the creation of new nodes.

We now summarize the results more fully, before moving on in subsequent sections to the details behind them.

The nature of the attacks. We assume the social network is an n-node graph $G = (V, E)$, representing interactions in an on-line system. Nodes correspond to user accounts, and an edge (u, v) indicates that u has communicated with v (again, consider the example of an e-mail or instant messaging network). The attacks become easier to carry out if the released graph data is directed; for most of the paper we will therefore consider the harder case of undirected graphs, in which we assume that the curator of the data — the agent that releases the anonymized network — eliminates the directions on the edges.

The active attacks will make use of the following two types of operations. First, an individual can create a new user account on the system; this adds a new node to G. Second, a node u can decide to communicate with a node v; this adds the undirected edge (u, v) to G. The goal of the attack is to take an arbitrary set of *targeted* users $w_1, \ldots, w_b$, and for each pair of them, to use the anonymized copy of G to learn whether the edge (w_i, w_j) in fact exists. This is the sense in which the privacy of these users will be compromised. (Other privacy compromises, such as learning the degree of a targeted user, also occur, but we focus our attention on learning about edges.)

The structure of the active attack is roughly as follows. Before the anonymized graph is produced, the attacker creates k new user accounts (for a small parameter k), and it links them together to create a subgraph H. It then uses these accounts to create links (e.g. by sending messages or creating address book entries) to nodes in $\{w_1, \ldots, w_b\}$, and potentially other nodes as well. Now, when the anonymized copy of G is released, this subgraph H will be present, as will the edges connecting H to $w_1, \ldots, w_b$. The attacker finds the copy of H that it planted in G, and from this it locates $w_1, \ldots, w_b$. Having identified the true location of these targeted users in G, the attacker can then determine all the edges among them, thereby compromising privacy.

There are a number of challenges in making this high-level approach actually work. First, if only a single copy of G is going to be released, then the attacker needs to construct H before having seen the structure of G. This means constructing a subgraph H that is likely to be uniquely identifiable in G, *regardless* of what G looks like. Second, the attacker needs to be able to efficiently find its copy of H hidden within G — in other words, it needs to create an instance of the subgraph isomorphism problem that is tractable to solve, even in a graph G with several million nodes.

The passive attack is based on the observation that most nodes in real social network data already belong to a small uniquely identifiable subgraph. Hence, if a user u is able to collude with a coalition of $k - 1$ friends after the release of the network, he or she will be able to identify additional nodes that are connected to this coalition, and thereby learn the edge relations among them.

It is also worth noting, however, that even the active attacks only involve the use of completely innocuous operations in the context of the system being compromised — the creation of new accounts, and the creation of links to existing accounts. In this sense, while the active attacker's aims are nefarious (and, in almost any imaginable scenario, prohibited either by research ethics guidelines or the terms of service of the system, or both), none of the individual steps from which the attack is constructed could be viewed at a syntactic level as "breaking into" parts of the system where it is not allowed. We also note, without going into the technical details here, that the active attacks are not something that degrade if many people are trying to execute them: even if many separate parties simultaneously run copies of the active attack, the high probability outcome is that all of them will succeed.

Parameters of the active attacks. To produce a subgraph likely to be unique in the network, an active attacker can use random generation: it creates k user accounts, and produces links by creating an edge between each pair independently at random.

We present two different active attacks employing this high-level idea, but differing in their specifics. For the first attack, we show that with $k = \Theta(\log n)$ new accounts, a randomly generated subgraph H will be unique with high probability, regardless of what G looks like and regardless of how H is attached to the rest of G. Moreover, if the maximum node degree in H is $\Theta(\log n)$, then H is recoverable efficiently, together with the identities of up to $b = \Theta(\log^2 n)$ targeted nodes to whom the attacker created links from H. The recovery algorithm for H uses a search over short walks in G, and accordingly we call it the *walk-based attack*.

In practice, k can be set to values even smaller than the bounds suggest, and recovery is very efficient. In computational experiments on a 4.4-million-node social network, a subgraph built using $k = 7$ new nodes, and degrees comparable to those of typical nodes in the network, can reveal an average of 70 targeted nodes, and hence the $\binom{70}{2} = 2415$ edge relations among them. We also provide evidence that it may be hard to detect whether such a subgraph H has been inserted into G; we will discuss the issue of detection in more detail below. Finally, we note that for the passive attack, we use the efficient recovery algorithm designed for this walk-based attack in order to identify a small coalition of existing nodes in the anonymized network.

The second active attack is similar in flavor; it also constructs H by including edges at random, but it attaches H to G using very few edges and recovers it using a more complex computation based on Gomory-Hu cut trees [16, 18]. Hence we will refer to it as the *cut-based attack*. The "thin" attachment of H to the rest of G implies that H will likely be unique and efficiently findable at an asymptotically smaller value of k: the cut-based attack uses $k = O(\sqrt{\log n})$ to reveal the identities of $\Theta(\sqrt{\log n})$ targeted nodes.

There are some trade-offs between the two active attacks. The walk-based attack comes with an extremely fast recovery algorithm that easily scales to millions of nodes, and it appears to be very hard to detect. The cut-based attack has the advantage of matching the tight theoretical bound on the number of nodes needed — we can show that an attacker must create at least $\Omega(\sqrt{\log n})$ new nodes in the worst case to begin compromising the privacy of arbitrary targeted nodes. The use of Gomory-Hu trees in the cut-based attack makes its recovery algorithm more expensive than that of the walk-based attack (though see the recent successes with Gomory-Hu computations on large-scale network analysis in [16]). Finally, the walk-based attack has the potential to compromise $\Theta(k^2)$ users, while the cut-based attack can only compromise $O(k)$, and it also appears easier to detect that the cut-based attack has taken place.

Related work. In a variety of settings different from the social network context here, recent work has considered ways of attacking anonymization and related schemes using content analysis of the text generated by users [7, 25], time series analysis of the time-stamps of user actions [24], or linkages among user records in different datasets [26]. In our case, however, both the passive and active attackers do not have access to highly resolved data like time-stamps or other numerical attributes; they can only use the binary information about who links to whom, without other node attributes, and this makes their task more challenging. Indeed, constructing the subgraph H can be seen as a kind of *structural steganography*, hiding secret messages for later recovery using just the social structure of G.

In this way, our approach can be seen as a step toward understanding how techniques of privacy-preserving data mining (see e.g. [8, 10, 12, 15, 23] and the references therein) can inform how we think about the protection of even the most skeletal social network data. We take up this discussion further in the final section.

2. THE WALK-BASED ATTACK

We begin by describing the specifics of the walk-based attack; we then analyze the method in Section 2.2, and report on computational experiments with it in Section 2.3.

2.1 Description of the Attack

Let $G = (V, E)$ be the n-node graph representing the anonymized social network that is released. As noted above, we consider the undirected case, in which there is an undirected edge (u, v) if at least one of the directed edges (u, v) or (v, u) is present. We focus on the undirected case because the attack becomes easier if the graph is directed.

Let us consider the problem from the perspective of the attacker. For ease of presentation, we begin with a slightly simplified version of the attack, and then show how to extend it to the attack we really use. We first choose a set of k named users, $W = \{w_1, \ldots, w_k\}$, that we wish to target in the network — we want to learn all the pairs (w_i, w_j) for which there are edges in G. To find each w_i in the anonymized graph, we use the following strategy. We first create a set of k new user accounts, $X = \{x_1, \ldots, x_k\}$, which will appear as nodes in the system. We include each undirected edge (x_i, x_j) independently with probability $1/2$. This produces a random graph H on X.

