Detecting Social Cliques for Automated Privacy Control in Online Social Networks

Hakan Yıldız
University of California, Santa Barbara, USA
hakan@cs.ucsb.edu

Christopher Kruegel
University of California, Santa Barbara, USA
chris@cs.ucsb.edu

***Abstract*—As a result of the increasing popularity of online social networking sites, millions of people spend a considerable portion of their social life on the Internet. The information exchanged in this context has obvious privacy risks. Interestingly, concerns of social network users about these risks are related not only to adversarial activities but also to users they are directly connected to (friends). In particular, many users want to occasionally hide portions of their information from certain groups of their friends.**

To satisfy their users' needs, social networking sites have introduced privacy mechanisms (such as Facebook's friend lists) that enable users to expose a particular piece of their information only to a subset of their friends. Unfortunately, friend lists need to be specified manually. As a result, users frequently do not use these mechanisms, either due to a lack of concern about privacy, but more often due to the large amount of time required for the necessary setup and management.

In this paper, we propose a privacy control approach that addresses this problem by automatically detecting social cliques among the friends of a user. In our context, a social clique is a group of people whose members share a significant level of social connections, possibly due to common interests (hobbies) or a common location. To find cliques, we present an algorithm that, given a small number of friends (seed), uses the structure of the social graph to generate an approximate clique that contains this seed. The cliques found by the algorithm can be transformed directly into friend lists, making sure that a piece of sensitive data is exposed only to the members of a particular clique. Our evaluation on the Facebook platform shows that our method delivers good results, and the cliques that our algorithm identifies typically cover a large fraction of the actual social cliques.

I. Introduction

Over the past few years, online social networking sites have experienced a massive growth, and they have become environments where hundreds of millions of people interact with each other. The information exchanged on these sites is not only of enormous amounts but also of many different kinds. Most popular sites enable their users to exchange information in the forms of profiles, messages, wall posts, photographs, videos, and group activities.

The large amount of diverse information poses a considerable privacy threat to users. The lack of protection of the private user information may result in adversarial activities such as stalking, identity theft, and blackmail [1], [2]. To mitigate these privacy risks, most sites implement some sort of privacy policy that exposes user information only to the user's friends. Moreover, some sites, such as Facebook, allow their users to configure the visibility of their data even further, making it accessible to friends of friends or everyone.

In many situations, however, the privacy options that treat all friends of a user equally do not suffice (this is not surprising, considering that the average Facebook user has 130 friends [3]). In fact, it is quite common that a user might want to share a particular piece of information only with certain people and hide it from others. To deal with this demand, Facebook introduced *friend lists*. Friend lists allow one to manage the privacy (visibility) of certain objects, enabling users to expose a piece of their data to a subset of their friends only. (Recently, Facebook also launched a similar but a less specific mechanism called *groups*.) Note that while our discussion focuses mostly on Facebook, similar problems and privacy solutions such as friend lists exist also in other online social networks. Thus, in the following text, we refer to the set of friends who are allowed to see a piece of information more generally as *exposure set*.

Given mechanisms such as friend lists and groups, users have significant control over their privacy since they can specify any subset of their friends that they want to share a particular piece of information with (the exposure set). However, the effectiveness of these mechanisms in eliminating the privacy risks is subject to debate. Users, on average, have a large number of friends, often hundreds, and sometimes even thousands. This makes it time-consuming and tedious to manually create friend lists (or groups) and associate them with private data. As a result, many Facebook users do not effectively use these mechanisms. Even Facebook's founder Mark Zuckerberg, when asked about friend lists, pointed out that while they would be the "the ideal solution for sharing different things with different people," they have unfortunately failed because "nobody wants to make lists" [4].

Recently, Facebook introduced a new mechanism that generates pre-defined friend lists that classify the friends of a user based on the profile data they provide (e.g., location, school, family information). The aim of this scheme is to save the user from the hassle of creating lists. This mechanism, however, has still some disadvantages. First, when the user shares a piece of information, she still has to assign a friend list manually. Second, the list generation is dependent on the data provided by the user. In the case where the user does not prefer to disclose this data, this scheme fails.

