DOI:10.1145/2043174.2043199

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

By Lars Backstrom, Cynthia Dwork, and Jon Kleinberg

**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.**

## 1. INTRODUCTION

### 1.1. Anonymized social networks

Digital traces of human social interactions can now be found in a wide variety of online settings, and this has made them rich sources of data for large-scale studies of social networks. While a number of these online data sources are based on publicly crawlable blogging and social networking sites, 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 email, phone, and messaging networks, as well as the link structure of closed (i.e., "members-only") online communities. As a useful working example, consider a "communication graph," in which nodes are email addresses, and there is a directed edge $(u, v)$ if $u$ has sent at least a certain number of email 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 timestamps 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"—for example, email 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, 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 earth-dwellers: 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.

A previous version of this paper was published in the *Proceedings of the 16th International Conference on World Wide Web: WWW2007* (Banff, Alberta, Canada, May 8–12, 2007).

In contrast, an adversary in an active attack tries to compromise privacy by strategically creating new user accounts and links before the anonymized network is released so that these new nodes and edges will then be present in the anonymized network.

### 1.2. 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 do not 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 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 outlinks 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.

In the next section, we provide some background and context for our work in terms of the broader area of data privacy. We then present our two main classes of active attacks on anonymized social networks; we refer to them as *walk-based attacks* and *cut-based attacks*, with the names reflecting the underlying techniques being used. We then describe the use of passive attacks and conclude with a general discussion.

## 2. RELATED WORK

This work fits within a growing literature that has considered ways in which private online data can be divulged against users' wishes, via carefully devised privacy-breaching attacks. Such attacks have been based on a variety of features in the data; for example, the queries entered by users into search engines can be used to uniquely identify them,[17] and the writing styles of users in online discussion can likewise be used to find the same person writing under different pseudonyms.[22] Temporal data can also be an effective feature in privacy-breaching attacks: since it is unlikely for two users to perform a nontrivial set of actions at almost exactly the same sets of times, the sequence of times at which a user performs these actions becomes a type of identifying signature.[20]

We note that in our case, both the passive and active attackers do not have access to highly resolved data like timestamps or other textual or 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, the secret subgraph $H$ constructed as part of our attacks can be thought of 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 fundamental techniques of data privacy (see, e.g., Dwork[9] and the references therein) can inform how we think about the protection of even the most skeletal social network data. We discuss this further in the final section.

In the time since the conference proceedings version of our work appeared, there has been continued research exploring mechanisms by which private data can be revealed online. Concurrent with our work, Hay et al.[15] considered a set of methods for identifying nodes in anonymized social networks by looking at successively larger neighborhoods of a node. More recently, Narayanan and Shmatikov[21] have shown how access to multiple networks containing overlapping sets of people can enable approaches to de-anonymization based on approximately aligning the portions of the networks that overlap.

In a related but different direction, several lines of recent work have shown how the principle of *homophily*—that neighbors in social networks have similar characteristics—can be used to discover private information: even if a user keeps profile information private, it can often be estimated from attributes in public profiles maintained by his or her friends.[16,19,25] Finally, recent work shows how social links themselves can be discovered from information about the times and places at which individuals perform activities; using data from a photo-sharing site, Crandall et al.[6] showed that if two users uploaded photos from approximately the same location at approximately the same time on multiple occasions, there was a sharply increased probability that they were linked on the site's social network.

## 3. THE WALK-BASED ATTACK

To set the stage for our first active attack, we begin with some definitions and notation. We assume the social network is an $n$-node graph $G = (V, E)$, representing interactions in an online system. Nodes correspond to user accounts and an edge $(u, v)$ indicates that $u$ has communicated with $v$ (again, consider the example of an email 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.

### 3.1. Description of the attack

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. Recall that as an attacker, our basic approach is to create a set of new user accounts with links among them that will "stand out" when the anonymized graph is released. Thus, we first choose a set of $k = \Theta(\log n)$ 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 $G$. We 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 $i$. (In terms of the underlying social network, 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 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 $G-H$. 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 $G$.