We also create an edge (x_i, w_i) for each i. (As discussed above, this involves having x_i send w_i a message, or include w_i in an address book, or some other activity depending on the nature of the social network.) For describing the basic version of the attack, we also assume that, because the account x_i corresponds to a fake identity, it will not receive messages from any node in $G - H$ other than potentially w_i, and thus will have no link to any other node in $G - H$. However, we will see later that the attack can be made to work even when this latter assumption does not hold.

When the anonymized graph G is released, we need to find our copy of H, and to correctly label its nodes as $x_1, \ldots, x_k$. Having found these nodes, we then find w_i as the unique node in $G-H$ that is linked to x_i. We thus identify the full labeled set W in G, and we can simply read off the edges between its elements by consulting G.

A number of technical ingredients are needed in order to make this plan work, based on whether certain subgraphs have the same structure as each other, and whether they have any internal symmetries. To express such questions, we use the following terminology. For a set of nodes S, we let $G[S]$ denote the subgraph of G induced by the nodes in S. An *isomorphism* between two sets of nodes S and S' in G is a one-to-one correspondence $f : S \to S'$ that maps edges to edges and non-edges to non-edges: (u, v) is an edge of $G[S]$ if and only if $(f(u), f(v))$ is an edge of $G[S']$. In this case, $G[S]$ and $G[S']$ are *isomorphic* — they are the same graph up to relabeling. An *automorphism* is an isomorphism from a set S to itself — a relabeling of the nodes $f : S \to S$ that preserves graph's structure. An automorphism f is *non-trivial* if it is not the identity function.

Thus, the construction of H succeeds if

(i) there is no $S \neq X$ such that $G[S]$ and $G[X] = H$ are isomorphic;

(ii) the subgraph H can be efficiently found, given G; and

(iii) the subgraph H has no non-trivial automorphisms.

If (i) holds, then any copy of H we find in G must in fact be the one we constructed; if (ii) holds, then we can in fact find the copy of H quickly; and if (iii) holds, then once we find H, we can correctly label its nodes as $x_1, \ldots, x_k$, and hence find $w_1, \ldots, w_k$.

The full construction is almost as described above, with the fol-

lowing three additions. First, the size of the targeted set W can be larger than k. The idea is that rather than connect each w_i to just a single x_i, we can connect it to a subset $N_i \subseteq X$, as long as w_i is the only node in $G - H$ that is attached to precisely the nodes in N_i — this way w_i will still be uniquely identifiable once H is found. Second, we will explicitly randomize the number of links from each x_i to $G - H$, to help in finding H. And third, to recover H, it is helpful to be able to traverse its nodes in order $x_1, x_2, \ldots, x_k$. Thus, we deterministically include all edges of the form (x_i, x_{i+1}), and randomly construct all other edges.

The Construction of H. With this informal discussion in mind, we now give the full specification of the attack.

(1) We choose $k = (2 + \delta) \log n$, for a small constant $\delta > 0$, to be the size of X. We choose two constants $d_0 \leq d_1 = O(\log n)$, and for each $i = 1, 2, \ldots, k$, we choose an *external degree* $\Delta_i \in [d_0, d_1]$ specifying the number of edges x_i will have to nodes in $G - H$. Each Δ_i can be chosen arbitrarily, but in our experiments with the algorithm, it works well simply to choose each Δ_i independently and uniformly at random from the interval $[d_0, d_1]$.

(2) Let $W = \{w_1, w_2, \ldots, w_b\}$ be the users we wish to target, for a value $b = O(\log^2 n)$. We also choose a small integer constant c ($c = 3$ will suffice in what follows). For each targeted node w_j, we choose a set $N_j \subseteq \{x_1, \ldots, x_k\}$ such that all N_j are distinct, each N_j has size at most c, and each x_i appears in at most Δ_i of the sets N_j. (This gives the true constraint on how large $b = O(\log^2 n)$ can be.) We construct links to w_j from each $x_i \in N_j$.

(3) Before generating the random internal edges of H, we add arbitrary further edges from H to $G - H$, so that each node x_i has exactly Δ_i edges to $G - H$. We construct these edges subject only to the following condition: for each $j = 1, 2, \ldots, b$, there should be no node in $G - H$ other than w_j that is connected to precisely the nodes in N_i.

(4) Finally, we generate the edges inside H. We include each edge (x_i, x_{i+1}), for $i = 1, \ldots, k - 1$, and we include each other edge (x_i, x_j) independently with probability $1/2$. Let Δ_i' be the degree of x_i in the full graph G (this is Δ_i plus its number of edges to other nodes in X).

This concludes the construction. As a first fact, we note that standard results in random graph theory (see e.g. [9]) imply that with high probability, the graph H has no non-trivial automorphisms. We will assume henceforth that this event occurs, i.e., that H has no non-trivial automorphisms.

Efficiently recovering H given G. When the graph G is released, we identify H by performing a search to find the path $x_1, x_2, \ldots, x_k$. We start at every node in G of degree Δ_1', and successively try adding any node with the correct degree and the correct edges back to nodes already on the path. In this way, we are pruning the search based on two kinds of tests: a *degree* test, that each possible candidate for node x_i should have the correct degree Δ_i'; and an *internal structure test*, that each possible candidate for node x_i should have edges to the correct subset of $\{x_1, x_2, \ldots, x_{i-1}\}$.

Here is a full description of the search algorithm.

(1) A rooted search tree $\mathcal{T}$ represents the progress of our search. Each node α in $\mathcal{T}$ other than the root corresponds to a node in G, which we will denote $f(\alpha)$, and the same node in G can potentially appear multiple times in $\mathcal{T}$. We construct $\mathcal{T}$ so that for every path of nodes $\alpha_1, \ldots, \alpha_\ell$ from the root, the corresponding nodes $f(\alpha_1), \ldots, f(\alpha_\ell)$ form a path in G with the same degree sequence and same internal edges as $x_1, \ldots, x_\ell$; and conversely every such path in G corresponds to a distinct rooted path in $\mathcal{T}$.

(2) We construct $\mathcal{T}$ by initially creating a dummy root node α^*. At any intermediate point in the construction, we take each current leaf node α, with a path $\alpha^* = \alpha_0, \alpha_1, \ldots, \alpha_\ell = \alpha$ leading to it, and we find all neighbors v of $f(\alpha)$ in G for which the degree of v is $\Delta_{\ell+1}'$, and $(f(\alpha_i), v)$ is an edge if and only if $(x_i, x_{\ell+1})$ is an edge for each $i = 1, \ldots, \ell$. For each such v, we create a new child β of α, with $f(\beta) = v$.

(3) Finally, if there is a unique rooted length-k path in $\mathcal{T}$, then this must correspond to the nodes of H, in order. Having found H, we then find the targeted nodes $w_1, \ldots, w_b$ owing to the fact that w_j is the only node with connections to precisely the nodes in N_j. Note that the total running time is only a small factor larger than the size of $\mathcal{T}$.

2.2 Analysis

To prove the correctness and efficiency of the attack, we show two things: with high probability the construction produces a unique copy of H in G, and with high probability, the search tree $\mathcal{T}$ in the recovery algorithm does not grow too large. It is important to stress that although these proofs are somewhat intricate, this complexity is an aspect of the *analysis*, not of the algorithms themselves. The construction of H and the recovery algorithm have already been fully specified in the previous subsection, and they are quite simple to implement. In keeping with this, we have structured this subsection and the next (on computational experiments) so they can be read essentially independently of each other.