One solution to the aforementioned problems is to automate

both the generation and the assignment of the exposure sets (friend lists) *during the sharing of information*. In particular, we imagine a privacy policy in which the exposure set for a piece of data is generated and associated with it automatically, as soon as the data is put on the social network. The exposure set is generated in such a way that it includes only the users who are likely to be related to the data being shared, in contrast to being confined to a pre-defined list. If implemented properly, such an automated scheme guarantees that every piece of data is seen only by the people related to it. Moreover, it functions as a privacy defense against friend impersonators, as they will need to establish considerable social relationship before being granted access.

At this point, it should be noted that it is practically impossible to exactly generate the ideal exposure set that is desired by the user without getting any directives from her. However, if an accurate approximation for this ideal set is generated, it still provides an effective privacy protection in the sense that most of the people excluded by the ideal set are also excluded by its approximation. Furthermore, the generated exposure set can be displayed to the user to receive feedback. If the user is not happy with the result, she can update it accordingly. Notice that correcting a (good) approximation is much less time consuming than creating an exposure set from scratch.

Now, the following question arises. Given a piece of data shared on a social network, how can we determine an accurate exposure set for it? One way to answer this question is to use the idea of social cliques. A *social clique* is a group of people having significant social interaction with each other due to a particular cause (e.g., families, classmates, colleagues). A piece of data is often of concern only to a particular social clique. One can attempt to identify this clique by examining the data itself. Most data shared on social networking sites contains some (meta) information that identifies the users who are directly related to or who contribute to this data. Thus, each piece of data is associated with a (possibly empty) group of users. We call this group the *participating group* of the data. Intuitively, in most cases, the social clique concerned with a piece of data is the one that contains its participating group. As an example, consider a family photograph being shared on a social network. There are a number of family members who appear in it, and these members form the participating group of the photograph. This photograph is most likely of concern only to the members of the family, where the family is a social clique that contains the participating group (those members that appear in the picture).

In this paper, we present a novel approach for detecting social cliques among the friends of a user in a social network. Specifically, given a user and a subset S of her friends, we determine a (typically larger) subset S' of the user's friends so that S' forms a social clique that contains S. We describe an algorithm to solve this problem, whose behavior can be changed according to the parameters provided by the user. We then evaluate our approach through experiments performed on the Facebook platform. Note that our techniques can used to automatically set up friend lists for shared data via a plug-in for a social network.

Once social cliques (or communities) have been identified and translated into friend lists, the user has to decide which information she wants to share with each community. This problem is outside the scope of this work. Interestingly, a previous paper [5] has proposed "privacy wizards" that can help to automatically configure privacy settings for different friend lists. Our work, which proposes a novel community detection system, can be used as an input to the privacy wizard system, which currently leverages a clique detection algorithm based on the idea of edge betweenness [6].

II. Related Work

In this section, we briefly review past research related to privacy and community detection in social networks.

A. Privacy in Online Social Networks

The increasing popularity of online social networks has prompted the interest of security researchers to explore the possible adversarial activities that target these networks. Substantial research has been done on privacy in social networks, revealing that the information stored in these networks can be exploited through different types of attacks, including identity theft and spam (e.g., [2], [7]). A fair portion of research covers techniques that aim to hide user information from third parties and central servers, using cryptography (e.g., NOYB [8]) and network decentralization (e.g., [9]). These mitigation techniques, however, are not effective against social engineering type of attacks in which users are tricked into trusting entities controlled by adversaries. Scientists introduced some techniques against this type of attack (e.g., SybilGuard [10]), which work by analyzing the social network structure to assess the trust of an entity. However, recent research [11] shows that the existence of effective defenses against online social engineering attacks is still debatable.

B. Community Detection in Social Networks

Community detection is the identification of densely-connected groups of nodes in networks. It is applicable to many different kinds of networks, such as social, biological, and computer networks. Even before the community detection methods started to be explored, there were efforts to establish formal definitions for communities in social networks. Concepts such as K-clique [12], K-club [13], and K-plex [14] were introduced as a result of these efforts. The first community detection attempt was by Small and Griffith, who grouped the documents in the Science Citation Index based on a co-citation graph [15]. Since then, there has been considerable research on community detection [16]. Some studies focused on a specialized community detection problem known as local community detection (e.g., [17], [18], [19]). In this problem, one aims to detect the community that a particular node belongs by accessing only local graph information. The evaluation of community detection algorithms has used mainly two classes of metrics. One class of metrics is

based on maximizing or minimizing formulas that describe the structural properties of the graphs (e.g., modularity [6]). The other class of metrics is based on ground truth communities that are inferred from the dataset and compared to the results. For example, for a community detection problem on academic co-authorship graphs, publication venues (such as conferences and journals) may serve as ground truth communities (an idea used in [20]).

