Retrieving Hidden Friends: A Collusion Privacy Attack Against Online Friend Search Engine

Yuhong Liu and Na Li

***Abstract*— Online social networks (OSNs) are providing a variety of applications for human users to interact with families, friends, and even strangers. One such application, the friend search engine, allows the general public to query individual users' friend lists and has been gaining popularity recently. However, without proper design, this application may mistakenly disclose users' private relationship information. Our previous work has proposed a privacy preservation solution that can effectively boost OSNs' sociability while protecting users' friendship privacy against attacks launched by individual malicious requestors. In this paper, we propose an advanced collusion attack, where a victim user's friendship privacy can be compromised through a series of carefully designed queries coordinately launched by multiple malicious requestors. The effect of the proposed collusion attack is validated through synthetic and real-world social network data sets. The in-depth research on the advanced collusion attacks will help us design a more robust and secure friend search engine on OSNs in the near future.**

***Index Terms*— Social computing, privacy, network security.**

I. Introduction

ONLINE social networks (OSNs) have become very popular in recent years, such as Facebook and Twitter, which have been part of many people's daily life. The OSNs provide different applications for people to share their information and interact with each other. One of the most popular applications is the friend search engine, which allows users to query friend lists of other users. To increase their sociability and attract more users, OSNs tend to release users' friends as many as possible, as it is believed that the larger number of common friends are displayed, the more likely the requestor and the queried user would connect later.

However, this search engine may expose more friendship information than what a queried user is willing to share, which is considered as a privacy breach. A few researchers have observed such an issue by randomly crawling an OSN through the friend search API [1]. Also, they concluded that without appropriate defenses, one could discover all users' friendships in the OSN without using many queries [2], [3]. If such a privacy breach is not well dealt with, the OSN users may feel panic and hesitate to continue using the OSNs.

In our preliminary work, we designed a privacy-aware friend display scheme in [4], which cannot only successfully preserve users' friendship privacy but also boost the sociability of the OSN. This scheme is one of the most advanced researches on preserving user privacy for friend search engines, which has been verified to successfully prevent attacks from being launched by independent attackers. However, collusion attacks, where multiple malicious requestors share their knowledge and coordinately launch queries, may make the defense scheme ineffective. In this paper, we particularly focus on the design of collusion attacks against users' friendship privacy in OSNs. The major contributions of this paper are listed as follows.

First, to the best of our knowledge, we are the first researchers studying such advanced privacy attacks as collusion attacks against friend search engine in OSNs.

Second, in-depth analysis has been provided on querying a small scale complete graph as well as a general network in various scenarios, which well explains the fundamental reasons of why and how the proposed attack is designed. In particular, we observe the defense scheme's [4] asymmetric disclosure of users' symmetric friendships. By taking advantage of it, we design an advanced collusion attack, in which multiple malicious requestors closely coordinate with one another to launch their queries on different but related users in well designed orders. The design logic can be generally applied to launch attacks against any friendship privacy preserving solutions that disclose the symmetric friendship in an asymmetric way.

Third, the proposed collusion attack is designed to carefully select which users to query, which can significantly reduce the total amount of query effort.

Fourth, to evaluate the effectiveness of our proposed attack strategy, we implement and run it on one synthetic data set and three large scale real-world data sets. Experiment results demonstrate that the proposed attack strategy works efficiently and effectively on large scale data sets. By comparing the proposed collusion attack with a naive direct attack, we find that our strategy performs better in terms of both the success rate and the required number of malicious requestors to compromise a user's friendship privacy.

Last but not least, our research on this advanced collusion attack helps us better understand the attack design and shed lights on the design of a securer privacy preserving friend search engine in the future.

Manuscript received January 17, 2018; revised May 31, 2018; accepted August 3, 2018. Date of publication August 21, 2018; date of current version October 30, 2018. This work was supported by NSF under Grant 1712496. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Anna Squicciarini. *(Corresponding author: Yuhong Liu.)*

Y. Liu is with the Department of Computer Engineering, Santa Clara University, Santa Clara, CA 95050 USA (e-mail: yhliu@scu.edu).

N. Li is with the Department of Computer Science, Prairie View A&M University, Prairie View, Prairie View, TX 77446 USA (e-mail: nali@pvamu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2018.2866309

II. Related Work

A. Preservation of OSN Friendship Privacy

As millions of users use OSNs everyday to conduct social activities, search new friends and connect to them, the development of privacy preserving friend search engines has attracted wide attention. Fogues *et al.* [5] have presented a list of threats against OSN users' relationship privacy and the corresponding requirements that privacy mechanisms should fulfill. In [6], a trust chain based friend recommendation algorithm is proposed with the purpose of preserving users' privacy. A few recent studies also work on protecting users' location information so that their sensitive friendship will not be exposed because of frequent co-locations [7]. In addition, there is a line of studies focusing on anonymizing users' sensitive relationship when OSNs publish the data to third parties [8]–[17].

Little attention, however, has been paid to addressing vulnerabilities of friend search services provided by OSN themselves, where the goal is to satisfy users' friendship privacy need while maintaining OSNs' sociability at the same time. The current solution taken by many OSNs in practice is that, each individual user can choose to either completely hide or completely display his/her entire friend list. Nevertheless, the default setting of this configuration is to expose the entire friend list, of which most users are not aware [18]. The reason for OSNs to "quietly" make such default setting is that if a large number of users make their privacy settings as to completely hide the friend list, OSN's sociability will be dramatically affected [19]. Finer grained friend list display strategies, which maintain OSNs' sociability by allowing privacy-aware users to partially display their friend lists, are required.

Some OSNs take actions accordingly. For example, Facebook has once limited the number of friends exposed in the public listings to a fixed number, eight [2]. However, even with this scheme implemented, users' friendship privacy is still under high risks due to three major issues. First, these eight exposed friends are chosen randomly so that the OSN's sociability can be maintained by displaying diverse friend lists to requestors. Bonneau *et al.* [2] show that eight friends are already enough for a third party to crawl data so as to estimate the network topology. Second, private friendship information may be breached due to inconsistent privacy settings from different users. For example, although user A hides his/her friend list from queries, his/her friendship with user B, whose friendships are all open to the public, can be disclosed when user B's friend list is queried. This problem is called mutual effect in [4]. Various attack strategies to discover private friendships by taking advantage of mutual effect are discussed in [3]. Third, the Facebook solution sets a global value, eight, as the number of friends to display for every single user, limiting the flexibility of individual users to change their privacy settings.

To enhance the protection of user friendship privacy, we have previously proposed a privacy preserving scheme in [4]. Compared to the Facebook solution, it provides the flexibility for individual users to determine the number of friends, say k, to display in response to friend queries. In addition, these "k" friends are determined as a fixed set of the most "influential" friends to avoid privacy breach caused by randomness while still maintaining OSNs' sociability. To the best of our knowledge, it is also the first work to successfully handle the mutual effect while considering OSNs' sociability. With the deployment of this scheme, none independent attackers can violate the privacy of any given target user [4].

B. Attacks Against OSN Friendship Privacy

Along with defense research, privacy attacks against OSN friendship privacy are also extensively studied in the scientific community. Specifically, such research can be classified into two categories as attacks launched by (1) independent attackers or (2) colluded attackers.

There are ample examples of independent attacks. A neighborhood attack defined in [9] studies that an individual attacker with some knowledge of the neighbors of a target node and their relationship information is able to identify the target node from a social network graph where user identifiers are removed. Additionally, a few researchers have observed that by randomly crawling an OSN through the friend search API [1], an individual attacker can glean all friendship connections of users in the OSN without many queries [2], [3].

Collusion attacks can be defined as attacks that involve multiple malicious entities aiming at obtaining greater gain than what the entities benefit from individually launched attacks. The multiple entities can be fake accounts created by a single attacker or by different real attackers [20]–[23]. Compared to individual attacks, collusion attacks can use more complicated attack strategies and often exploit system vulnerabilities that cannot be discovered by individual attacks.