We begin with the uniqueness result.

THEOREM 2.1. *Let $k \geq (2 + \delta) \log n$ for an arbitrary positive constant $\delta > 0$, and suppose we use the following process to construct an n-node graph G:*

(i) We start with an arbitrary graph G' on $n - k$ nodes, and we attach new nodes $X = \{x_1, \ldots, x_k\}$ arbitrarily to nodes in G'.

(ii) We build a random subgraph H on X by including each edge (x_i, x_{i+1}) for $i = 1, \ldots, k - 1$, and including each other edge (x_i, x_j) independently with probability $1/2$.

Then with high probability there is no subset of nodes $S \neq X$ in G such that $G[S]$ is isomorphic to $H = G[X]$.

Proof. To begin, let $\mathcal{F}_0$ be the event that there is no subset of nodes S disjoint from X such that $G[S]$ is isomorphic to H. (Note the difference between $\mathcal{F}_0$ and the statement of the theorem — in $\mathcal{F}_0$, we require that S be disjoint from X, not just unequal to X.) We first prove

(CLAIM 1.) *With high probability, the event $\mathcal{F}_0$ holds.*

We can prove Claim 1 by adapting a short argument with its roots in lower bounds for Ramsey numbers [5, 14]. For an ordered sequence $S = (s_1, s_2, \ldots, s_k)$ of k nodes in $G - H$, let $\mathcal{E}_S$ denote the event that the function $f : S \to X$ given by $f(s_i) = x_i$ is an isomorphism. Since all but $k - 1$ of the edges in H are chosen independently with probability $1/2$, and since S is disjoint from X, we have $\Pr[\mathcal{E}_S] = 2^{-\binom{k}{2}+(k-1)} = 2^{-\binom{k-1}{2}} = 2^{1+3k/2} \cdot 2^{-k^2/2}$.

Now $\mathcal{F}_0 = \cup_S \mathcal{E}_S$, where the union is over all sequences of k nodes from $G - H$. There are fewer than n^k such sequences S, so using the Union Bound and the fact that $n \leq 2^{k/(2+\delta)}$,

$$\begin{aligned} \Pr[\mathcal{E}] &< n^k \cdot 2^{-\binom{k-1}{2}} \leq 2^{k^2/(2+\delta)} \cdot 2^{1+3k/2} \cdot 2^{-k^2/2} \\ &= 2^{[-\delta k^2/2(2+\delta)]+3k/2+1}, \end{aligned}$$

which goes to 0 exponentially quickly in k. This completes the proof of Claim 1.

This short argument for why $\mathcal{F}_0$ holds provides the technical intuition behind the more general statement of the theorem. All the remaining complexity in the proof comes from sets S that may partially overlap X — and indeed this is the trickier kind of S to deal with, since one can try to construct an isomorphism with H by combining large parts of H with a few extra nodes elsewhere in the graph.

Due to this complexity, we need two facts asserting that H does not have much internal symmetry. For the second, we use the following definition: a node v is a *fixed point* of an isomorphism $f : S \to S'$ if $v \in S \cap S'$ and $f(s) = s$.

(CLAIM 2.) *For any constant $c_1 > 4$, let $\mathcal{F}_1$ denote the event that there are no disjoint sets of nodes Y and Z in H, each of size $c_1 \log k$, such that $H[Y]$ and $H[Z]$ are isomorphic. With high probability, the event $\mathcal{F}_1$ holds.*

(CLAIM 3.) *Suppose event $\mathcal{F}_1$ holds; then for any constant $c_2 \geq 3c_1$ the following holds. Let A, B, and Y be disjoint sets of nodes in G, with $B, Y \subseteq X$, and let $f : A \cup Y \to B \cup Y$ be an isomorphism. Then the set $f(A)$ contains at most $c_1 \log k$ nodes not in B, and the set Y contains at most $c_2 \log k$ nodes that are not fixed points of f.*

The proof of Claim 2 closely parallels that of Claim 1, and we omit this proof here.

To prove Claim 3, we build the following directed graph K on $A \cup B \cup Y$: if $f(v) = w$, then we include a directed edge from v to w. Note that in K, nodes in A have out-degree 1 and in-degree 0, nodes in B have out-degree 0 and in-degree 1, and nodes in Y have out-degree 1 and in-degree 1. Thus K consists of node-disjoint paths, cycles, and self-loops, with the cycles and self-loops fully in Y, and each path beginning at a node in A, possibly passing through nodes in Y, and ending at a node in B. We say a path component of K is *non-trivial* if it includes at least one node of Y.

First, note that there can be at most $c_1 \log k$ non-trivial path components in K; otherwise, if we let $Y' \subseteq Y$ consist of all the penultimate nodes on these paths, and $f(Y') = B' \subseteq B$, then Y' and B' are disjoint subsets of X, of size more than $c_1 \log k$ each, for which $H[Y']$ and $H[B']$ are isomorphic. This contradicts the assumption that $\mathcal{F}_1$ holds. It follows that $f(A)$ contains at most $c_1 \log k$ nodes not in B.

Next, let Z be the set of nodes in Y that are not fixed points of f. Nodes in Z correspond to the nodes on the cycle components in K, and the interior nodes on the path components. Suppose we choose every other edge on each cycle and each path (starting with the second edge on each path): we obtain at least $|Z|/3$ edges, since the worst case is a length-3 cycle, where we get only one edge. Let $Z_1 \subseteq Z$ be the tails of all these edges, and let $Z_2 \subseteq Z \cup B$ be their heads. Then $f(Z_1) = Z_2$, and so $G[Z_1]$ and $G[Z_2]$ are isomorphic. But Z_1 and Z_2 are disjoint subsets of X, so since $\mathcal{F}_1$ holds, we have $|Z_1| = |Z_2| \leq c_1 \log k$, and hence $|Z| \leq 3c_1 \log k \leq c_2 \log k$. This completes the proof of Claim 3.

Finally, we set up the calculation that will conclude the proof of the theorem. Suppose events $\mathcal{F}_0$ and $\mathcal{F}_1$ hold, and that there is a non-empty set $A \subseteq V - X$ such that, for some non-empty $Y \subseteq X$, the subgraph $G[A \cup Y]$ is isomorphic to H. Let $f : A \cup Y \to X$ be the isomorphism, and $B = X - Y$. Let $C = f(A)$, and D be the set consisting of all nodes of Y that are not fixed points of f. By Claim 3, we have $|C - B| \leq c_1 \log k$ and $|D| \leq c_2 \log k$. Thus, if $j = |A| = |B| = |C|$, then the set of fixed points $Y' = Y - D - C$ has size at least $k - (c_1 + c_2) \log k - j$. We write $k' = k - (c_1 + c_2) \log k$; since $k = (2 + \delta) \log n$, we have $k' \geq (2 + 2\delta_1) \log n$ for a smaller constant $\delta_1 > 0$ and n sufficiently large.