It is worth noting that this type of attack only involves 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.

A number of technical ingredients are needed in order to make this attack 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 $f: S \rightarrow S'$ 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 $f: S \rightarrow S$ that preserves graph's structure. An automorphism $f$ is *nontrivial* 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 $G$.
(iii) The subgraph $H$ has no nontrivial 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 following three additions. First, the size of the targeted set $W$ can be larger than $k$. The idea is that rather than connecting each $w_i$ with 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 $k = (2 + \delta) \log_2 n$, 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 $\Delta_i$ can be chosen arbitrarily, but in our experiments with the algorithm, it works well simply to choose each $\Delta_i$ 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 $\Delta_i$ 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 $\Delta_i$ 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 $N_j$.

(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 $\Delta'_i$ be the degree of $x_i$ in the full graph $G$ (this is $\Delta_i$ 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., Bollobás[5]) imply that with high probability, the graph $H$ has no nontrivial automorphisms. We will assume henceforth that this event occurs, that is, that $H$ has no nontrivial automorphisms.

We also note that the attack will work even if multiple copies of the construction are carried out simultaneously. That is, we can choose different sets of nodes to attack, $W_1, W_2, \ldots, W_t$, each of size $\Theta(\log n)$; for each $W_i$, we add a distinct set of new nodes $X_i$ to the graph $G$, building a graph $H_i$ on each $X_i$ with the different random constructions performed independently.

**Efficiently Recovering *H* Given *G*.** When the graph $G$ is released, we want to identify $H$: that is, we want to find the subset of nodes of $G$ that correspond to the set of nodes $x_1, x_2, \ldots, x_k$ of $H$. Since we have constructed $H$ to contain a path through the nodes $x_1, x_2, \ldots, x_k$, we will search along $k$-node paths in $G$, looking for a $k$-node path $P$ for which the edges induced among the nodes of $P$ have precisely the structure of $H$.

At a high level (ignoring issues of efficiency, which we discuss next), our algorithm works simply as follows. For every $k$-node path $P = \{y_1, y_2, \ldots, y_k\}$ in $G$, we visit the nodes of $P$ in order, declaring $P$ to have *failed* in the comparison to $H$ as soon as we reach a node $y_i$ that fails one of the following two tests.

(i) **A degree test**: The degree of node $y_i$ should be equal to the value $\Delta'_i$, which we know to be the degree of node $x_i$ in $G$.

(ii) **An internal structure test**: For each $j < i$, there should be an edge $(y_j, y_i)$ in $G$ if and only if $(x_j, x_i)$ is an edge of $H$.

Finally, if we reach the end of the path $P$ without any of its nodes having failed either of these tests, then by definition we have found a copy of $H$ in $G$. (As we note later, the degree test is not necessary either for the correctness of the algorithm or the bound on the worst-case running time, but it is extremely useful in practice.)

There will typically be an extremely large number of distinct $k$-node paths in $G$, so we need to organize the computation carefully in order for the search algorithm to run efficiently. We do this as follows:

- First, we loop over all nodes $v$ of $G$, trying each as the candidate starting point $y_1$ for the path $P$ (the node that will correspond to $x_1$ in $H$). If the degree of $v$ is not equal to $\Delta'_1$, then we skip $v$ in this process, since it cannot correspond to the node $x_1$ in $H$.
- For each node $v$ of degree $\Delta'_1$, in $G$, we will organize all paths originating at $y_1 = v$ into a search tree $\mathcal{T}_v$ in the natural way: each node $\alpha$ in $\mathcal{T}_v$, at depth $l$, will correspond to an $l$-node path in $G$, starting at $y_1 = v$, that has not yet failed any of the degree or internal structure tests.
- We grow $\mathcal{T}_v$ one level at a time. For each node $\alpha$ of $\mathcal{T}_v$, at depth $l$, corresponding to an $l$-node path $P = \{v = y_1, y_2, \ldots, y_l\}$ in $G$, we first check whether $y_l$ passes the degree and internal structure tests. If it does not, we declare $\alpha$ to be a leaf of $\mathcal{T}_v$. If it does pass, then we create a new child $\alpha'$ of $\alpha$ in $\mathcal{T}_v$ for each way of extending $P$ by adjoining a neighbor of $y_l$ that does not already appear on $P$.