There are a few simple collusion attacks researched in the literature. For example, as mentioned in [24], some revoked users can collude with legitimate users to continue accessing the private data. However, less attention has been paid to designing sophisticated collusion attack strategies where multiple malicious users conduct their behaviors in a highly coordinated way and dynamically adjust their behaviors according to systems' feedback on other colluders' behaviors. To our best knowledge, there is very limited research on designing comprehensive collusion attacks against user friendship privacy in OSNs.

In this paper, we fill in this gap by proposing an advanced collusion attack where multiple attackers share their query results with one another and coordinately launch their queries based on others' query results so that they can successfully violate users' friendship privacy.

III. Adopted Defense Scheme

We research the advanced collusion attack upon the defense scheme proposed in our previous work [4], which designs a privacy-aware display strategy to handle the tradeoff between preserving user privacy and facilitating site sociability.

In [4], individual users are allowed to control their privacy by setting a k value to indicate the number of friends that they are willing to display in response to queries through the

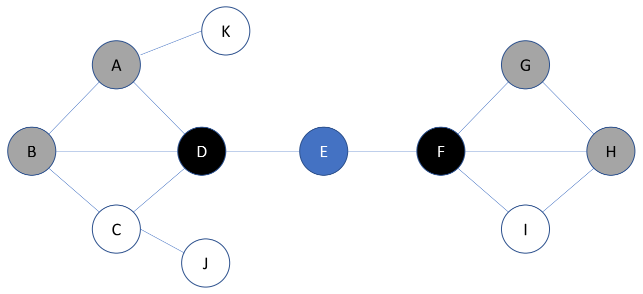

Fig. 1. Illustration of previous defense scheme.

friend search engine. Different users can set different k values according to their privacy preferences. We also assume that an individual adversary is motivated to make many queries to discover more friends than what users are willing to expose to the public. The privacy preservation goal is to ensure that all the users, including both the users queried and their friends released in the query results, will have no more than k friends exposed. On the other hand, the sociability of the site, which is measured by the impact of the nodes appearing in the query results, should be considered because ocial site owners are prone to show their users who can stimulate more interactions on the sites, such as motivating the querier to request to be friend with any displayed users. The tradeoff addressed in [4] is not trivial because the defender (i.g., social site administrator/owner) can hardly predict the sequence of the users the adversary plans to query. Therefore, it is very challenging to provide the optimal set of users in response, where the optimization means maximizing the sociability without breaching the privacy. The proposed defense algorithm that we proposed in [4] was able to well address such tradeoff in a heuristic way.

Technically, we use a graph model to represent a friendship network, where nodes and edges denote users and their friendships, respectively. When a node x is queried, the defense strategy first returns the friends whose connections with x were already disclosed. Next, it sorts the rest neighboring friends of x in the descending order of their impact on the sociability of the OSN, checks each friend from the beginning of the sequence and then chooses the neighboring friends for release if the disclosure of their connections with x won't violate the privacy of other friends who are visible in the query results. In total, it will display no more than k friends for x. The purpose of sorting friends is to ensure that friends with high sociability impact will be considered first in responding to queries. This is how the scheme handles the aforementioned trade-off between privacy and sociability. Note that in the rest of this paper, nodes with high sociability impact in the OSN are named as influential nodes for simplicity reasons.

In particular, we illustrate the defense scheme through an example in Figure 1. Assume $k = 2$, when a requestor queries node D and node F, the defense scheme will release node A and B as D's friends, and node G and H as F's friends. If node C is queried by the same requestor, although D is the high influential friend, node B and J will be released as D already has its two friends released. If node E is queried by the same requestor, the defense scheme will not release any friends as both D and F have reached their maximum number of released friends.

By adopting the above mentioned defense scheme, the OSN can record the prior query results for each individual requestor. Such records are used to ensure that (1) the newly released query results for any individual requestor will not violate the privacy of any nodes that are queried by this requestor before; and (2) when a node is queried by different requestors with the same query history/order, the released query results are always the same. The adopted defense scheme has been verified to be effective when defending against independent attacks [4].

IV. Collusion Attack Strategy

The adopted defense scheme achieves its privacy preservation goal by recording each requestor's query history and dynamically adjusting friendships to disclose based on the requestor's query history/order. However, it also opens the possibilities for multiple colluded malicious requestors to coordinately launch their queries in carefully designed orders to retrieve extra information. In this section, we will investigate the potential collusion attack strategies in details.

A. Attack Model

1) Attack Assumptions: In this work, we make the following assumptions.

- Defense scheme. Network nodes' privacy can be easily violated if no defense scheme is employed. In this paper, we adopt the scheme proposed in [4] as the defense scheme to ensure that queries launched by individual malicious requestor cannot violate users' privacy. In addition, the top k friends displayed by the defense scheme are the nodes with the highest influence on the sociability of the OSN.
- The maximum number of friends to display - the k value. This k value may vary due to different nodes' personal preferences. However, as this work is the first step to investigate collusion attacks, we assume all the network nodes set the same k value for simplicity reason. If a node has friends less than k, it is impossible to violate this node's privacy. The study of assigning different k values to different nodes will be conducted in the future work.
- Multiple **malicious requestors** collude to compromise the victim's privacy by coordinately launching queries on network nodes' friendships and sharing their query results. These malicious requestor accounts can be created and manipulated by either a single or multiple real human attacker(s).
- Attackers' prior knowledge. Generally speaking, the design of attack strategies is closely related with attackers' knowledge of the network. Attackers with more knowledge tend to be more successful in achieving their attack goals. In this paper, we assume that the attacker has very limited prior knowledge - only the existence of the victim node in the OSN.

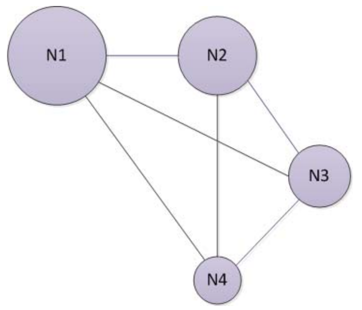

Fig. 2. A complete graph with four nodes.

2) Attack Goal: The attack goal is to violate privacy of a given victim node, in other words, to retrieve more than k of the victim node's friends. Note that there is only one victim node for each collusion attack. Although some other nodes' privacy may be violated through the collusion attack, we consider a collusion attack to be successful only when the victim node's privacy is breached.

3) Attack Challenge: The design of collusion attacks is challenging. Specifically, if colluders behave in exactly the same way as querying the same set of users in the same order, each one of them will retrieve the same friendship information due to the adoption of the defense scheme [4]. Such collusion fails as it retrieves no more information than independent attacks do. On the other hand, if colluders' query behaviors deviate too much, which leads to completely non-related query results, the merging of such query results will not be able to facilitate the privacy violation at all.

A successful collusion attack should have all the colluded malicious requestors dynamically adjust their query targets and orders according to their accomplices' query results, so that the victim node's privacy can be violated in an effective and efficient way. The design of such attacks is not trivial. Please note that although the malicious requestors only know the victim node before attack, they will learn more social network users through continuous queries. Assume m represents the total number of users that have been exposed to the colluded malicious requestors so far, there would be $m!$ unique query orders to query these m nodes. Having $m!$ malicious requestors where each requestor tries one unique order is intractable even with a moderate m value. More importantly, the m value may keep growing when malicious requestors continue their queries.

In this paper, we start by investigating collusion attacks in a special social network graph, and then generalize the strategies to fit more general network graphs.

B. Collusion Attacks in a Small Scale Complete Graph

To investigate collusion attack strategies, we would like to start with a simple scenario - a small scale complete graph with four nodes, as shown in Figure 2, where each node connects to all other nodes.

For simplicity, we assume that each node only allows the OSN to release one of its friends (i.e., $k = 1$). Assume these nodes have different influences on the sociability of the OSN, as $I_{N_1} > I_{N_2} > I_{N_3} > I_{N_4}$, where I_{N_i} represents the influence of node N_i. Note that the influence can be measured and quantified through various information gathered by the OSNs. We differentiate these nodes in sizes to represent their different influences. In the following discussion, we will check, for each individual node given in the Figure 2, whether a collusion attack can successfully violate its privacy.