To show that there is unlikely to be a second copy of H in G, we search over all possible choices for A, B, C, and D within the appropriate size bounds. (We keep track of the order of the elements in A and C, which encodes the bijection between them.) Thus, let $\mathcal{E}_{ABCD}$ be the event that $G[A \cup Y]$ is isomorphic to H (where $Y = X - B$), via an isomorphism f in which $C = f(A)$ and all elements in $Y' = Y - D - C$ are fixed points of f. At most $j - 1$ edges inside C belong to the path $x_1, x_2, \ldots, x_k$ for which edges were explicitly included; thus, at least $\binom{j}{2} - (j - 1)$ edges inside C are randomly generated. In order for $\mathcal{E}_{ABCD}$ to hold, all of these must match the corresponding edges inside A (recall that we are keeping track of the ordering of A and C). Similarly, the $\geq (k' - j)j - 2j$ random edges created between C and Y' match those between A and Y'.

Since $(k' - j)j + \binom{j}{2} - 3j \geq \frac{1}{2}k'j - \frac{7}{2}j$, we have

$$\Pr[\mathcal{E}_{ABCD}] \leq 2^{-\frac{1}{2}k'j + \frac{7}{2}j} \leq 2^{-j(1+\delta_1)\log n} 2^{\frac{7}{2}j} = n^{-(1+\delta_1)j} 2^{\frac{7}{2}j}.$$

Finally,

$$\begin{aligned}\Pr[\mathcal{E}] &\leq \sum_{A,B,C,D} \Pr[\mathcal{E}_{ABCD}] \leq \sum_{j \geq 1} n^j k^{2j} k^{c_2 \log k} n^{-(1+\delta_1)j} 2^{\frac{7}{2}j} \\ &= \sum_{j \geq 1} k^{c_2 \log k} \left(\frac{2^{\frac{7}{2}} k^2}{n^{\delta_1}} \right)^j,\end{aligned}$$

and this last expression goes to 0 as n increases. ■

Since the running time of the recovery algorithm is only a small factor larger than the total number of nodes in the search tree $\mathcal{T}$, we can bound the running time by bounding the size of this tree.

THEOREM 2.2. *For every $\varepsilon > 0$, with high probability the size of $\mathcal{T}$ is $O(n^{1+\varepsilon})$.*

Proof. Recall that k denotes the size of H, and let d be the maximum degree (in G) of a node in H. Both of these quantities are $O(\log n)$. Let Γ' be a random variable equal to the number of paths of $G - H$ corresponding to some node of $\mathcal{T}$, and let Γ'' be the number of paths in G that meet H and correspond to some node in $\mathcal{T}$. Then $\Gamma = \Gamma' + \Gamma''$ is the number of nodes in $\mathcal{T}$, the quantity we are seeking to bound. We will show how to bound $E[\Gamma]$, assuming that the events $\mathcal{F}_0$ and $\mathcal{F}_1$ from the proof of Theorem 2.1 hold.

We first bound $E[\Gamma']$, as follows. For a path P in $G - H$, let $\Gamma'_P = 1$ if P corresponds to a node in $\mathcal{T}$, and $\Gamma'_P = 0$ otherwise. Call P *feasible* if the degree of every node on P is at most d. If P is not feasible, then $\Gamma'_P = 0$ with probability 1. Now consider a feasible P of length $j \leq k$; for P to be represented in $\mathcal{T}$, we need the edges among nodes on P to match the edges among $\{x_1, x_2, \ldots, x_j\}$. We can imagine the edges among $\{x_1, x_2, \ldots, x_j\}$ (other than those on the known path $x_1, x_2, \ldots, x_j$) as being generated after P is chosen, so $E[\Gamma'_P] = \Pr[\Gamma'_P = 1] = 2^{-\binom{j-1}{2}}$. The total number of feasible P of length j is at most nd^{j-1}. Thus,

$$E[\Gamma'] \leq n \sum_{j=1}^{k} d^{j-1} 2^{-\binom{j-1}{2}} = n \sum_{j=1}^{k} (d/2^{(j-2)/2})^{j-1}.$$

Once j is $\Theta(\log \log n)$, each term inside the sum is $O(1)$, so

$$E[\Gamma'] \leq nkd^{O(\log \log n)} = O(n 2^{O((\log \log n)^2)}).$$

Now we bound Γ'', by decomposing it into a separate random variable for each possible pattern in which a path could snake in and out of H. Thus, we say that a *template* τ is a sequence of $\ell \leq k$ symbols $(\tau_1, \ldots, \tau_\ell)$, where symbol τ_i is either a distinct node of H or a special symbol $*$. We call the set of all nodes of H that appear in τ the *support* of τ, and denote it $s(\tau)$. We will say

that a path P in G is *associated* with τ if the i^{th} node on P lies in $G - H$ for $\tau_i = *$, and otherwise is equal to $\tau_i \in X$. Finally, we say that the *reduction* of τ, denoted $\overline{\tau}$, is the template for which $\overline{\tau}_i = *$ whenever $\tau_i = *$, and for which $\overline{\tau}_i = x_i$ otherwise. (We will call such a $\overline{\tau}$ a *reduced template.*)

Let Γ''_τ be a random variable equal to the number of paths P associated with τ that are represented in the search tree $\mathcal{T}$. If at least one such path exists, then there is an isomorphism $f : s(\tau) \to s(\overline{\tau})$ given by $f(x_r) = x_i$ when $\tau_i = x_r$. Since we are assuming that $\mathcal{F}_1$ holds, Claims 2 and 3 from the proof of Theorem 2.1 imply that at all but at most $O(\log k)$ nodes are fixed points of f, and hence that τ agrees with $\overline{\tau}$ on all but $O(\log k)$ positions. Hence, the only templates τ for which Γ''_τ can be non-zero are those that differ in at most $O(\log k)$ positions from a reduced template.

We further decompose Γ''_τ into a sum over random variables $\Gamma''_{\tau P}$, for feasible paths P associated with τ, where $\Gamma''_{\tau P} = 1$ if P is represented in $\mathcal{T}$, and 0 otherwise. Now, there are at most k^j reduced templates with j $*$'s, and hence at most $k^{O(\log k)} \cdot k^j$ arbitrary templates with j $*$'s for which Γ''_τ can be non-zero. For each such τ, there are at most d^j feasible paths P associated with τ. Each such P has a probability of at most $2^{-\binom{j-1}{2}}$ of being represented in $\mathcal{T}$. Summing over all j gives

$$\begin{aligned} E\left[\Gamma''\right] &\leq \sum_{\tau,P} E\left[\Gamma''_{\tau P}\right] \leq \sum_{j=1}^{k} k^j d^j k^{O(\log k)} 2^{-\binom{j-1}{2}} \\ &\leq k^{O(\log k)} \sum_{j=1}^{k} kd \left(\frac{kd}{2^{(j-2)/2}}\right)^{j-1} \end{aligned}$$

Once j is $\Theta(\log kd) = \Theta(\log \log n)$, each term is $O(1)$, so

$$\begin{aligned} E\left[\Gamma''\right] &\leq k^{O(\log k)} O(\log \log n)(kd)^{O(\log \log n)} \\ &= O(2^{O((\log \log n)^2)}). \end{aligned}$$

■

We conclude with some comments on the tests used in the recovery algorithm. Recall that as we build $\mathcal{T}$, we eliminate paths based on an internal structure check (do the edges among path nodes match those in H?) and a degree check (do the nodes on the path have the same degree sequence as H?). Although the proofs of Therems 2.1 and 2.2 use just the internal structure check to prove uniqueness and to bound the size of $\mathcal{T}$ respectively, it is very important in practice that the algorithm use both checks: as the experiments in the next subsection will show, one can get unique subgraphs at smaller vales of k, and with much smaller search trees $\mathcal{T}$, by including the degree tests. But it is interesting to note that since these theorems can be proved using only internal structure tests, the attack is robust at a theoretical level provided only that the attacker has control over the internal structure of X, even in scenarios where nodes elsewhere in the graph may link to nodes in X without the knowledge of the attacker. (In this case, we still require that the targeted nodes $w_j \in W$ are uniquely identifiable via the sets N_j, and that all degrees in X remain logarithmic.)