If $\mathcal{T}_v$ ever acquires a node at depth $k$, then this corresponds to a $k$-node path in $G$ that has passed all of our tests, and hence is a copy of $H$. Conversely, if there is such a path $P$ originating at $v$, then our tree-growing procedure will continue adding nodes to $\mathcal{T}_v$ until it produces a node at depth $k$ corresponding to $P$.

Note that the total running time of this algorithm is only a small factor larger than the total number of nodes in all search trees $\mathcal{T}_v$ (summed over all nodes $v$ in $G$), and so a key issue in the analysis is to show that with high probability, the total number of nodes in all $\mathcal{T}_v$ is not too large.

## 3.2. 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 total number of nodes in all search trees $\mathcal{T}_v$ in the recovery algorithm does not grow too large.

The formal statements of these two claims are as follows.

- **Uniqueness.** *Let $k \geq (2+\delta)\log_2 n$ 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 $H = G[X]$.*

- **Efficiency.** *For every $\varepsilon > 0$, with high probability, the total number of nodes appearing in all the search trees $\mathcal{T}_v$ (over all v in G) is $O(n^{1+\varepsilon})$.*

While the proofs of these claims are somewhat involved, the basic idea underlying them is rooted in an argument formulated by Paul Erdös to prove a result in Ramsey Theory.[5, 11] In the simplest form of the argument, let us suppose we have an $n$-node graph $G$ and a random $k$-node graph $H$ on nodes $\{x_1, x_2, \ldots, x_k\}$, with each edge $(x_i, x_j)$ present independently with probability 1/2. What is the probability that $G$ contains a $k$-node subgraph that is isomorphic to $H$? For any $k$-tuple of nodes $v_1, v_2, \ldots, v_k$ in $G$, the probability that the subgraph of $G$ on this $k$-tuple is isomorphic to $H$, under the mapping sending $v_i$ to $x_i$, is precisely $2^{-\binom{k}{2}} = 2^{-k(k-1)/2}$, since the presence or absence of the random edge $(x_i, x_j)$ has to match the presence or absence of $(v_i, v_j)$ for each $(i, j)$ pair. But there are fewer than $n^k$ such $k$-tuples of nodes in $G$, and so the probability that *any* of them yields such an isomorphism is less than $n^k 2^{-k(k-1)/2}$. Now a direct calculation shows that once $k$ exceeds $2 \log_2 n$, this probability shrinks rapidly to 0, and hence it is likely that there is no isomorphic copy of $H$ in $G$.

This gives the central idea of the proofs, but the details become more complicated because the graph $H$ in the active attack is necessarily being attached by edges to the graph $G$—and this creates the possibility of isomorphisms that create a second copy of $H$ out of parts of the original $H$ together with parts of the rest of $G$. Showing that this is unlikely to happen requires a more intricate argument.

It is important to stress, however, that the intricacy of the proofs 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.

We conclude with some comments on the tests used in the recovery algorithm. Recall that as we build $\mathcal{T}_v$, 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 $H$?). The proofs of our two main claims require just the internal structure check to prove uniqueness and to bound the size of $\mathcal{T}_v$, respectively, but it is 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 values of $k$, and with much smaller search trees $\mathcal{T}_v$, 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.)