Recall the assumption in Section IV-A that malicious requestors have no knowledge about the network except the victim node. Collusion attacks under such assumption should always start by querying the victim node. During the query process, malicious requestors gradually learn more nodes that are exposed in their query results, and from which they further choose nodes to query afterwards.

First, let's consider N_1, the most influential node, as the victim node. Malicious requestor MR_1 starts off by querying N_1, and then the OSN will respond with the friendship between N_1 and N_2 (i.e. $E_{(N_1,N_2)}$), since N_2 is the most influential friend of N_1. If MR_1 continues to query N_2, and the OSN will respond with $E_{(N_2,N_1)}$, as N_1 is also the most influential friend of N_2. Note that $E_{(N_1,N_2)}$ and $E_{(N_2,N_1)}$ represent the same relationship due to the symmetry of friendship. By queries, MR_1 will not be able to detect other friendship information than $E_{(N_1,N_2)}$. Therefore, individual attack fails.

As a matter of fact, even if a new malicious requestor MR_2 is involved to query N_2, no additional information can be discovered, since both N_1 and N_2 are the top one influential friend for each other. The collusion attack fails as well. The query process is summarized as follows.

MR_1:
Query $N_1 ->$ retrieve $E_{(N_1,N_2)}$ *(N_1's top 1)*
Query $N_2 ->$ retrieve $E_{(N_2,N_1)}$ *(N_2's top 1)*
MR_2:
Query $N_2 ->$ retrieve $E_{(N_2,N_1)}$ *(Not Violate)*
Collusion Attack Result: Fail

In the same way, if the victim node is N_2, regardless of the number of malicious requestors used in the attack, only the friendship $E_{(N_1,N_2)}$ will be retrieved. The privacy of N_2 cannot be violated.

Next, let us consider N_3 as the victim node. In other words, the attacker(s) aim to find more than one friend of N_3. Malicious requestor MR_1 starts off by querying node N_3, and the OSN will return $E_{(N_1,N_3)}$ as N_1 is the most influential friend of N_3. Then if MR_1 continues to query N_1, the OSN will return $E_{(N_1,N_3)}$ instead of $E_{(N_1,N_2)}$ although N_2 is actually the most influential friend of N_1. The reason is that the defense scheme [4] records the query history of MR_1 and figures out that releasing $E_{(N_1,N_2)}$ will disclose two friends of N_1, thereby violating N_1's privacy. The defense scheme [4] can well protect nodes' privacy against individual malicious queries.

MR_1:
Query $N_3 ->$ retrieve $E_{(N_3,N_1)}$ *(N_3's top 1)*
Query $N_1 ->$ retrieve $E_{(N_1,N_3)}$ *Protect N_1's privacy*
Individual Attack Result: Fail

However, if a new malicious requestor MR_2 gets involved and queries N_1, $E_{(N_1,N_2)}$ will be released since the OSN considers MR_2 as a new requestor who has no prior knowledge about N_1. Due to the release of $E_{(N_1,N_2)}$, both N_1 and N_2 have reached their privacy restriction settings where $k = 1$. We consider that N_1 and N_2 are **occupied** by MR_2, since no more of their friendships can be released to MR_2. The definition of occupation is given as follows.

Definition 1 (Occupation): Given a victim node N_x and its friend N_y, N_y is considered as "occupied" by a requestor R_i if the defense scheme believes that R_i has already learned N_y's k friends (excluding N_x).

Then the same requestor MR_2 can continue to query the victim node N_3. As a result of the occupation, the defense scheme will not release $E_{(N_1,N_3)}$ or $E_{(N_2,N_3)}$. Instead, it has to display another friendship of N_3, $E_{(N_3,N_4)}$, considering node N_4 is the next most influential friend of N_3. As a consequence, N_3's privacy is violated as the attackers can see two of N_3's friends, while $k = 1$. We summarize the violation process as follows.

$\mathbf{MR_1}$:
Query N_3−> retrieve $E_{(N_3,N_1)}$ *(N_3's top 1)*
$\mathbf{MR_2}$:
Query N_1−> retrieve $E_{(N_1,N_2)}$ *(Occupy N_1)*
Query N_3−> retrieve $E_{(N_3,N_4)}$ *(Violate)*
Collusion Attack Result: N_3's privacy violated

In the similar way, if the victim node is N_4, its privacy can also be violated by occupying node N_1. As a summary, we observe that when $k = 1$, the collusion attack can succeed for all other nodes except for the top 2 influential nodes.

C. Observations and Analysis

1) Observations: Through the above simple case studies in a small-scale complete graph, we are able to make some interesting observations.

- For the top $k + 1$ influential nodes, collusion attacks cannot retrieve more information than individual attacks and thus cannot successfully violate privacy of any one of these nodes. We consider these nodes form a **"Black Clique"**. A clique is a graph in which all nodes directly connect to each other [25].
- For all other nodes, colluded malicious requestors are able to violate their privacy through **occupation**. In particular, colluded malicious requestors share their query results with each other, while hiding their knowledge from the defense scheme and fooling the defense scheme to treat them as independent requestors.

2) Analysis: In this section, we will study the fundamental reason behind the above observations. We start the investigation from the basic relationship: a friendship between two nodes. In particular, the existence of an edge $E_{(A,B)}$ between two nodes A and B indicates their friendship.

We use two different sizes, as large and small, to indicate whether a node is in the top k influential friend list of its friend.

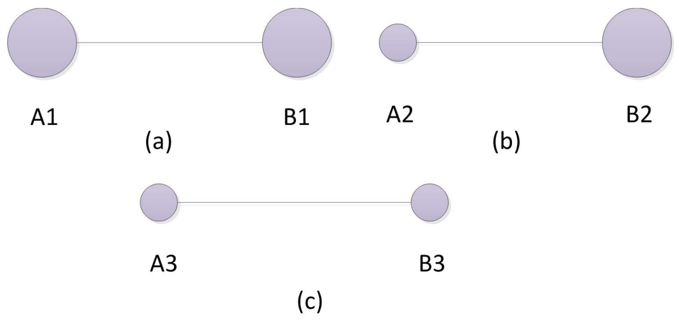

Fig. 3. Three types of friendship between two nodes.

For example, node A is a large node if it is one of node B's top k influential friends. Otherwise, it will be a small node. The size of node B is determined in the same way. According to the discussion above, the friendship between two nodes can be classified into three categories in terms of their relative sizes: large-large friendship, small-large (or large-small) friendship and small-small friendship, as shown in Figure 3. We discuss these three types of friendship in details in the following paragraphs.

a) Large-large friendship: The large-large friendship, as illustrated in Figure 3-(a), represents that both node A_1 and node B_1 are in each other's top k friend list. Therefore, the OSN will respond the friendship $E_{(A_1,B_1)}$ to the queries on either A_1 or B_1. This is considered as ***the defense system's symmetric responses on the symmetric friendship***. In this case, *node A_1's privacy cannot be compromised by directly querying B_1 regardless the number of malicious requestors involved in the collusion attack.*

b) Top k + 1 "black clique": The large-large friendship explains the observation of the top $k + 1$ "black clique" discussed previously. In the previous discussion, we observe that the top $k+1$ nodes (i.e. N_i, $1 \leq i \leq k+1$) form a "black clique" in the graph illustrated in Figure 2. For these top $k+1$ nodes, they have large-large friendship between any two of them. Therefore, no matter which node is queried, the OSN always releases the other k most influential nodes as friends. No additional friendships can be discovered. In this scenario, neither individual attack nor collusion attack is able to violate their privacy.

c) Small-large friendship: In the small-large friendship illustrated in Figure 3-(b), A_2 is represented by a small node and B_2 is represented by a large one, indicating that B_2 is one of A_2's top k friends while A_2 is not in B_2's top k friend list. In this scenario, when a new requestor launches a query, the same friendship $E_{(A_2,B_2)}$ will be released if A_2 is queried, or be hidden if B_2 is queried. We consider this as ***the defense scheme's [4] asymmetric responses on the symmetric friendship***. The defense scheme is designed to release a node's top k influential friends to boost OSN's sociability. In addition, it also helps the OSN prevent privacy leakage caused by random disclosure.