C. Our Contributions

Our work contributes to both social network privacy research and community detection research. The majority of research on social network privacy is concerned with protecting sensitive user information against adversaries. Our work takes a different angle; it focuses on keeping particular data items private from the friends of a user. From this point of view, we show that it is possible to perform accurate privacy protection with minimal user effort.

The algorithm that we propose for finding cliques among friends roughly falls into the class of local community detection techniques. We present new clique relaxation schemes for this algorithm as an alternative to similar schemes such as K-clique, K-club, and K-plex. Our approach yields better results than previous techniques. Finally, to the best of our knowledge, we are the first to use real-world photo tags on an actual social network (Facebook) to evaluate the performance of community detection algorithms.

III. Approach

Consider a piece of data shared in a social network. We call the user who hosts (publishes) the data in her account the *host user*. The goal of our system is to protect the privacy of the host user. To this end, we want to find the subset of the host user's friends who should have access to this data.

The host user's friends are collectively called *candidates*, since they are candidates for the exposure set for the piece of data. The users who are members of the participating group related to the data (e.g., people in a picture, or users who posted to a discussion thread) are called the *participants*. We assume that all participants are friends of (connected to) the host user. This implies that all participants are also candidates (since they are friends with the host user). If this is not the case, one can easily handle this problem by introducing virtual friendship links between the host user and the participants. Note that these assumptions do not limit the generality of our algorithm. They merely make the subsequent notations and descriptions simpler.

Our goal is to approximately determine which candidates form a social clique with the participants. Users that are part of this clique[1] should then be granted access to the data (even though they have not yet contributed to it).

The approach we propose is based on the structure of the social graph where the nodes represent the individuals (users), and the edges represent the friendship relationships (connections) between them. Note that the relationships due to user activities (e.g., being tagged in the same photo) are ignored. We choose to use only the friendship information because it is easier to analyze and all social networks provide this minimal information.

[1]Throughout the paper, the term "clique" refers to a social clique and is not used in the graph theoretical sense.

procedure $DetectClique(G, s, P, f)$
  $C \leftarrow P$
  while $true$
    $t \leftarrow null$
    for each $v \in (n_G(s) - C)$
      if $f(G, C, v) = true$
        if $t = null$ **or** $h(G, C, v) > h(G, C, t)$
          $t \leftarrow v$
    if $t = null$
      break
    else
      $C \leftarrow C \cup \{t\}$
  return C

Fig. 1. The pseudocode for the clique detection algorithm.

Our clique-finding algorithm works on the *local social graph*. The local social graph is a subgraph of the entire social graph of the network, and it includes (a) the host user, (b) the friends of the host user (i.e., the candidates), and (c) the friends of all candidates. That is, all users (nodes) that are at most two degrees away from the host user are included. An advantage of using the local social graph is that the graph size is very small compared to the entire social network, thus, the computation can be carried out efficiently.

A. Algorithm

We now describe an algorithm that is applied to the local social graph to generate a clique from a set of participants. The main idea is to iteratively grow the clique, starting from the set of participants. This type of agglomerative algorithms are often used in local community detection (e.g., [17], [18], [19]). The pseudocode of the algorithm is provided in *Figure 1*. The inputs are the social graph G, the host user s (as a node in G), the set of participants P, and a predicate function f called the clique expansion function. We use the notation $n_G(v)$ to denote the set of the friends of v. Thus, $n_G(s)$ denotes the candidates.

The algorithm starts by creating an initial clique C that consists of only the participants. Then, in each iteration of the main loop, this clique is expanded by inserting one of the candidates into it. The inserted candidate is the one who maximizes the heuristic function h, selected among all candidates that satisfy the clique expansion function f. The growth of the clique stops when no candidate outside the clique can be found that satisfies f (We discuss f in the next section). The final clique is returned as the output of the procedure.