2.3 Computational Experiments

Social Network Data. We now describe computational experiments with the algorithm on real social network data drawn from an on-line setting. We find that the algorithm scales easily to several million nodes, and produces efficiently findable unique subgraphs for values of k significantly smaller than the upper bounds in the previous subsections.

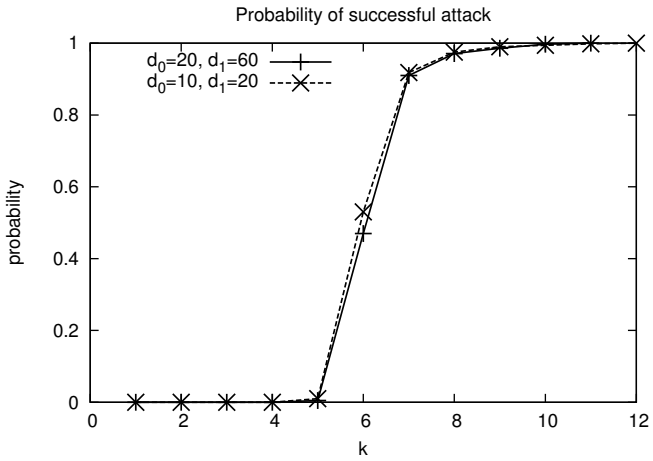

Figure 1: For two different choices of d_0 and d_1, the value $k = 7$ gives the attack on the LiveJournal graph a high probability of success. Both of these choices for d_0 and d_1 fall well within the degrees typically found in G.

As data, we use the network of friendship links on the blogging site LiveJournal, constructed from a crawl of this site performed in Februrary 2006. Each node in LiveJournal corresponds to a user who has made his or her blog public through the site; each user can also declare friendship links to other users. These links provide the edges of the social network we construct; they are directed, but we follow the principle of the previous subsections and convert them to undirected edges for purposes of the experiments. LiveJournal thus works well as a testbed; it has 4.4 million nodes and 77 million edges in the giant component of its undirected social network, and it exhibits many of the global structural features of other large online social networks. Finally, we emphasize that while LiveJournal has the right structure for performing our tests, it is not in reality an anonymous network — all of the nodes in the network represent users who have chosen to publish their information on the Web.

We simulate anonymization by removing all the user names from the nodes; we then run our attack and investigate the ranges of parameters in which it successfully identifies targeted nodes. As a first question, we examine how often H can be found uniquely for specific choices of d_0, d_1, and k. In our construction, we generate a random external degree Δ_i for each node x_i uniformly from $[d_0, d_1]$. We then create links to targeted nodes sequentially. Specifically, in iteration i we choose a new user w_i in $G - H$ to target; we then pick a minimal subset $X' \subseteq X$ that has not been used for any w_j for $j < i$, and where the degrees of nodes in X' are less than their randomly selected target degrees. We add an edge between w_i and each user in X'. We repeat this process until no such X' can be found. If, at the end of the process, some nodes in X have not yet reached their target degrees, we add edges to random nodes in G (and remove nodes from W so that no two nodes are connected to the same subset of X).

Uniqueness. We say the construction *succeeds* if H can be recovered uniquely. Figure 1 shows the success frequency for two different choices of d_0 and d_1 (the intervals $[10, 20]$ and $[20, 60]$), and varying values of k. We see that the success frequency is not significantly different for our two choices. In both cases the number of nodes we need to add to achieve a high success rate is very small – only 7. With 7 nodes, we can attack an average of 34 and 70 nodes for the smaller and larger degree choices, respectively.

We also note that the degree tests are essential for producing unique identifiability of H at such a small value of k. In fact, each of the 734 possible Hamiltonian graphs on 7 nodes actually occurs in the LiveJournal social network, so it is only because of its degree sequence in G that our constructed subgraph H is unique. (Theorem 2.1 does guarantee that a large enough H will be unique purely based on its internal structure; this is compatible with our findings since the analyzed bound of $(2+\delta)\log n$ is larger than the value $k = 7$ with which we are succeeding in the experiments.)

Efficient Recovery. In addition to being able to find H reliably, we must be able to find H quickly. We argued above that the size of $\mathcal{T}$ would remain sufficiently small that our search algorithm would be near-linear. In our experiments on the LiveJournal friendship graph we find that, in practice, the size of $\mathcal{T}$ is not much larger than the number of nodes u such that $d(u) = d(x_1)$. For instance, when $d_0 = 10$ and $d_1 = 20$, there are an average of 70,000 nodes which have $d(u) = d(x_1)$, while the size of $\mathcal{T}$ is typically about 90,000.

Detectability. Finally, we consider the *detectability* of the attack. Specifically, from the point of view of the attacker, it is important that the curator of the data, who is releasing the anonymized version, not be able to discover and remove H. As the curator does not have access to the secret degree sequence or the edges within H, they cannot employ the same algorithm the attacker uses to discover H. However, if H were to stand out significantly in some other way, there might be an alternate means for finding it.

This is a difficult issue to capture formally, but we provide the following indications that the subgraph H may be hard to discover. First is the simple fact that H has only 7 nodes, so it is difficult for any of its graph-theoretic properties to stand out with much statistical significance. Second, we describe some particular ways in which H does *not* stand out. To begin with, the internal structure of H is consistent with what is present in the network. For example, we have already mentioned that every 7-node Hamiltonian graph already occurs in LiveJournal, so this means that there are already subgraphs that exactly match the internal structure of H (even if not its pattern of attachment to G, which is also used to identify it). More generally, almost all nodes in LiveJournal are part of a very dense 7-node subgraph: If we look at all the nodes with degree at least 7, and consider the subgraph formed by those nodes and their 6 highest-degree neighbors, over 90% of such subgraphs have at least $11 > \frac{1}{2}\binom{7}{2}$ edges. These subgraphs are also almost all comparably well-connected to the rest of G.

3. THE CUT-BASED ATTACK

In the walk-based attack just presented, one needs to construct a logarithmic number of nodes in order to begin compromising privacy. On the other hand, we can show that at least $\Omega(\sqrt{\log n})$ nodes are needed in any active attack that requires a subgraph H to be uniquely identifiable with high probability, independent of both the structure of $G - H$ and the choice of which users to target.

It is therefore natural to try closing this gap between the $O(\log n)$ number of nodes used by the first attack, and the $\Omega(\sqrt{\log n})$ lower bound required in any attack. With this in mind, we now describe our second active attack, the *cut-based attack*; it matches the lower bound by compromising privacy using a subgraph H constructed on only $O(\sqrt{\log n})$ nodes. While the bound for the cut-based attack is appealing from a theoretical perspective, there are several important respects in which the walk-based attack that we saw earlier is likely to be more effective in practice. First, the walk-based attack comes with a much more efficient recovery algorithm; and second, the walk-based attack appears to be harder for the curator of the data to detect (as the cut-based attack produces a densely connected component attached weakly to the rest of the graph, which is uncommon in many settings).