## 3.3. Computational experiments

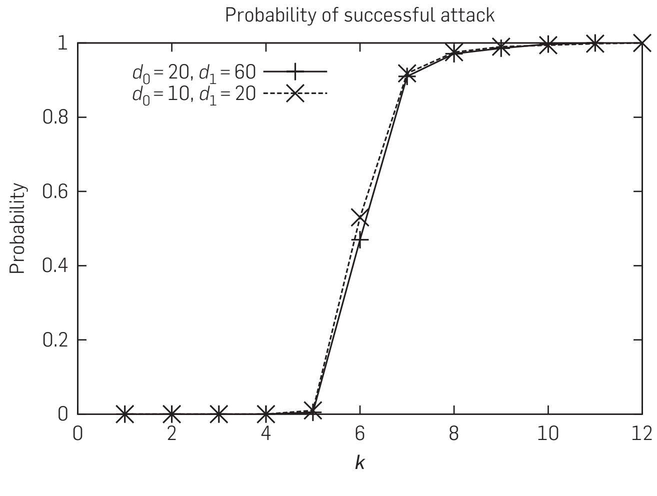
**Social Network Data**. We now describe computational experiments with the algorithm on real social network data drawn from an online setting. We find that the algorithm scales easily to several million nodes and produces efficiently findable unique subgraphs for values of $k$ significantly smaller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

As data, we use the network of friendship links on the blogging site LiveJournal, constructed from a crawl of this site performed in February 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. The LiveJournal data 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 our tests, it is not in reality an anonymous network—all 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 $\Delta_i$ 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

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



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. (Our Uniqueness result 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 $(2 + \delta) \log_2 n$ 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 total number of nodes in all search trees $\mathcal{T}_v$ would be sufficiently small that our search algorithm would be near-linear. In our experiments on the LiveJournal friendship graph, we find that, in practice, the total number of nodes in all $\mathcal{T}_v$ is not much larger than the number of nodes $v$ whose degree in $G$ is equal to $\Delta_1'$. (Recall that we only build search trees for those $v$ that have this degree.) For instance, when $d_0 = 10$ and $d_1 = 20$, there are an average of 70,000 nodes that have degree $\Delta_1'$, while the total number of nodes in all search trees $\mathcal{T}_v$ is typically about 90,000.

**Detectability**. Our simple attack shows that simple anonymization does not preserve privacy of links. One might wonder about the *detectability* of the attack: can the curator of the data, who is releasing the anonymized version, not be able to discover and remove $H$? The curator does not have access to the secret degree sequence or the edges within $H$ and so 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 subtle issue is worthy of more rigorous treatment; here, 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 $H$ as an induced 7-node subgraph. (We are still able to find $H$ because of the pattern of edges that connect nodes of $H$ to nodes of $G - H$.) 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 $G$.

## 4. 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 $k$: 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 nontrivial automorphisms.[5] In what follows, we will assume that all these properties hold: $\gamma(H) = \delta(H) \geq k/3 > b$, and $H$ has no nontrivial 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 $G$. Thus, $b$ of the nodes of $H$ each have a single edge to a node of $G - H$, while the other $k - b$ nodes of $H$ have no edges to nodes of $G - H$.

A crucial property of $H$ that we will use is the following: there are $b$ edges in total that have one end in $H$ and the other end in $G - H$; on the other hand, each node in $H$ has more than $b$ edges to other nodes of $H$.

Finally, we note that as with the walk-based attack in the previous section, we can also carry out multiple copies of the present construction simultaneously, if desired, so as to attack multiple sets of targeted users $W_1, W_2, \ldots, W_t$.

**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 $G$—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 $T$.[13]

Computing $T$ is the most expensive step of the recovery algorithm, computationally. The best running time known for constructing a Gomory–Hu tree in a graph with $n$ nodes and $m$ edges is $O(mn)$ times a factor that is polynomial in $\log(m + n)$.[3] This is a much larger worst-case bound than we have for the walk-based attack. On the other hand, computational experiments in Web graph analysis indicate that Gomory–Hu tree computations can in fact be made to scale to very large graphs in practice.[12]

(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 $k! = o(n)$. By adapting our proof of Uniqueness from the walk-based attack, we can show a form of uniqueness for $H$ here too:

- *With high probability, there will be a single i such that G[Si] 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.