In each iteration, only one candidate is added to the intermediate clique, even though there may be many candidates who

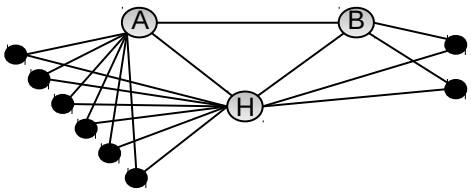

Fig. 2. A real social graph of three Facebook users (grey) and their common friends (black). Only the edges of the grey nodes are shown. A and H seem to be closer friends.

satisfy f. Thus, one needs an extra measure that determines which candidate is best used to expand the clique. The heuristic function h serves this purpose. The candidate with the highest heuristic value is considered to have the highest level of social relationship to the clique and is, therefore, used to expand the clique. The heuristic function is defined as

$$h(G,C,v) = \begin{array}{c} |V_G| \times |C \cap n_G(v)| \\ + \\ \sum_{c \in C} |n_G(c) \cap n_G(v)|/|C| \end{array}$$

where V_G is the set of users in graph G. h is based on two measures. The primary measure is the number of friends of v that are in C. This is captured by the first term. The factor $|V_G|$ in the term ensures that this term is the dominant measure. The number of friends of v in C is occassionally insufficient to differentiate between the candidates. Especially for small clique sizes, there can be many candidates with equal number of friends in C. To resolve such situations, we utilize a secondary measure, represented by the second term, that evaluates to the average number of common friends between the users in the clique C and v. Notice that we use the *number of common friends* between two individuals to "estimate" or "guess" the strength of their relationship. It turns out that real social graphs have many instances where one can differentiate the level of relationships by looking at the number of common friends (see Figure 2 for a simple example).

Note that, in general, a high number of common friends does not necessarily indicate a close friendship, and there may be cases where our heuristic measure is misleading. However, we believe such cases are in minority and emphasize the fact that the common friends measure uses only very local information, and is, therefore, very appropriate for our local clique detection algorithm.

B. Clique Expansion Schemes

The clique expansion function f is an important parameter of the algorithm that affects how a clique grows and that determines when it stops growing. In this section, we present three clique expansion schemes.

1) CLQ Expansion Scheme.: We first introduce an expansion scheme that will serve as the baseline for the evaluation of the two other schemes. In the CLQ scheme, we require that every person in a social clique is a friend of each other. To fulfill this requirement, we formally define f as

$$f(G,C,v) = \begin{cases} true & \text{if } n_G(v) \cap C = C \\ false & \text{otherwise} \end{cases}$$

When the CLQ scheme is used, the output of the algorithm is a complete subgraph.

2) BAND$_K$ Expansion Scheme.: Complete subgraphs are often considered to be too strict to represent social cliques, as cliques display looser connections in reality. Considerable previous work has focused on finding different definitions for cliques. The efforts yielded to clique definitions such as K-clique [12], K-club [13], and K-plex [14]. Most of these definitions, however, are inappropriate to find cliques among the friends of a user and, instead, are designed to operate on the entire graph. To address the limitation of existing approaches, we introduce a new clique relaxation scheme called *K-band*. A K-band is a social group such that every person in the group has at least K common friends with each of the other persons in the group. The reader will notice that the number of common friends is again used as a measure of the social relationship between two persons.

The BAND$_K$ scheme requires that the intermediate clique is always a K-band during the execution of the algorithm. Accordingly, we define f formally as:

$$f(G,C,v) = \begin{cases} true & \text{if } \forall c \in C. |n_G(c) \cap n_G(v)| \geq K \\ false & \text{otherwise} \end{cases}$$

Notice that the case $K = 1$ does not make sense for our algorithm because every candidate has at least one common friend with every member of the intermediate clique, namely the host user. So, meaningful values for K are integers greater than 1.

3) IN$_K$ Expansion Scheme.: The main idea of the IN$_K$ scheme is to adapt the expansion function to the current tightness of an intermediate clique. In particular, if the initial clique formed by the participant set is loose, we want to grow it into a loose and large final clique. On the other hand, for a tight initial clique, the aim is to grow into a tight and small final clique. This behavior helps to detect and adapt to social cliques of variable tightness.

As part of IN$_K$ scheme, we first define a function r_G, which serves as a measure of the tightness of the relationship between an individual v and a clique C. r_G is defined as:

$$r_G(v,C) = \frac{|n_G(v) \cap C|}{|C|}$$

Basically, r_G is the fraction of v's friends among all individuals in C. We reasonably assume that the higher this ratio is, the more links v has to C, making it socially closer to C.