However, taking advantage of this, attackers can violate the privacy of the small node by using only two new requestors marked as MR_1 and MR_2. Specifically, attackers can retrieve the friendship between A_2 and B_2 by using MR_1 to query A_2. Although this information is shared with MR_2, the defense scheme is fooled due to the belief that MR_2 does not know this friendship. Then node B_2 will be "occupied" once it is queried by MR_2, as its maximum number of friends (excluding A_2) are released. Now when MR_2 continues to query A_2, the defense scheme [4] will hide $E_{(A_2,B_2)}$ so that B_2 will not have more than k friends exposed to MR_2. To disclose k of A_2's friends, the defense scheme has to replace $E_{(A_2,B_2)}$ by another new friendship of A_2 which has not been discovered by MR_1. As a result, the privacy of A_2 is compromised.

$\mathbf{MR_1}$:

Query A_2—> retrieve top k friendships (including $E_{(A_2,B_2)}$)

$\mathbf{MR_2}$:

Query B_2—> retrieve top k friendships (excluding $E_{(B_2,A_2)}$)

Query A_2—> retrieve top k friendships (excluding $E_{(A_2,B_2)}$)

Collusion Attack Result: A_2*'s privacy violated*

As a summary, if the victim node is identified to be involved in a small-large friendship, in which it is the small node, its privacy can be violated when a fresh malicious requestor queries the large friend first and the victim node next. This inspires our design of collusion attack strategies in a general network later.

d) Small-small friendship: A small-small friendship, as shown in Figure 3-(c), represents that although A_3 and B_3 are friends, they are not in each others' top k friend list. Therefore, when either A_3 or B_3 is queried, the other node will not be released as a top k friend of the queried node. As a result, it is hard for attackers to discover this kind of relationship, let alone directly utilizing it in collusion attacks. Thus, we will not focus on this relationship in this paper.

D. Collusion Attacks in a General Network

In the above sections, we have investigated collusion attacks in a small scale complete graph. The investigation not only helps us better understand the working mechanism of collusion attacks in a special context but also sheds light on the design of attack strategies in more general social network graphs. In this section, we will further investigate collusion attack strategies in general social network settings.

Recall that if any malicious requestor is able to occupy at least one of the top k friends of the victim node, he/she can then compromise the victim node by directly querying it and retrieving its additional friend(s). Following this logic, the original attack goal of violating the victim's privacy is converted to a new goal as how to occupy at least one of the victim's top k friends.

Through this study, we discover that the answer to this "how" question actually depends on the relationship between the victim node and its top k friends. According to such relationship, we would like to first classify victim nodes into two categories as **popular nodes** and **non-popular nodes**, and then answer the "how" question accordingly.

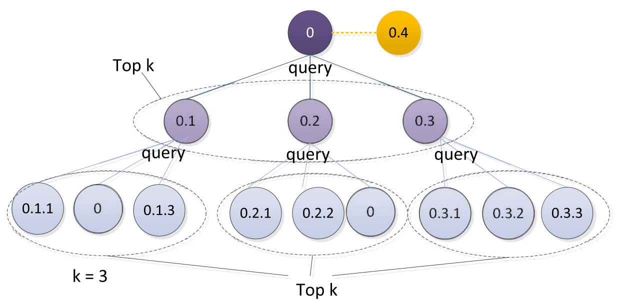

Fig. 4. An example of violating a non-popular node.

In particular, given an arbitrary network node N_a and its top k friend list $F_a^k = \{N_{a.i} : i = 1 \dots k\}$, assuming each of these top k friends has its own top k list as $F_{a.i}^k = \{N_{a.i.j} : j = 1 \dots k\}$, N_a will be determined as either a non-popular or a popular node as follows.

Definition 2 (Non-Popular Node): N_a is a non-popular node if $\exists N_{a.i} \in F_a^k$, and $N_a \notin F_{a.i}^k$. In other words, a non-popular node has at least one small-large friendship with its top k friends, where the non-popular node is small in that friendship.

Definition 3 (Popular Node): N_a is a popular node if $N_a \in F_{a.i}^k$ for $\forall N_{a.i} \in F_a^k$. In other words, a popular node has large-large friendships with all its top k influential friends.

According to whether a victim node is a non-popular node or a popular one, the collusion attack strategy to "occupy" its top k friends will be different. In the following subsections, we will discuss collusion attack strategies against these two types of nodes separately.

1) Violating Non-Popular Nodes: Given a non-popular node N_a as the victim, malicious requestors can always identify at least one node $N_{a.i}$ where $N_{a.i} \in F_a^k$, but $N_a \notin F_{a.i}^k$. A fresh malicious requestor will then be able to easily occupy $N_{a.i}$ by directly querying it. The victim node N_a's privacy will then be violated when it is queried next.

Let's use a simple example illustrated in Figure 4 to demonstrate the collusion attack against a non-popular victim node N_0, where $k = 3$. Specifically, in Figure 4, each node is connected to its top 3 friends through solid lines and to its other friends through dotted lines. In addition, we place the top 3 friends in the order from the left to the right. For example, $N_{0.1}$, $N_{0.2}$ and $N_{0.3}$ are placed at the left, the middle and the right under N_0, indicating that they are N_0's top 1, 2 and 3 friends, respectively.

From Figure 4, we observe a large-large friendship between N_0 and $N_{0.1}$ since both of them have listed each other as their top k friend. The same observation can be made for N_0 and $N_{0.2}$. Node $N_{0.3}$, however, does not have N_0 as its top k. It indicates that the friendship between N_0 and $N_{0.3}$ is a small-large friendship, where N_0 is the small node. Therefore, N_0 is a non-popular node, whose privacy can be violated if malicious requestors occupy $N_{0.3}$. Specifically, since the malicious requestors don't know which nodes have the small-large friendship with the victim node, all discovered k friends of the victim node need to be queried. We illustrate

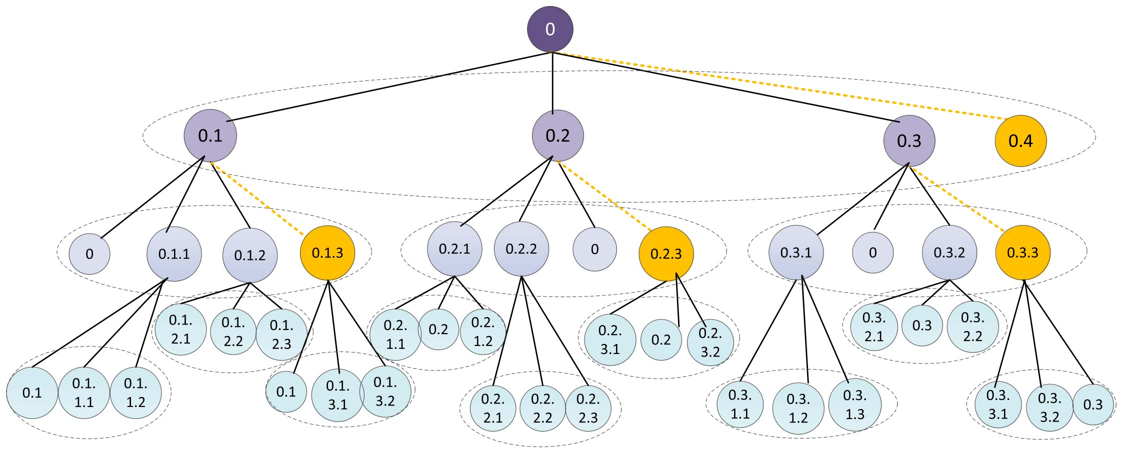

Fig. 5. An example of violating a popular node.

the collusion attack procedure as follows.