The Construction of H. We begin the description of the cut-based attack with the construction of the subgraph H.

(1) Let b, the number of users we wish to target, be $\Theta(\sqrt{\log n})$, and let $w_1, w_2, \ldots, w_b$ be these users. First, for $k = 3b + 3$, we construct a set X of k new user accounts, creating an (undirected) edge between each pair with probability $1/2$. This defines a subgraph H that will be in G.

(2) Let $\delta(H)$ denote the minimum degree in H, and let $\gamma(H)$ denote the value of the minimum cut in H (i.e. the minimum number of edges whose deletion disconnects H). It is known that for a random graph H such as we have constructed, the following properties hold with probability going to 1 exponentially quickly in k [9]: first, that $\gamma(H) = \delta(H)$; second, that $\delta(H) \geq (1/2 - \varepsilon)k$ for any constant $\varepsilon > 0$; and third, that H has no non-trivial automorphisms. In what follows, we will assume that all these properties hold: $\gamma(H) = \delta(H) \geq k/3 > b$, and H has no non-trivial automorphisms.

(3) We choose b nodes $x_1, \ldots, x_b$ in H arbitrarily. We create a link from x_i to w_i, so that the edge (x_i, w_i) will appear in the anonymized graph G.

Efficiently recovering H given G. Now, when G is released, we identify the subgraph H and the targeted users $w_1, \ldots, w_b$ using the following recovery algorithm.

(1) We first compute the Gomory-Hu tree of G [16, 18] — this is an edge-weighted tree T on the node set V of G, such that for any $v, w \in V$, the value of the minimum v-w cut in G is equal to the minimum edge weight on the v-w path in T. Computing T is the most expensive step of the recovery algorithm, computationally. While the Gomory-Hu tree is constructable in polynomial time, it is significantly less efficient than the method employed by the walk-based attack. On the other hand, recent experiments in Web graph analysis indicate that Gomory-Hu tree computations can in fact be made to scale to very large graphs [16].

(2) We delete all edges of weight at most b from T, producing a forest T'. To find the set of nodes X we constructed, we iterate through all components of T' of size exactly k — let them consist of node sets $S_1, S_2, \ldots, S_r$ — and for each such S_i we test whether $G[S_i]$ is isomorphic to H. These isomorphism tests can be done efficiently, even by brute force, since $k! = o(n)$. Below, we prove that with high probability, there will be a single i such that $G[S_i]$ is isomorphic to H, and that S_i is equal to our set X of new nodes.

(3) Since H has no non-trivial automorphisms, from knowledge of S_i we can identify the nodes $x_1, \ldots, x_b$ that we linked to the targeted users $w_1, \ldots, w_b$ respectively. Hence we can identify the targeted users as well, which was the goal.

If we wish to target a much larger number b of users, we choose a number $b' = \Theta(\sqrt{\log n})$, and we partition the targeted users into sets $U_1, U_2, \ldots, U_s$, where $s = \lceil b/b' \rceil$, and each U_i except possibly the last has size b'. We then apply the above construction to each U_i, using a subgraph H_i chosen independently for the attack on U_i (and note that we still compromise edges between all pairs in $W = \cup_{i=1}^{s} U_i$).

Analysis of the cut-based attack. We focus on the version where $b = \Theta(\sqrt{\log n})$. The crux of the analysis is the proof of the following claim stated earlier.

THEOREM 3.1. *Let T be the Gomory-Hu tree of G, let T' be the forest obtained by deleting all edges of weight at most b, and let $S_1, S_2, \ldots, S_r$ be the node sets of all components of T' that have size exactly k. Then with high probability, there is a single i such that $G[S_i]$ is isomorphic to H, and the set S_i is equal to X.*

Proof. We first argue that X appears in the list of sets $S_1, \ldots, S_r$, and to do this, it is enough to show that X forms a single component in T'. Indeed, if $v, w \in X$ belonged to different components of T', then the v-w path in T would have to contain an edge of weight at most b, contradicting the fact that $\gamma(H) > b$. Further, if $v \in X$ and $w \notin X$ belonged to the same component of T', then the minimum v-w cut in G would have weight greater than b, contradicting the fact that there is a b-edge cut separating H from $G - H$.

Thus $S_i = X$ for some i. We now argue that with high probability, the subgraph $G[S_j]$ is not isomorphic to $H = G[X]$ for any $j \neq i$. Let $S_j = \{s_{j,1}, \ldots, s_{j,k}\}$, and let $X = \{x_1, \ldots, x_k\}$. For a bijection f from $\{1, 2, \ldots, k\}$ to itself, let $\mathcal{E}_{j,f}$ be the event that the subgraphs $G[S_j]$ and H are isomorphic under the mapping that sends $s_{j,i}$ to $x_{f(i)}$. Since the sets S_j and X are disjoint, $\Pr[\mathcal{E}_{j,f}] = 2^{-\binom{k}{2}}$. As long as $k \geq 1 + \sqrt{(2+\varepsilon)\log n}$ for any constant $\varepsilon > 0$, we have

$$\Pr[\mathcal{E}_{j,f}] = 2^{-\binom{k}{2}} \leq 2^{-(1+\varepsilon/2)\log n} = n^{-1-\varepsilon/2}.$$

We are interested the probability of the event $\mathcal{E} = \cup_{j,f} \mathcal{E}_{j,f}$. Since there are at most n/k possible sets S_j, we have

$$\Pr[\mathcal{E}] \leq \sum_{j,f} \Pr[\mathcal{E}_{j,f}] \leq (n/k)k! \cdot 2^{-\binom{k}{2}} \leq (k-1)! \cdot n^{-\varepsilon/2},$$

which goes to 0 with n since $k!$ grows more slowly than n^α for any constant $\alpha > 0$ when k is $O(\sqrt{\log n})$. ■

Some specific numbers for the cut-based attack. It is useful to supplement the asymptotic results for the cut-based attack with some specific numbers. If the network G has 100 million nodes, then by creating 12 new user accounts we can succeed in identifying 3 chosen users in the system with probability at least .99. Creating 15 new user accounts leads to a microscopically small failure probability.

The calculation is as follows. We first generate 100 random 12-node graphs $H_1, \ldots, H_{100}$, and see if any of them lacks non-trivial automorphisms and has a minimum cut of size at least 4. If any of them does, we choose one as our 12-node subgraph H. Computational experiments show that a random 12-node graph will have no non-trivial automorphism, and $\gamma(H) \geq 4$ with probability roughly 0.25. Thus, with probability well over 0.999, one of the 100 graphs H_i will have this pair of properties. Now, if we use the i^{th} of these random graphs in the construction, for a fixed i, then applying the argument and notation from the proof of Theorem 3.1, there are at most 8333333 possible components S_j of size 12 in the forest T', and so $\Pr[\mathcal{E}] \leq 8333333 \cdot 12! \cdot 2^{-66} < 6 \cdot 10^{-5}$. Hence the probability that *any* H_i will lead to non-uniqueness when attached to G is at most .006, and so in particular this holds for the H_i that we choose as H.

4. PASSIVE ATTACKS

In a passive attack, regular users are able to discover their locations in G using their knowledge of the local structure of the network around them. While there are a number of different types of passive attacks that could be implemented, here we imagine that a small coalition of passive attackers collude to discover their location. By doing so, they compromise the privacy of some of their neighbors: those connected to a unique subset of the coalition, and hence unambiguously recognizable once the coalition is found.