We now define a function t_G that measures the tightness within a clique. For this, we simply average the r_G values of each individual in the clique with respect to the remaining members of this clique. This yields to the following definition

for t_G:

$$t_G(C) = \begin{cases} \sum_{c \in C} \frac{r_G(c, C - \{c\})}{|C|} & \text{if } |C| > 1 \\ 0 & \text{otherwise} \end{cases}$$

where C is the clique. The second line of the definition handles the case in which C consists of a single user. In this case, we define $t_G(C)$ as 0.

In the IN_K scheme, a candidate is allowed to expand an intermediate clique if her tightness to it is larger than K times the tightness within the clique. This is captured by the following definition of f:

$$f(G, C, v) = \begin{cases} true & \text{if } r_G(v) > K \times t_G(C) \\ false & \text{otherwise} \end{cases}$$

C. Practical Usage

The clique detection algorithm that we have discussed, when used with an appropriate clique expansion scheme, provides a way to perform automated privacy control for social networking sites. Whenever a user uploads a piece of data that has a participating group, our algorithm can be run to automatically determine an exposure set for it. Moreover, the site may provide different levels of privacy protection, which can be implemented by adjusting the parameters for the clique expansion schemes.

IV. Evaluation

In this section, we present the results of our evaluation of the quality of the cliques that our system produces. To test our approach in a real social network and to evaluate its performance, we developed a Facebook application named "AutoClique.". 30 Facebook users volunteered to install AutoClique. We accessed the friend lists of these users and their friends (via crawling), and also gathered data to conduct our experiments.

1) Experimental Methodology: For experiments, we made use of the pictures that users have uploaded. More precisely, we extracted the tags associated with each picture that was uploaded by our 30 users. This allowed us to identify the set of (tagged) people that appear on each picture.

Now, the basic assumption for our experiments is that all people in a picture belong to the same social clique. We believe that this assumption holds for majority of photos in Facebook. It is important to observe that this assumption is not used at all by the algorithm to detect cliques; it is merely used for the evaluation of the results of the system. Moreover, as discussed later, the cases in which the assumption does not hold only make our final results appear worse than they would be if ground truth was actually available.

Leveraging on the assumption that we can use the tags on pictures as ground truth, we perform a single test on a photo in the following manner: The photo has a set of users associated with it through photo tags. We call this set T. Under the assumption that T represents a real social clique, or a part of it, we randomly select a subset P of T as participants. We then execute our algorithm to generate a social clique C from P. Finally, we compare C to T to see how many of the remaining people in T (those not selected for P) are covered by C.

It is important to observe that, in most cases, the set T of people in a photo do not constitute a complete clique. Instead, the sizes of typical cliques of friends are in the order of tens of people (consider a circle of friends in the local sport club, relatives, friends in school, ...). Thus, when our algorithm produces a clique C from the subset P of people on a photo, we expect that this clique C covers more people than shown on the photo. The key point that we attempt to evaluate is whether the remaining people on the photo ($T - P$) are part of our inferred clique C. When this is the case, we consider the output of the algorithm as a success.

Of course, we would hope that our algorithm finds a social clique that covers all users in T. To quantify the results of a single test, we use the two metrics *recall* and *coverage*. *Recall* is defined as:

$$\frac{|T \cap C| - |P|}{|T| - |P|}$$

It captures the fraction of people on the photo (T) that is covered by our clique C. Notice that the participant set P is excluded from this ratio since it is guaranteed to be in C. The second metric is *coverage*, which is defined as $|C|/n$ where n is the number of friends the host user has. It stands for the ratio of the host user's friends covered by C.

A good clique detection algorithm has a high recall but a low coverage. We motivate this with the following reasoning: A high recall (optimally, 1) indicates that most (all) of the users in the real clique are included in the clique that our algorithm detected. This is essential to make sure that the majority of the users concerned with a piece of data, such as a photo, are included in its exposure set.

Of course, one can easily achieve 100% recall by simply generating a clique that contains all of the host user's friends. However, such a clique includes many individuals who are not concerned with the data, and thus, our system would not succeed in protecting privacy. Generally, a piece of data only concerns a small portion of the host user's friends. Consequently, to exclude irrelevant users, the detected clique should cover only a small portion of the host user's friends. This is indicated by a low coverage of the user's entire set of friends. That is, a low coverage indicates that cliques are reasonably tight and meaningful.