MR$_1$:

Query $N_0 ->$ retrieve $E_{(N_0,N_{0.1})}$, $E_{(N_0,N_{0.2})}$, $E_{(N_0,N_{0.3})}$

MR$_2$:

Query $N_{0.1} ->$ retrieve $E_{(N_{0.1},N_{0.1.1})}$, $E_{(N_{0.1},N_0)}$, $E_{(N_{0.1},N_{0.1.3})}$

Query $N_{0.2} ->$ retrieve $E_{(N_{0.2},N_{0.2.1})}$, $E_{(N_{0.2},N_{0.2.2})}$, $E_{(N_{0.2},N_0)}$

Query $N_{0.3} ->$ retrieve $E_{(N_{0.3},N_{0.3.1})}$, $E_{(N_{0.3},N_{0.3.2})}$, $E_{(N_{0.3},N_{0.3.3})}$

// $N_{0.3}$ is occupied

Query $N_0 ->$ retrieve $E_{(N_0,N_{0.1})}$, $E_{(N_0,N_{0.2})}$, $E_{(N_0,N_{0.4})}$

Collusion Attack Result: Succeed

2) Violating Popular Nodes: In this section, we will discuss how to compromise a popular node's privacy. The basic idea, like compromising a non-popular node, is also to occupy at least one of the victim node's top k friends. However, this time directly querying these friends will not be able to "occupy" them (Definition 1), as the victim node appears in the top k friend lists of all its top k friends. Therefore, we propose an alternative way to occupy them indirectly, which is defined as **"passive displays"**.

Definition 4 (Passive Display): A node N_x is passively displayed once to a requestor R_i, if R_i queries one of N_x's friends N_y and retrieve the friendship $E_{(N_x,N_y)}$.

The name "passive display" comes from the fact that the node N_x's friendship is displayed to a requestor although it is not directly queried. We further illustrate this concept with examples in Figure 5, where $k = 3$. Let us use $N_{0.1}$ and $N_{0.1.1}$ as an example. Assume $N_{0.1.1}$ is queried by a requestor R_i, its friendship with $N_{0.1}$, its top 1 friend, will be exposed to R_i, although $N_{0.1}$ is not queried by R_i directly. In this case, we consider $N_{0.1}$ is passively displayed once.

According to Definition 1, if one of the victim node's (i.e. N_a) top k friends, say $N_{a.i}$, can be passively displayed enough times without showing its friendship with N_a, it is "occupied." The remaining violation process becomes the same as violating a non-popular node. We consider the node to occupy (i.e. $N_{a.i}$) as the target node. Therefore how to find "sufficient number" of friends to passively display the target node becomes the key issue. It turns out that this number varies according to the rank of N_a in $N_{a.i}$'s top k friend list.

Let us assume that N_a is ranked as the r^{th} friend of $N_{a.i}$, where $1 \le r \le k$. The rest $k-1$ friends of $N_{a.i}$ can then be placed into two sets:

$$F^{k.1}_{a.i} = \{N_{a.i.p} : 1 \le p < r\} \quad F^{k.2}_{a.i} = \{N_{a.i.q} : r < q \le k\}, \tag{1}$$

where $N_{a.i.p}$ and $N_{a.i.q}$ represent friends that are ranked above and below N_a, respectively. When $N_{a.i}$ is queried, the $r-1$ friends in $F^{k.1}_{a.i}$ will be released before N_a anyway. Since the OSN will only display k friends of $N_{a.i}$, in order to "push" N_a out of the k released friends of $N_{a.i}$, the malicious requestors need to retrieve **another** $k-(r-1)$ **friends** who can passively display $N_{a.i}$. $N_{a.i}$'s friends in set $F^{k.2}_{a.i}$ may help.

In the following paragraphs, we set N_0 as the victim node and use its top 3 friends, $N_{0.1}$, $N_{0.2}$ and $N_{0.3}$, as examples to demonstrate the occupation process. Please note that none of the three friends, $N_{0.1}$, $N_{0.2}$ or $N_{0.3}$, can be occupied by direct querying since they all display N_0 as their top 3 friend when queried.

In the first scenario, we set $N_{0.1}$ as the target node to occupy. As shown in Figure 5, N_0 is ranked as the top 1 friend of $N_{0.1}$, indicating an empty set of $F^{k.1}_{N_{0.1}}$. To occupy $N_{0.1}$, we need to passively display it three times (i.e. $k-(r-1)$, where $k=3, r=1$) so that its friendship with N_0 will not be displayed by the OSN. Due to the limited attack knowledge, malicious requestors can only query $N_{0.1.1}$ and $N_{0.1.2}$.

As shown in Figure 5, when $N_{0.1.1}$ is queried, $N_{0.1}$ can be passively displayed once, whereas when $N_{0.1.2}$ is queried, $N_{0.1}$ cannot be passively displayed since it is not in $N_{0.1.2}$'s top k list. Intuitively, it seems that $N_{0.1}$ can only be passively displayed once. However, $N_{0.1.2}$ is actually "occupied" as it does not display $N_{0.1}$ when queried. Such occupation can be used to retrieve a new node that can passively display $N_{0.1}$.

Specifically, when $N_{0.1.2}$ is queried by a fresh malicious requestor, $N_{0.1.2.1}$, $N_{0.1.2.2}$ and $N_{0.1.2.3}$ will be released as its top 3 friends. Then when $N_{0.1}$ is queried by the same requestor, $N_{0.1.2}$ cannot be displayed as $N_{0.1}$'s friend since it already has the maximum number of friends displayed. As a consequence, a new friend of $N_{0.1}$ (e.g. $N_{0.1.3}$) will be retrieved. If $N_{0.1.3}$ can passively display $N_{0.1}$, a malicious requestor will be able to passively display $N_{0.1}$ twice by querying $N_{0.1.1}$ and $N_{0.1.3}$. Otherwise, similar to $N_{0.1.2}$, $N_{0.1.3}$ can be used to retrieve another new friend of $N_{0.1}$. This process can be repeated until a friend that can passively display $N_{0.1}$ is found.

As a result, with two friends ranked below N_a, malicious requestors can always passively display $N_{0.1}$ twice. However, $N_{0.1}$ can only be occupied if it can be passively displayed by k times, where $k = 3$. Consequently, the attack will fail. We demonstrate the detailed process below.

Through this process, we observe that **a target node $N_{a.i}$ cannot be occupied if $F^{k.1}_{N_{a.i}}$ is empty**. However, for each node $N_{a.i.q}$ in set $F^{k.2}_{a.i}$, it can either be used to passively display $N_{a.i}$ or be occupied to retrieve a new node $N_{a.i.w}$, $w > k$ that can passively display $N_{a.i}$, leading to $k-r$ times of passively displays of $N_{a.i}$. Nevertheless, recall that $N_{a.i}$ can only be occupied when it is passively displayed by $k-(r-1)$ times, with only nodes in set $F^{k.2}_{a.i}$, $N_{a.i}$ cannot be occupied. Hence, we need to consider utilizing nodes in set $F^{k.1}_{a.i}$.