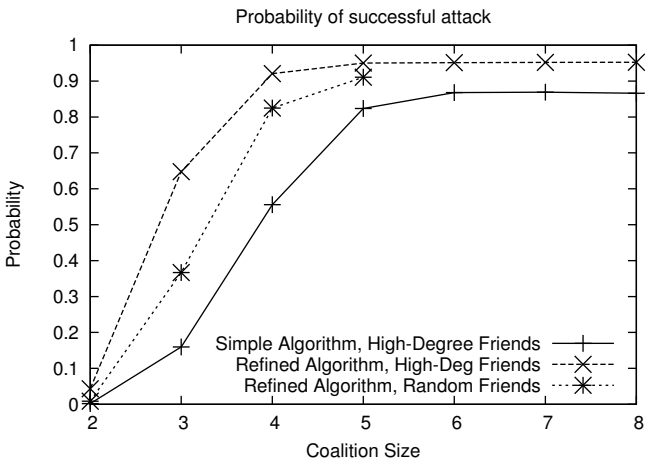

Figure 2: Probability of success for different coalition sizes, in the LiveJournal graph. When only the degrees and internal structure of the coalition are taken into account, a coalition of size 5 is needed to give a high probability of success. When the more refined version of the algorithm is used, and the edges connecting H to $G - H$ are considered, only 4 users need collude.

Here, we imagine that a coalition X of size k is initiated by one user who recruits $k-1$ of his or her neighbors to join the coalition. (Other structures could lead to analogous attacks.) We assume that the users in the coalition know the edges amongst themselves – the internal structure of $H = G[X]$, using the terminology from the active attack. We also assume that they know the names of their neighbors outside X. This latter assumption is reasonable in many cases: for example, if G is an undirected graph built from messages sent and received, then each user in X knows its incident edges. Other scenarios imply different levels of information: for example, if an undirected released network G is obtained from a directed graph where (u, v) indicates that v is in u's address book, then a node u does not necessarily know all its inbound edges, and hence doesn't know its full neighbor set in the undirected graph G. However, in the comparably plausible variant in which the directed version of an address book network is released, the nodes in X will have all the information they need for the passive attack.

This brings us to the details of the attack, which is analogous to the walk-based attack, except that the structure of H occurs organically as a natural function of individuals using the system. A user x_1 selects $k-1$ neighbors to form a coalition $X = \{x_1, x_2, \ldots, x_k\}$. The coalition knows whether the edge (x_i, x_j) is in G or not. The coalition also knows the neighbors outside X of each x_i. Once G is released, the coalition runs the same search algorithm described in the walk-based attack, with a minor modification due to the fact that H need not have a Hamiltonian path, but instead has a single node connected to all others.

To help the passive attack succeed, we can incorporate a further optimization that was not explicitly discussed earlier in the walk-based active attack experiments. For each non-empty set $S \subseteq \{1, 2, \ldots, k\}$, we let $g(S)$ denote the number of users to whom exactly the coalition members $\{x_i : i \in S\}$ are connected. Using this information, a path in $\mathcal{T}$, corresponding to nodes $f(\alpha_1), \ldots, f(\alpha_k = \alpha)$, must satisfy an additional constraint. If we define $g_\alpha(S)$ analogously to $g(S)$, but for the sequence $f(\alpha_1), \ldots, f(\alpha_k = \alpha)$ in-

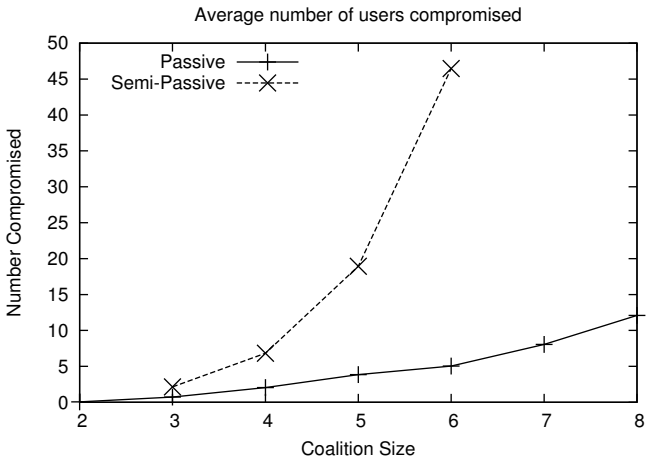

Figure 3: As the size of the coalition increases, the number of users in the LiveJournal graph compromised under the passive attack when the coalition successfully finds itself increases superlinearly. The number of users the semi-passive attack compromises increases exponentially.

stead of $x_1, \ldots, x_k$, then for α to correspond to a match of H, it must have $g(S) = g_\alpha(S)$, for all non-empty $S \subseteq \{1, \ldots, k\}$.

Once the coalition X finds itself, it is able to determine the identity of some subset of its neighbors in $G - X$. If a user w is connected to $\{x_i : i \in S\}$, and $g(S) = 1$, then the identity of the user w can be uniquely recovered in G. As the coalition has not specifically targeted any nodes, it is possible (and indeed likely for small coalitions) that although they can uniquely find themselves, they cannot locate any specific users other than themselves. However, empirically, we find that once a coalition is moderately-sized, it can compromise the privacy of at least some users.

Since the structure of H is not randomly generated, there is no *a priori* reason to believe that it will be uniquely findable, or that the above algorithm will run efficiently. Indeed, for pathological cases of G and H the problem is NP-Hard. However, we find on real social network data that the instances are not pathological, and that subgraphs on small coalitions tend to be unique and efficiently findable.

The primary disadvantage of this attack in practice, as compared to the active attack, is that it does not allow one to compromise the privacy of arbitrary users. However, a natural extension is a *semi-passive attack* whereby a coalition of existing users colludes to attack specific users. To do this, the coalition X forms as described above with x_1 recruiting $k-1$ neighbors. Next, the coalition compares neighbor sets to find some set $S \subseteq X$ such that $g(S) = 0$. Then, to attack a specific user w, each user in $\{x_i : i \in S\}$ adds an edge to w. Then, assuming that the coalition can uniquely find H, they will certainly find w as well.

4.1 Computational Experiments

Here we consider the passive attack on the undirected version of the LiveJournal graph. For varying k, we consider a coalition of a user x_1, and his or her $k-1$ highest-degree neighbors. (We also consider the case where x_1 selects $k-1$ neighbors at random; the success rate here is similar.) We do this for a randomly chosen sample of users x_1 whose degree is at least $k-1$. We then imagine that these users carry out the attack described above, searching all of G for a match. In our experiments, we consider both the simple version where the coalition uses only the internal structure of H and the degree sequence, and also the version where additional structure of the links between H and $G - H$ is used via the function $g(S)$.

We find that even coalitions as small as 3 or 4 users can often find themselves uniquely, particularly when using the refined version of the algorithm. Figure 2 summarizes the success rates for different-sized coalitions using both recovery algorithms. Furthermore, with minimal preprocessing, G can be searched for a particular coalition almost immediately: On a standard desktop, it takes less than a tenth of a second, on average, to find a coalition of size 6.

At first glance, these results seem at odds with the results for the active attack in Figure 1, as the passive attack is producing a higher chance of success with fewer nodes. However, in the active attack, we limited the degrees of the users created in an effort to remain inconspicuous. In the passive attack, there is no such limit, and many users' highest-degree neighbor has degree well over the limit of 60 that we imposed on the active attack. Since there are fewer users with higher degrees, this has the effect of increasing the findability of H. When we consider only those coalitions whose members all have degrees analogous to those in the active attack, the results are similar to the active attack.