At this point, it is worth revisiting the assumption that individuals tagged in the same photograph really form a social clique (or a part of it). Clearly, this assumption is not always true. It is possible that a photograph contains people who have weak social relationships or no social relationships at all. However, we believe that such instances are not too frequent. Moreover, such cases make the performance of our algorithm appear *worse*. The reason is that our algorithm might produce an accurate clique that correctly excludes a "random" person that happens to be in the picture. However, the recall score is lowered, because the basic assumption for the test is that all people on the photo should be in the same clique.

TABLE I
THE NUMBER OF PHOTOS CONTAINING SPECIFIC NUMBER OF TAGS.

Number of Tags	2	3	4	5	6	7	8	9+
Number of Photos	634	337	157	105	63	43	24	55

2) Test Data: AutoClique crawled the local social graphs of the 30 users who installed the application. During this process, we attempted to collect the sets of friends for $8,596$ users (host users and their friends). The average number of friends per user was 338. The user with the highest number of friends had $5,002$ connections. We then gathered information about the photos (and the photo tags) of our users. More precisely, we gathered $1,416$ photos from 22 of the 30 users (the rest had no photos), together with the tags in each photo. *Table I* shows the distribution of the photos with respect to their number of tags. Note that we ignored the photos with only a single tag as we need at least two tags for evaluation. Of course, users were clearly informed about the data that we collected, and they gave their consent when installing our application.

3) Experimental Results: In our tests, we examined seven schemes: **CLQ, BAND$_2$, BAND$_3$, BAND$_4$, IN$_{0.3}$, IN$_{0.5}$, IN$_{0.7}$**. Additionally, to compare our algorithm's accuracy, we examined the following two local community detection algorithms:

- **CLA**: Clauset's greedy algorithm with the stopping condition of decreasing local modularity [17]. According to Chen et al. [19], this algorithm outperforms most local community detection algorithms.
- **CZG**: Chen, Zaïane, and Goebel's algorithm with the discovery and examination phases [19].

To restrict the detected cliques to the friends of the host user (and for efficiency reasons), we applied the CLA and the CZG algorithms to the subgraphs induced by the friends of each host user.

The test run for a single scheme on the data set of $1,416$ photos was done in the following manner: For every photo X and for each positive integer Y less than or equal to 8 (but also less than the number of tags in X), we ran ten tests. A single test was performed as described previously under "Methodology," and we used a different, random participant group of size Y for each of the ten tests per photo. This resulted in a total number of $33,860$ tests for each scheme.

Table II shows the (averaged) results for these test runs. The CLQ scheme has the lowest average recall, as expected, since the resulting cliques are too tight (observe the average clique size). The other schemes achieve better average recall values that change according to their parameter. For example, we see that BAND$_2$ has a recall of 0.90 and a coverage of 0.28 on average. Overall, it can be observed that our approach is capable of detecting social cliques with approximately 90% recall and 30% coverage, using both BAND$_K$ and IN$_K$ schemes with appropriate parameters. Notice the high recall and the low coverage values that indicate that our algorithm is appropriate for effective (though not perfect) privacy protection among friends. Additionally, it is easily observed that our schemes significantly beat CLA and CZG.

TABLE II
AVERAGE RECALL, COVERAGE AND DETECTED CLIQUE SIZE FOR SEVERAL SCHEMES ALONG WITH THEIR STANDARD DEVIATIONS. THE RESULTS FOR EACH SCHEME ARE OBTAINED THROUGH 33,860 TESTS WITH VARIABLE PARTICIPANT GROUP SIZES (1 TO 8) OVER A SET OF 1,416 PHOTOS. THE AVERAGE NUMBER OF TAGS IN THE PHOTO IS 5.4 PER TEST. THE AVERAGE SIZE OF THE PARTICIPANT GROUP IS 2.4 PER TEST.