MR$_1$:

Query $N_0 -> E_{(N_0,N_{0.1})}, E_{(N_0,N_{0.2})}, E_{(N_0,N_{0.3})}$

MR$_2$:

Query $N_{0.1} -> E_{(N_{0.1},N_0)}, E_{(N_{0.1},N_{0.1.1})}, E_{(N_{0.1},N_{0.1.2})}$

MR$_3$:

Query $N_{0.1.1} -> E_{(N_{0.1.1},N_{0.1})}, E_{(N_{0.1.1},N_{0.1.1.1})}, E_{(N_{0.1.1},N_{0.1.1.2})}$

// passively display $N_{0.1}$

Query $N_{0.1.2} -> E_{(N_{0.1.2},N_{0.1.2.1})}, E_{(N_{0.1.2},N_{0.1.2.2})}$, *and* $E_{(N_{0.1.2},N_{0.1.2.3})}$

// occupy $N_{0.1.2}$

Query $N_{0.1} -> E_{(N_{0.1},N_0)}, E_{(N_{0.1},N_{0.1.1})}, E_{(N_{0.1},N_{0.1.3})}$

// retrieve a new friend: $N_{0.1.3}$

MR$_4$:

Query $N_{0.1.3} -> E_{(N_{0.1.3},N_{0.1})}, E_{(N_{0.1.3},N_{0.1.3.1})}, E_{(N_{0.1.3},N_{0.1.3.2})}$

// passively display $N_{0.1}$ *once*

Query $N_{0.1.1} -> E_{(N_{0.1.1},N_{0.1})}, E_{(N_{0.1.1},N_{0.1.1.1})}, E_{(N_{0.1.1},N_{0.1.1.2})}$

// passively display $N_{0.1}$ *twice*

Query $N_{0.1} -> E_{(N_{0.1},N_0)}, E_{(N_{0.1},N_{0.1.1})}, E_{(N_{0.1},N_{0.1.3})}$

Collusion Attack Result: Fail

In the second scenario, we set $N_{0.2}$ as the target node to occupy. As shown in Figure 5, N_0 is ranked as the third friend of $N_{0.2}$ (i.e. $r = 3$), indicating that both $N_{0.2.2}$ and $N_{0.2.3}$ belong to the set of $F^{k.1}_{a.i}$. To occupy $N_{0.2}$, as discussed before, the malicious requestors only need to retrieve one extra friend (i.e. $k-(r-1)$, where $k = 3, r = 3$) beyond $N_{0.2.1}$ and $N_{0.2.2}$, say $N_{0.2.3}$, that can passively display $N_{0.2}$.

Then the key part becomes how to retrieve the new friend $N_{0.2.3}$. Let us take a closer look at $N_{0.2.1}$ and $N_{0.2.2}$. When $N_{0.2.1}$ is queried, $N_{0.2}$ will be displayed as a friend, which could not help to retrieve $N_{0.2.3}$. Different from $N_{0.2.1}$, when $N_{0.2.2}$ is queried, $N_{0.2.2.1}$, $N_{0.2.2.2}$ and $N_{0.2.2.3}$ will be released as its top 3 friends, while $N_{0.2}$ is not displayed. This way, $N_{0.2.2}$ is occupied. In other words, when $N_{0.2}$ is queried by the same requestor afterwards, $N_{0.2.2}$ cannot be displayed, leading to the retrieval of another new friend of $N_{0.2}$.

The following process becomes exactly the same as the first scenario. If the newly released friend cannot passively display $N_{0.2}$, it can be occupied to retrieve another new friend of $N_{0.2}$. Hence, we are always able to retrieve a new friend (i.e. $N_{0.2.3}$) that can passively display $N_{0.2}$. Once $N_{0.2.3}$ is retrieved, a fresh malicious requestor can query $N_{0.2.3}$ first to passively display $N_{0.2}$ once, then query $N_{0.2}$ next. Because the OSN believes that the requestor already obtains one friendship of $N_{0.2}$, only two more friendships will be released. Following the friend rank, only $N_{0.2.1}$ and $N_{0.2.2}$ will be released and N_0 will be "pushed out" of the top k list. As a result, when N_0 is queried afterwards, the OSN will display three friends of N_0 without showing $E_{(N_0,N_{0.2})}$.

Through this process, we observe that **if any node in the set $F^{k.1}_{a.i}$ can be occupied when it is directly queried, $N_{a.i}$ will be occupied through passive displays**. The reason is as follows. First, malicious requestors can easily display $k-1$ friends of $N_{a.i}$, because (1) the $k-r$ friends in set $F^{k.2}_{a.i}$ can be utilized to passively display $N_{a.i}$ for $k-r$ times; and (2) the $r-1$ friends in set $F^{k.1}_{a.i}$ can be displayed before N_a when $N_{a.i}$ is queried. Second, if any node in the set $F^{k.1}_{a.i}$ can be occupied when it is queried, a new friend of $N_{a.i}$ will be retrieved. This newly retrieved friend can be used either to passively display $N_{a.i}$ or to retrieve another node that can passively display $N_{a.i}$.

MR$_1$:

Query $N_0 -> E_{(N_0,N_{0.1})}, E_{(N_0,N_{0.2})}, E_{(N_0,N_{0.3})}$

MR$_2$:

Query $N_{0.2} -> E_{(N_{0.2},N_{0.2.1})}, E_{(N_{0.2},N_{0.2.2})}, E_{(N_{0.2},N_{0.2.3})}$

MR$_3$:

Query $N_{0.2.1} -> E_{(N_{0.2.1},N_{0.2.1.1})}, E_{(N_{0.2.1},N_{0.2})}, E_{(N_{0.2.1},N_{0.2.1.2})}$

Query $N_{0.2.2} -> E_{(N_{0.2.2},N_{0.2.2.1})}, E_{(N_{0.2.2},N_{0.2.2.2})}, E_{(N_{0.2.2},N_{0.2.2.3})}$

// $N_{0.2.2}$ *is occupied*

Query $N_{0.2} -> E_{(N_{0.2},N_{0.2.1})}, E_{(N_{0.2},N_0)}, E_{(N_{0.2},N_{0.2.3})}$

// retrieve a new friend: $N_{0.2.3}$

MR$_4$:

Query $N_{0.2.3} -> E_{(N_{0.2.3},N_{0.2.3.1})}, E_{(N_{0.2.3},N_{0.2})}, E_{(N_{0.2.3},N_{0.2.3.2})}$

// $N_{0.2.3}$ *can passively display* $N_{0.2}$

Query $N_{0.2} -> E_{(N_{0.2},N_{0.2.1})}, E_{(N_{0.2},N_{0.2.2})}, E_{(N_{0.2},N_{0.2.3})}$

// $N_{0.2}$ *is occupied*

Query $N_0 -> E_{(N_0,N_{0.1})}, E_{(N_0,N_{0.3})}, E_{(N_0,N_{0.4})}$

Collusion Attack Result: Succeed

Nevertheless, it is possible that none of the nodes in the set $F^{k.1}_{a.i}$ can be occupied when it is directly queried. Let us use another example to illustrate such scenario.

In the third scenario, we set $N_{0.3}$ as the target node to occupy. As shown in Figure 5, the victim node N_0 is ranked as the second friend of $N_{0.3}$, in the middle of $N_{0.3.1}$ and $N_{0.3.2}$. We first check if $N_{0.3.1}$, the only node ranked above N_0, can be occupied. If yes, following our observations made in the second scenario, $N_{0.3}$ can be occupied. Nevertheless, as shown in Figure 5, $N_{0.3.1}$ displays $N_{0.3}$ as one of its top three friends, indicating that $N_{0.3.1}$ cannot be occupied when it is directly queried.

To handle this type of scenario, we propose a recursive process by setting $N_{0.3.1}$ as the second layer target, shifting our focus from occupying the first layer target (i.e. $N_{0.3}$) through passive displays to occupying the second layer target $N_{0.3.1}$ through passive displays. **Please note that as discussed in the first scenario, to occupy the first layer target, the second layer target (e.g. $N_{0.3.1}$) has to be ranked above the victim node (e.g. N_0) - the parent node of the first layer target (e.g. $N_{0.3}$)**. Once $N_{0.3.1}$ is occupied, a new friend that passively displays $N_{0.3}$ (e.g. $N_{0.3.4}$) can be retrieved, which causes the occupation of $N_{0.3}$, leading to the violation of N_0's privacy in the end.

This process actually indicates the "occupation chain": occupying the target node by occupying its friend, which pushes malicious queries to the next layer - "friends of friends." The recursive process will succeed once a target node can be occupied, or fail in two conditions: (1) the maximum number of layer has been reached or (2) the current target (i.e. N_t) cannot be occupied and there is no node available at the next layer that can be chosen as the next layer target. We demonstrate the example of the detailed attack process as follows.