While the above results show that a coalition can find itself easily, this does not mean that it can identify other nodes with certainty. Clearly, a coalition of size k cannot compromise more than $2^k - 1$ users, and in practice we see that the actual number is typically much smaller than this. Figure 3 shows the average number of users compromised by successful coalitions of various sizes. We see that even with a coalition of size 6, the number of compromised users tends to be small. However, with a semi-passive attack, we can greatly increase the number of users compromised.

Figure 3 shows the increased number of users typically compromised by the semi-passive attack (and recall that these users can be chosen arbitrarily by the coalition). Moreover, when the coalition is compromising as many users as possible, the semi-passive attack tends to have a higher success rate.

5. DISCUSSION

It is natural to ask what conclusions about social network data should be drawn from this work. As noted at the outset, our work is not directly relevant to all settings in which social network data is used. For example, much of the research into on-line social networks is conducted on data collected from Web crawls, where users have chosen to make their network links public. There are also natural scenarios in which individuals work with social network data under safeguards that are primarily legal or contractual, rather than computational, in nature — although even in such cases, there are compelling reasons why researchers covered by contractual relationships with a curator of sensitive data should still only publicly release the results of analyses that are carried out through a privacy mechanism, to prevent the information in these analyses from implicitly compromising privacy. In cases such as these, where computational safeguards are not the primary focus, important questions of data utility versus privacy still arise, but the questions in these cases are not something that our results directly address.

What our results do show is that one cannot rely on anonymization to ensure individual privacy in social network data, in the presence of parties who may be trying to compromise this privacy. And while one natural reaction to these results is to try inventing methods of thwarting the particular attacks we describe, we think this misses the broader point of our work: true safeguarding of privacy requires mathematical rigor, beginning with a clear description of what it means to compromise privacy, what are the computational and behavioral capabilities of the adversary, and to what information does it have access.

There is a growing literature to which we can turn for thinking about the problem of ensuring privacy in settings such as these. There has been extensive recent work on privacy-preserving data mining, beginning with [3, 4, 27], which rekindled interest in a field quiescent since the 1980s, and increasingly incorporating approaches from modern cryptography for describing and reasoning about information leakage (e.g. [15, 23, 10, 8, 12] and the references therein). The notion of ϵ-differential privacy gives very strong guarantees, independent of the auxiliary information and computational powers of the adversary [12, 11]. This notion departs from previous ones by shifting away from comparing what can be learned about an individual with versus without the database, instead concentrating on how the database behaves with versus without the data of an individual.

A general *interactive* mechanism for ensuring differential privacy is given in [12]. In such a mechanism, a question is posed, the exact answer is computed by the curator, and then a noisy version of the true answer is returned to the user. If the questions are known in advance then a good interactive mechanism directly yields a good non-interactive mechanism: the curator computes the answers to the questions, adds noise as appropriate, and releases this simulated transcript of an interactive conversation.

The advantage of interaction lies in the case in which there is no known, fixed-in-advance, small superset of questions to be asked – precisely the typical situation for current research on social networks. Intuitively, to be "useful" the curator must produce an object that answers all, or at least very many, questions fairly accurately, while simultaneously preserving privacy. The problem is not just one of conciseness: roughly speaking, the magnitude of the noise added to protect privacy must increase with the number of questions to be answered. This turns out to be inherent: there are natural cases for which any object (interactive or otherwise) that permits "too many" questions to be answered "too accurately" results in blatant non-privacy [10, 13].

In summary, the design of non-interactive mechanisms for ensuring reasonable notions of privacy in social network data is an open question, and potential results here are constrained by these existing impossibility results. Hence, when computational safeguards are sought to protect social network data, the only techniques of which we are aware at the present time for simultaneously ensuring individual privacy and permitting accurate analysis, when the questions are not known in advance, are interactive.

6. REFERENCES

[1] L. Adamic, E. Adar. How to search a social network. Social Networks 27(2005).

[2] L. Adamic, O. Buyukkokten, E. Adar. A Social Network Caught in the Web. First Monday, 8(2003).

[3] D. Agrawal. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. Proc. PODS, 2001.

[4] R. Agrawal, R. Srikant. Privacy-preserving data mining. Proc. SIGMOD, 2000.

[5] N. Alon, J. Spencer, *The Probabilistic Method*, 1992.

[6] L. Backstrom, D. Huttenlocher, J. Kleinberg, X. Lan. Group formation in large social networks: Membership, growth, and evolution. Proc. KDD, 2006.

[7] M. Barbaro, T. Zeller. A Face Is Exposed for AOL Searcher No. 4417749. New York Times, 9 August 2006.

[8] A. Blum, C. Dwork, F. McSherry, K. Nissim. Practical privacy: The SuLQ framework. Proc. PODS, 2005.

[9] B. Bollobás. Random Graphs. Cambridge, 2001.

[10] I. Dinur, K. Nissim. Revealing information while preserving privacy. Proc. PODC, 2003.

[11] C. Dwork. Differential Privacy, Proc. ICALP, 2006.

[12] C. Dwork, F. McSherry, K. Nissim, A. Smith. Calibrating noise to sensitivity in private data analysis. Proc. TCC, 2006.

[13] C. Dwork, F. McSherry, and K. Talwar, The Price of Privacy and the Limits of LP Decoding, submitted for publication.

[14] P. Erdos. Some remarks on the theory of graphs. Bull. AMS 53 (1947), 292–294.

[15] A. Evfimievski, J. Gehrke, R. Srikant. Limiting privacy breaches in privacy preserving data mining. Proc. PODS, 2003.

[16] G. Flake, R. Tarjan, K. Tsioutsiouliklis. Graph Clustering and Minimum Cut Trees. Internet Math. 1(2004).

[17] S. Golder, D. Wilkinson B. Huberman. Rhythms of Social Interaction: Messaging within a Massive Online Network. Proc. 3rd Intl. Conf. on Communities and Technologies, 2007.

[18] R. Gomory, T.C. Hu. (1961). Multi-Terminal Network Flows. SIAM J. Appl. Math., 9:551–570.

[19] G. Kossinets and D. J. Watts. Empirical Analysis of an Evolving Social Network. Science 311:88–90, 2006

[20] R. Kumar, R. Novak, P. Raghavan, A. Tomkins. Structure and evolution of blogspace. CACM, 47(2004).

[21] R. Kumar, J. Novak, A. Tomkins. Structure and Evolution of Online Social Networks. Proc. KDD, 2006.

[22] D. Liben-Nowell, R. Kumar, J. Novak, P. Raghavan, A. Tomkins. Geographic routing in social networks. PNAS 102(2005).

[23] N. Mishra, M. Sandler. Privacy via Pseudorandom Sketches Proc. PODS, 2006.

[24] A Narayanan, V. Shmatikov How To Break Anonymity of the Netflix Prize Dataset. arxiv cs/0610105, Oct. 2006.

[25] J. Novak, P. Raghavan, A. Tomkins. Anti-Aliasing on the Web. Proc. WWW, 2004.

[26] L. Sweeney. Weaving technology and policy together to maintain confidentiality. J Law Med Ethics, 25(1997).

[27] L. Sweeney. k-anonymity: A model for protecting privacy. Intl. J. Uncertainty, Fuzziness and Knowledge-based Systems, 10(2002).