**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 0.99. 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 nontrivial 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 nontrivial 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 notation from the description of the attack above, there are at most 8333333 possible components $S_j$ of size 12 in the forest $T'$, one of which, say $S^*$, is our subgraph $H$. The probability that there even *exists* an $S_j \neq S^*$ that is isomorphic to $H$ is bounded by $8333333 \cdot 12! < 2^{-66} < 6 \cdot 10^{-5}$. Hence the probability that *any* $H_i$ will lead to non-uniqueness when attached to $G$ is at most 0.006, and so in particular this holds for the $H_i$ that we choose as $H$.

By way of comparison, the provable bounds for the walk-based attack require a number of new user accounts that is at least $2 \log_2 n$, which is approximately 53 when $n$ is 100 million. On the other hand, as we have seen in our computational experiments, the walk-based attack appears to require fewer nodes in practice than the provable guarantees suggest, suggesting that further empirical comparison of these two attacks would be an interesting open question.

## 5. 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.

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 among themselves—the internal structure of $H = G[X]$, using the terminology from the active attacks. 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.

The attack itself is analogous to the walk-based attack, except that the structure of $H$ arises organically from the behavior 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 which edges $(x_i, x_j)$ are in $G$ and also the neighbors of each $x_i$ in $G-X$. Once $G$ is released, the coalition runs the search algorithm from 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 used for the walk-based active experiments. For each nonempty set $S \subseteq \{1, 2, \ldots, k\}$, let $g(S)$ denote the number of nodes in $G$ that have edges to all the element of $\{x_i: i \in S\}$ and none of the elements of $\{x_i: i \notin S\}$. (In some places, we will abuse the notation for $g(\cdot)$ as follows: if $U$ is a set of nodes in $X$ rather than a set of indices, we will use $g(U)$ to denote the number of nodes in $G$ that have edges to all elements of $U$ and no elements of $X-U$.) Now, suppose we have a node $a$ in a search tree $\mathcal{T}_v$, corresponding to a path $y_1, y_2, \ldots, y_l$ in $G$. For each $S \subseteq \{1, 2, \ldots, l\}$, it should be the case that exactly $g(S)$

nodes of $G$ are connected to all members of $\{y_i: i \in S\}$ and none of $\{y_i: i \notin S\}$; otherwise, $\{y_1, \ldots, y_l\}$ cannot be the first $l$ nodes of the copy of $H$ in $G$.

Finally, once the coalition of users $X$ finds itself, it can determine the identity of any user $w \notin X$ whose neighbor set $S$ in $X$ satisfies $g(S) = 1$. (In this case, $w$ is uniquely identified by the identities of its neighbors in $X$.)

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.

**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 analyze the attack described above for a randomly chosen sample of users $x_1$ whose degree is at least $k-1$.

We find that even coalitions as small as three or four 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 based on both the "simple" algorithm using the internal structure of $H$ and the degree sequence, as well as the "refined" algorithm that incorporates the function $g(S)$. 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 so that $H$ would be 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; this makes it easier to find the resulting subgraph $H$. 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.

As Figure 3 shows, the passive attack identifies relatively few nodes outside the coalition, compared to the active attack. However, with a semi-passive attack, we can greatly increase the number of users compromised, as indicated by Figure 3 (and recall that these users can be chosen arbitrarily

**Figure 2. Probability of success for different coalition sizes in the LiveJournal graph, comparing a simple algorithm using only the degrees and internal structure of the coalition, and a more refined algorithm using the edges connecting *H* to *G–H*.**