MR$_1$:
Query $N_0 -> E_{(N_0,N_{0.1})}, E_{(N_0,N_{0.2})}, E_{(N_0,N_{0.3})}$
MR$_2$: *//set $N_{0.3}$ as target*
Query $N_{0.3} -> E_{(N_{0.3},N_{0.3.1})}, E_{(N_{0.3},N_0)}, E_{(N_{0.3},N_{0.3.3})}$
MR$_3$:
Query $N_{0.3.1} -> E_{(N_{0.3.1},N_{0.3.1.1})}, E_{(N_{0.3.1},N_{0.3})}, E_{(N_{0.3.1},N_{0.3.1.3})}$
// $N_{0.3.1}$ cannot be occupied through direct querying
MR$_4$: *//set $N_{0.3.1}$ as target*
Query $N_{0.3.1.1} -> E_{(N_{0.3.1.1},N_{0.3.1.1.1})}, E_{(N_{0.3.1.1},N_{0.3.1.1.2})}, E_{(N_{0.3.1.1},N_{0.3.1.1.3})}$
Query $N_{0.3.1} -> E_{(N_{0.3.1},N_{0.3})}, E_{(N_{0.3.1},N_{0.3.1.3})}, E_{(N_{0.3.1},N_{0.3.1.4})}$
// retrieve a new friend: $N_{0.3.1.4}$
MR$_5$:
Query $N_{0.3.1.4} -> E_{(N_{0.3.1.4},N_{0.3.1})}, E_{(N_{0.3.1.4},N_{0.3.1.4.2})}$, and $E_{(N_{0.3.1.4},N_{0.3.1.4.3})}$
Query $N_{0.3.1.3} -> E_{(N_{0.3.1.3},N_{0.3.1.3.1})}, E_{(N_{0.3.1.3},N_{0.3.1})}$, and $E_{(N_{0.3.1.3},N_{0.3.1.3.3})}$
Query $N_{0.3.1} -> E_{(N_{0.3.1},N_{0.3.1.1})}, E_{(N_{0.3.1},N_{0.3.1.3})}$, and $E_{(N_{0.3.1},N_{0.3.1.4})}$
// $N_{0.3.1}$ is occupied
Query $N_{0.3} ->$ retrieve $E_{(N_{0.3},N_0)}, E_{(N_{0.3},N_{0.3.3})}, E_{(N_{0.3},N_{0.3.4})}$
// retrieve a new friend: $N_{0.3.4}$

MR$_6$:
Query $N_{0.3.4} -> E_{(N_{0.3.4},N_{0.3.4.1})}, E_{(N_{0.3.4},N_{0.3})}, E_{(N_{0.3.4},N_{0.3.4.3})}$
Query $N_{0.3.3} -> E_{(N_{0.3.3},N_{0.3.3.1})}, E_{(N_{0.3.3},N_{0.3})}, E_{(N_{0.3.3},N_{0.3.3.3})}$
Query $N_{0.3} -> E_{(N_{0.3},N_{0.3.1})}, E_{(N_{0.3},N_{0.3.3})}, E_{(N_{0.3},N_{0.3.4})}$
// $N_{0.3}$ is occupied
Query $N_0 -> E_{(N_0,N_{0.1})}, E_{(N_0,N_{0.2})}, E_{(N_0,N_{0.4})}$
Collusion Attack Result: Succeed

In summary, given the victim node N_a as a popular node, none of its top k friends can be occupied through direct queries. Therefore, we propose to occupy at least one of its top k friends through passive displays. We then demonstrate the occupation process through three different scenarios. In all the three scenarios, we observe that whether the target node (i.e. $N_{a.i}$) can be occupied through passive displays depends on whether there exists at least one of its friends in the set $F_{a.i}^{k.1}$ that can be occupied. The procedure of violating a popular node is summarized in procedure 1, shown at the end of the paper.

3) Model Analysis: In this section, we briefly analyze the model performance. Specifically, when randomly given a target node, we would like to evaluate probability of the proposed attack to succeed as well as its complexity in terms of the number of malicious queries. Please note that as the problem is complex due to the heterogeneity of users' friendship, the discussions below are based on some simplified assumptions.

Specifically, when given a node with degree as d, we assume that the probability for one of its friends to be ranked as top k is $\frac{k}{d}$. Accordingly, let us consider a randomly selected victim node N_0 and one of its top k friends (i.e. $N_{0.i}$) with degree as $d_{0.i}$, the probability for N_0 to be on the top k list of $N_{0.i}$ is

$$\begin{cases} \frac{k}{d_{0.i}} & \text{if } d_{0.i} > k \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

which can also be represented as $\min(k/d_{0.i}, 1)$.

According to definition 3, N_0 is a popular node if it is on the top k lists of all its top k friends. Therefore, assuming that the probabilities for N_0 to be one of the top k friends for any two of its friends are independent, the probability for N_0 to be a popular node is

$$P(N_0 \in S_{pop}) = \prod_{i=1}^{k} \min(\frac{k}{d_{0.i}}, 1) \tag{3}$$

Please note that if N_0 is a non-popular node, its privacy can be easily violated at the first layer by using at most $R|_{l=1} = 1+k$ malicious requestors. Therefore, the probability to violate N_0's privacy at the first layer is

$$P(\tilde{N_0}|_{l=1}) = P(N_0 \in S_{non-pop}) = 1 - \prod_{i=1}^{k} \min(\frac{k}{d_{0.i}}, 1) \tag{4}$$

Next, we mainly focus on the case where N_0 is a popular node. Following the procedure to violate a popular node's privacy, we move on the the next layer and focus on N_0's top k friends. Take the first friend N_1 as an example, we assume its top k friends have degrees as $d_{0.1.1}, d_{0.1.2}, d_{0.1.3} \ldots d_{0.1.k}$

Procedure 1 Violation Process for Popular Nodes *bool VioPro(N_p, L, q_list)*

```
1: //$N_p$ is the target node of the upper layer, $L$ represents the layer number, $q\_list$ returns the list of nodes to query in order
2: if $L == 7$ then
3:   return false
4: end if
5: for each top $k$ friend of $N_p$ (i.e. $N_t$) do
6:   set $N_t$ as the target node
7:   query $N_t$ and get its top $k$ friends in set $F_{N_t}$
8:   if $N_p \in F_{N_t}$ and $N_p$'s rank is $r$ then
9:     for each node $f^i_{N_t}$ in $F_{N_t}$ ranked above $N_p$ ($i < r$) do
10:      if $f^i_{N_t}$ cannot passively display $N_t$ then
11:        add $f^i_{N_t}$ to $occupy\_list$
12:      end if
13:    end for
14:    if $occupy\_list$ is not empty then
15:      for each node $f^k_{N_t}$ in $F_{N_t}$ ranked below $N_p$ ($k > r$) do
16:        if $f^k_{N_t}$ can passively display $N_t$ then
17:          add $f^k_{N_t}$ to $q\_list$
18:        else
19:          add $f^k_{N_t}$ to $occupy\_list$
20:        end if
21:      end for
22:      have a new requestor $MR_n$
23:      for each node $N_{occ} \in occupy\_list$ do
24:        use $MR_n$ to query $N_{occ}$ and add it to $q\_list$
25:      end for
26:      use $MR_n$ to query $N_t$ and obtain $F'_{N_t}$
27:      for each node $f'^j_{N_t} \in (F'_{N_t} - F_{N_t})$ do
28:        if $f'^j_{N_t}$ can passively display $N_t$ then
29:          add $f'^j_{N_t}$ to $q\_list$
30:        else
31:          add $f'^j_{N_t}$ to $occupy\_list$
32:        end if
33:      end for
34:      add $N_t$ to $q\_list$
35:      return true
36:    end if
37:  else
38:    add $N_t$ to $q\_lsit$
39:    return true
40:  end if
41: end for
42: for each top $k$ friend of $N_p$ (i.e. $N_t$) do
43:   res = VioPro($N_t$, $L+1$, $q\_list$) //recursive call
44:   if $res == true$ //verify that $N_t$ can be occupied then
45:     assign a new requestor $MR_m$
46:     for each node $N_q$ in $q\_list$ do
47:       query $N_q$
48:     end for
49:     query $N_t$
50:     if $N_t$ is occupied then
51:       return true
52:     end if
53:   end if
54: end for
55: return false
```

and N_0 is ranked as $r_{0.1}$ among these k friends. Then the probability of successfully compromising N_0 through occupying $N_{0.1}$ is

$$P(\tilde{N_0}|_{\tilde{N_{0.1}}}) = 1 - \prod_{i=1}^{r_{0.1}} \min(\frac{k}{d_{0.1.i}}, 1) \tag{5}$$