Scheme	Avg. Recall	Avg. Coverage	Avg. Clique Size
CLQ	0.49 ± 0.41	0.05 ± 0.04	11 ± 4
BAND$_2$	0.90 ± 0.27	0.28 ± 0.19	65 ± 38
BAND$_3$	0.88 ± 0.30	0.25 ± 0.19	54 ± 33
BAND$_4$	0.85 ± 0.33	0.23 ± 0.18	48 ± 31
IN$_{0.3}$	0.92 ± 0.24	0.36 ± 0.20	97 ± 71
IN$_{0.5}$	0.86 ± 0.31	0.23 ± 0.16	53 ± 35
IN$_{0.7}$	0.74 ± 0.37	0.14 ± 0.11	30 ± 17
CLA	0.75 ± 0.40	0.28 ± 0.20	71 ± 59
CZG	0.57 ± 0.43	0.17 ± 0.16	36 ± 30

TABLE III
AVERAGE RECALL FOR FOUR SCHEMES AND DIFFERENT NUMBER OF PARTICIPANTS. THIS TABLE IS BASED ON A DATASET OF 55 PHOTOS WHICH HAVE AT LEAST 9 TAGS.

Number of participants	CLQ	BAND$_2$	IN$_{0.3}$	CLA	CZG
1	0.29	0.74	0.79	0.53	0.39
2	0.31	0.76	0.81	0.74	0.56
3	0.31	0.74	0.81	0.79	0.60
4	0.28	0.74	0.81	0.79	0.61
5	0.28	0.75	0.81	0.81	0.62
6	0.25	0.74	0.83	0.82	0.61
7	0.26	0.74	0.84	0.81	0.61
8	0.24	0.74	0.84	0.82	0.63

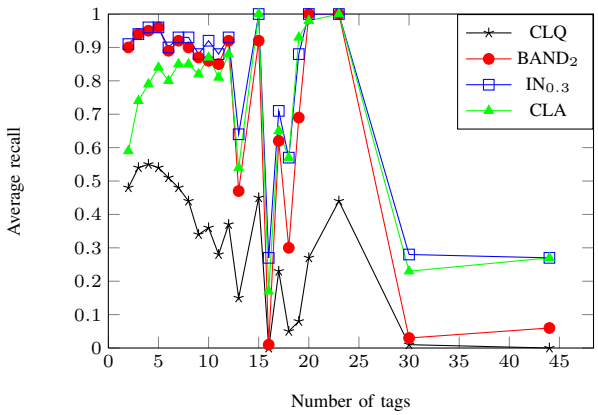

Fig. 3. Average recall vs. number of tags in the photo.

In *Table III*, we see how the number of participants (size of set P) affects the recall for various schemes. Observe that the change in the number of participants does not introduce a significant change in average recall for BAND$_K$ and IN$_K$ schemes. This suggests that our algorithm's accuracy is not affected by the size of the participant set. In contrast, the accuracy of CLA and CZG algorithms initially increase with increasing number of participants but then level out.

Figure 3 plots the average recall of the algorithm versus the number of tags on the photo. Except for photos with many

TABLE IV
AVERAGE RECALL FOR FIVE SCHEMES AND THREE DIFFERENT DATA TYPES (WITH STANDARD DEVIATIONS).

Data type	CLQ	$BAND_2$	$IN_{0.3}$
Albums	0.43 ± 0.38	0.88 ± 0.28	0.91 ± 0.24
Wall Threads	0.31 ± 0.43	0.78 ± 0.39	0.83 ± 0.36
Networks	0.13 ± 0.28	0.49 ± 0.41	0.64 ± 0.37

Data type	CLA	CZG
Albums	0.78 ± 0.37	0.62 ± 0.40
Wall Threads	0.75 ± 0.42	0.57 ± 0.47
Networks	0.55 ± 0.41	0.40 ± 0.38

(more than 25) tags, there is not a direct correlation between these two values. This suggests that our algorithm's accuracy is not affected by the social clique size.

Finally, we mention some evaluation results for other types of data. More precisely, in addition to photos, we also considered for ground truth photo albums, wall threads, and networks that a user had access to. The evaluation was done in a fashion similar to the evaluation for the photos, with the exception that the definition of T was adjusted to match the type of data. For albums, we defined T as all users tagged in an album. For wall threads, we defined T as the users who contributed to a wall thread. For networks, we defined T as the users in a network. *Table IV* shows the average recall values obtained for this evaluation.

4) Discussion: Our results indicated that the clique detection algorithm is successful in expanding a small set (seed) of people into a social clique that contains those people that are related to this seed. Moreover, these cliques are reasonably tight and do not simply grow to cover a large portion of a user's friends. We attempted to quantify the quality of our approach by leveraging the assumption that two people who appear in a common picture are also tightly related (and, hence, members of the same clique). While this assumption might not always be true, we note that our algorithm also works well when the basis for the evaluation is changed to wall threads or photo albums. Additionally, our novel expansion schemes clearly outperform existing local community detection techniques in all evaluation scenarios.