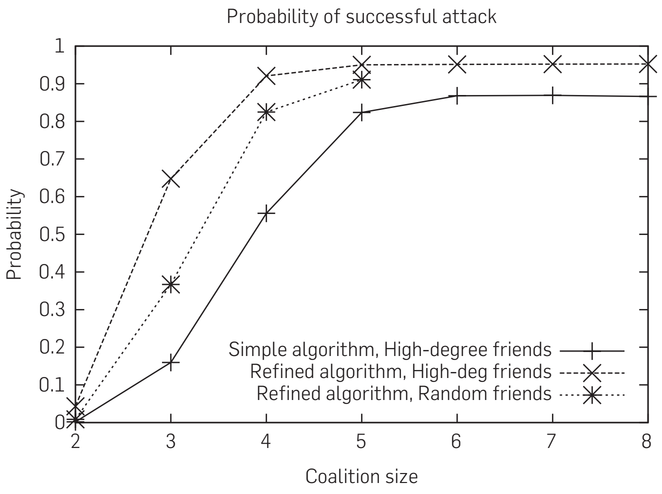
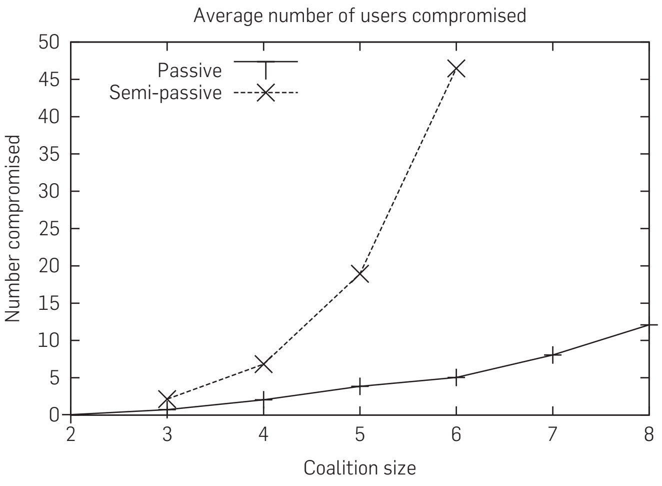


**Figure 3. As the size of the coalition increases, the number of users in the LiveJournal graph compromised under the passive attack when the coalition successfully finds itself increases superlinearly. The number of users the semi-passive attack compromises increases exponentially.**



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.

## 6. DISCUSSION

It is natural to ask what conclusions about private analysis of 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 online 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 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 these questions are not something 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 might it have access, now or in the future.

There is a growing literature to which we can turn for thinking about ensuring privacy in settings such as these. There has been extensive recent work on privacy-preserving data mining, beginning with Agrawal et al., Samarati, and Sweeney[1,2,23,24] which rekindled interest in a field quiescent since the 1980s, and increasingly incorporating approaches from modern cryptography for describing and reasoning about information leakage.[4,7,10,18] The notion of e-differential privacy gives very strong guarantees, independent of the auxiliary information and computational powers of the adversary (see Dwork et al.[8,9,10]). 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 simple and general *interactive* mechanism for ensuring differential privacy is given in Dwork et al.[10] In this 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. The advantage of interaction lies in the fact that accuracy must deteriorate with the number and complexity of questions asked (see Dinur and Nissim,[7] *et sequelae*). In a noninteractive solution, the curator must produce an object that answers all *potential* future questions; interactive approaches answer only those questions actually asked.

A lively literature (see, e.g., Hardt and Rothblum[14] and the references therein) explores the tradeoffs between accuracy, computation, and degree of differential privacy in answering very large numbers of *counting queries*, that is, questions of the form "How many people in the database satisfy property *P*?" In the context of a social network in which the goal is to protect the privacy of individual friendships, this captures questions of the form "How many edges (friendships) connect people with property *P* to people with property *Q*?" such as, "How many friendships are there between people who went to Princeton High School and Cornell graduates?"

The only privacy definition of which we are aware that protects against arbitrary auxiliary information is differential privacy. Further progress on differentially private analysis of social networks awaits compelling and precise analytical goals.