We propose that $BAND_2$ and $IN_{0.3}$ schemes are accurate enough for exposure set (friend list) generation for most social networking sites. Note that it is possible to adjust the behavior of these schemes by tuning their parameters in order to further satisfy a site's particular needs.

V. CONCLUSIONS

In this paper, we address the problem that users do not consider all their friends equal and, hence, want to hide certain pieces of information from certain groups. Unfortunately, manually managing such groups through the features provided by social networks (such as friend lists) is cumbersome and not used frequently. To solve this problem, we introduce a novel algorithm and expansion schemes that allow one to identify a social clique for a data item, starting from a small seed of friends that are directly related to this item. Our results demonstrate that the proposed algorithm is generally successful in expanding the seed into a clique that contains those people that are related to the seed. Moreover, the detected cliques are reasonably tight and do not simply grow to cover a substantial portion of a user's friends. Anecdotal evidence from user feedback and our own tests indicate that the system delivers very satisfying results. Finally, our novel expansion schemes clearly outperform existing local community detection techniques in all evaluation scenarios.

REFERENCES

[1] "Miss N.J. releases blackmail photos," http://today.msnbc.msn.com/id/19725822.

[2] R. Gross and A. Acquisti, "Information revelation and privacy in online social networks," in *Proc. of the 2005 ACM workshop on Privacy in the Electronic Society*. ACM, 2005, pp. 71–80.

[3] "Facebook Statistics," http://www.facebook.com/press/info.php?statistics.

[4] M. Siegler, "Zuckerberg: "Guess what? Nobody wants to make lists."," http://techcrunch.com/2010/08/26/facebook-friend-lists.

[5] L. Fang and K. LeFevre, "Privacy wizards for social networking sites," in *World Wide Web Confernce (WWW)*, 2010.

[6] M. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[7] G. Brown, T. Howe, M. Ihbe, A. Prakash, and K. Borders, "Social networks and context-aware spam," in *Proc. of the 2008 ACM Conference on Computer Supported Cooperative Work*. ACM, 2008, pp. 403–412.

[8] S. Guha, K. Tang, and P. Francis, "Noyb: Privacy in online social networks," in *Proc. of the first workshop on Online social networks*. ACM, 2008, pp. 49–54.

[9] L. Cutillo, R. Molva, and T. Strufe, "Privacy preserving social networking through decentralization," in *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*. IEEE, 2009, pp. 145–152.

[10] H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman, "Sybilguard: Defending against sybil attacks via social networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 16, no. 3, pp. 576–589, 2008.

[11] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All your contacts are belong to us: automated identity theft attacks on social networks," in *Proc. of the 18th International Conference on World wide web*. ACM, 2009, pp. 551–560.

[12] R. Luce, "Connectivity and generalized cliques in sociometric group structure," *Psychometrika*, vol. 15, no. 2, pp. 169–190, 1950.

[13] R. Mokken, "Cliques, clubs and clans," *Quality and Quantity*, vol. 13, no. 2, pp. 161–173, 1979.

[14] S. Seidman and B. Foster, "A graph-theoretic generalization of the clique concept*," *Journal of Mathematical Sociology*, vol. 6, no. 1, pp. 139–154, 1978.

[15] H. Small and B. Griffith, "The structure of scientific literatures I: Identifying and graphing specialties," *Science studies*, vol. 4, no. 1, pp. 17–40, 1974.

[16] M. Porter, J. Onnela, and P. Mucha, "Communities in networks," *Notices of the AMS*, vol. 56, no. 9, pp. 1082–1097, 2009.

[17] A. Clauset, "Finding local community structure in networks," *Physical Review E*, vol. 72, no. 2, p. 026132, 2005.

[18] F. Luo, J. Wang, and E. Promislow, "Exploring local community structures in large networks," *Web Intelligence and Agent Systems*, vol. 6, no. 4, pp. 387–400, 2008.

[19] J. Chen, O. Zaïane, and R. Goebel, "Local community identification in social networks," *2009 Advances in Social Network Analysis and Mining*, pp. 237–242, 2009.

[20] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney, "Statistical properties of community structure in large social and information networks," in *Proceeding of the 17th International Conference on World Wide Web*. ACM, 2008, pp. 695–704.