## Acknowledgments

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. C

**References**

1. Agrawal, D., Aggarwal, C. On the design and quantification of privacy preserving data mining algorithms. In *ACM Symposium on Principles of Database System* (2001).
2. Agrawal, R., Srikant, R. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD (*2000).
3. Bhalgat, A., Hariharan, R., Kavitha, T., Panigrahi, D. An $\tilde{O}(mn)$ Gomory-Hutree construction algorithm for unweighted graphs. In *Proceedings of ACM Symposium on Theory of Computing* (2007).
4. Blum, A., Dwork, C., McSherry, F., Nissim, K. Practical privacy: the SuLQ framework. In *ACM PODS* (2005).
5. Bollobás, B. *Random Graphs.* Cambridge University Press, Cambridge, U.K., 2001.
6. Crandall, D., Backstrom, L., Cosley, D., Suri, S., Huttenlocher, D., Kleinberg, J. Inferring social ties from geographic coincidences. *Proc. Natl. Acad. Sci., 107* (2010).
7. Dinur, I., Nissim, K. Revealing information while preserving privacy. In *Symposium on Principles of Database System* (2003).
8. Dwork, C. Differential privacy. *Proceedings of International Colloquium on Automata, Languages and Programming* (2006).
9. Dwork, C. A Firm Foundation for Private Data Analysis. *CACM 54,* 1 (2011).
10. Dwork, C., McSherry, F., Nissim, K., Smith, A. Calibrating noise to sensitivity in private data analysis. In *Proceedings of Theory of Cryptography Conference* (2006).
11. Erdös, P. Some remarks on the theory of graphs. *Bull. AMS 53* (1947).
12. Flake, G., Tarjan, R., Tsioutsiouliklis, K. Graph clustering and min cut trees. *Internet Math. 1* (2003).
13. Gomory, R., Hu, T.C. Multi-terminal network flows. *J. Soc. Ind. Appl. Math. 9* (1961).
14. Hardt, M., Rothblum, G. A multiplicative weights mechanism for privacy-preserving data analysis. In *Proceedings of FOGS* (2010).
15. Hay, M., Miklau, G., Jensen, D., Towsley, D., Weis, P. Resisting structural re-identification in anonymized social networks. In *Proceedings of the VLDB Endowment*, 1 (2008).
16. Jernigan, C., Mistree, B. Gaydar: Facebook friendships expose sexual orientation. *First Monday 14* (2009).
17. Kumar, R., Novak, J., Pang, B., Tomkins, A. On anonymizing query logs via token-based hashing. In *Proceedings of the 16th International World Wide Web Conference* (2007).
18. Mishra, N., Sandier, M. Privacy via pseudorandom sketches. In *ACM Symposium on Principles of Database System* (2006).
19. Mislove, A., Viswanath, B., Gummadi, P.K., Druschel, P. You are who you know: inferring user profiles in online social networks. In *ACM WSDM* (2010).
20. Narayanan, A., Shmatikov, V. Robust de-anonymization of large sparse datasets (How to break anonymity of the Netflix prize dataset). In *Proceedings of the IEEE Symposium on Security and Privacy* (2008).
21. Narayanan, A., Shmatikov, V. De-anonymizing social networks. In *Proceedings of the IEEE Symposium on Security and Privacy* (2009).
22. Novak, J., Raghavan, P., Tomkins, A. Anti-aliasing on the web. *Proceedings of the 13th International World Wide Web Conference* (2004).
23. Samarati, P. Protecting respondents' identities in microdata release. *IEEE TKDE 13* (2001).
24. Sweeney, L., k-anonymity: a model for protecting privacy. *Intl. J. Uncertainty Fuzziness Knowledge-Based Systems 10* (2002).
25. Zheleva, E., Getoor, L. The illusion of privacy in social networks with mixed public and private user profiles. *Proceedings of the 18th International World Wide Web Conference* (2009).

**Lars Backstrom** (lars@fb.com), Facebook, Palo Alto, CA.

**Cynthia Dwork** (dwork@microsoft.com), Microsoft Research, Silicon Valley Campus, Mountain View, CA.

**Jon Kleinberg** (kleinber@cs.cornell.edu), Cornell University, Ithaca, NY.