Applying the same analysis on other top k friends, we can obtain the probability to successfully compromising N_0's privacy at the second layer as

$$P(\tilde{N_0}|_{l=2}) = 1 - \prod_{i=1}^{k} \prod_{j=1}^{r_{0.i}} \min(\frac{k}{d_{0.i.j}}, 1) \tag{6}$$

If N_0 is compromised at the second layer, in the worst case, the total number of malicious requestors required is

$$R|_{l=2} = 1 + k + \sum_{i=1}^{k} r_{0.i} \tag{7}$$

If the malicious requestors launch queries till layer L, the probability of the attack to be succeed is

$$P(\tilde{N_0}|_{l=L}) = 1 - \prod_{i=1}^{k} \prod_{j=1}^{r_{0.i}} \prod_{h=1}^{r_{0.i.j}} \cdots \prod_{w=1}^{r_{0.i.j....v}} \min(\frac{k}{d_{0.i.j.h...v.w}}, 1) \tag{8}$$

where $r_{0.i}$ represents N_0's rank on $N_{0.i}$'s top k list, and $r_{0.i.j}$ represents $N_{0.i}$'s rank on $N_{0.i.j}$'s top k list, etc.

The corresponding total number of malicious requestors required is

$$R|_{l=L} = 1 + k + \sum_{i=1}^{k} r_{0.i} + \sum_{j=1}^{r_{0.i}} \sum_{i=1}^{k} r_{0.i.j} + \cdots + \sum_{r=1}^{r_{0.i.j...v}} r_{0.i.j.h...v.w} \tag{9}$$

In the worst case, when

$$r_{0.i} = r_{0.i.j} = \cdots = r_{0.i.j.h...v.w} = k,$$

the total number of malicious requestors required is

$$R|_{l=L} = \sum_{l=0}^{L} k^l. \tag{10}$$

Based on the above analysis, the success probability of the proposed scheme is closely related with four factors: the degree of the queried nodes d_q, the k value, the total number of layers L and the rank of the friends r.

V. Experiment

In this section, we would like to validate the effectiveness of the proposed collusion attack strategy through experiments. Our experimental study includes an intensive set of experiments on one synthetic data set and three large-scale real-world data sets.

TABLE I
SOCIAL NETWORK DATA SET PROPERTY

	Synthetic	**Facebook**	**Gowalla**	**Slashdot**
Vertices	1000	63731	196591	82168
Edges	10879	817090	582533	950327
Preprocessed Data(V, E)	(1000, 10879)	(63392, 816886)	(196591, 950327)	(82168, 582533)
Average Degree	21.758	25.7725	9.6681	14.1791
Max Degree	178	1098	14730	2554

A. Data Set

We first generate a synthetic data set by following the Barabasi-Albert preferential attachment model, which is a widely adopted model for generating social networks with power law degree distributions [26]. In this model, a graph of n nodes is grown by attaching new nodes each with m edges that are preferentially attached to existing nodes with high degree. In particular, we have set these two parameters as $n = 1000$ and $m = 11$. In addition, we also adopt three real-world social network data sets. All real-world data sets are available at the repository [27], except the Facebook [28] data set.

- Facebook data set [28]: The data was crawled from Facebook.com, capturing the friendships between users, which can be modeled as an undirected graph.
- Slashdot data set [29]: The data contains the friendship/foeship among users on Slashdot. Please note that the original data set does not distinguish friendship from foeship between user pairs. Therefore, the links in the original set are directional. In this work, we convert the Slashdot data set to an undirected graph to reflect user friendship. Specifically, if there is originally one link between two nodes, regardless of its direction, we create an edge between the node pairs in the corresponding undirected graph.
- Gowalla data set [30]: Gowalla is a location-based social networking website where users share their locations by checking in. The data collected from Gowalla represents the friendship network which is undirected.

We pre-process the original data sets by extracting the largest connected component from each of them to ensure the connectivity of all users. In Table I, we list the major features of each data set. As shown in Table I, the four datasets are different in the following aspects. First, they represent different scales of social networks, where the Gowalla dataset includes the maximum number of nodes and the synthetic dataset includes the minimum number of nodes. Second, they represent different distributions of users' relationship degree. In particular, the synthetic dataset generated based on the Barabasi-Albert model well follows the power law degree distributions. Other real world social network datasets, however, reflects more flexible and diverse connections among users. Third, the sparseness of these four datasets is different, where Gowalla is the densest one, and Facebook data is the sparsest one. Correspondingly, the maximum degree of these four datasets is also different, where Gowalla and the synthetic dataset has the largest and smallest maximum degree, respectively.

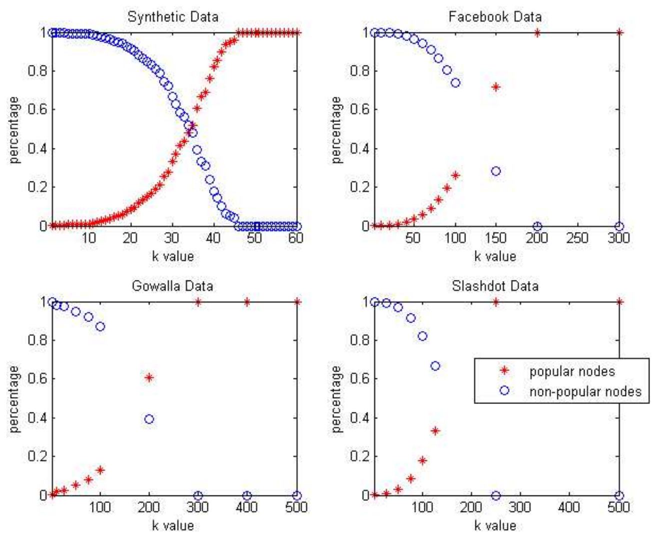

Fig. 6. Percentage of popular/non-popular nodes with degree greater than k.

B. Nodes With Degree Greater Than k

In this section, we analyze the percentage of popular nodes and non-popular nodes with degree greater than k. The reason is twofold. First, in the experiments, the proposed collusion strategy will only be applied on nodes with degree greater than k, as there is no way to find out more than k friends for a node with degree less than or equal to k. Second, the effectiveness of the proposed attack heavily relies on the property of popular nodes, since we have already proved in Section IV-D that privacy of non-popular nodes can always be violated by the proposed collusion strategy.

In particular, we demonstrate how the percentage of higher degree popular/non-popular nodes changes with k value in Figure 6, where the x-axis represents k value and the y-axis represents the node percentage. The red stars and blue circles represent the percentage of popular and non-popular nodes, respectively. From Figure 6, we observe that when the value of k increases, the percentage of popular nodes monotonically increases while that of non-popular nodes monotonically drops. When k exceeds a certain value, which we call the peak k value, the percentage of popular nodes reaches 1, indicating that all the nodes with degree greater than k are popular nodes. The peak k values for the four data sets are 46, 200, 300, and 250 respectively.

C. Comparison Scheme

We also compare the proposed attack to a naive direct query attack, where malicious requetors continuously retrieve new nodes and directly query these nodes to see if they are connected with the victim node. Specifically, we also assume that the attackers only know the victim node before attack. By directly querying the victim node, malicious requestors are able to retrieve k friends of the victim node. To violate

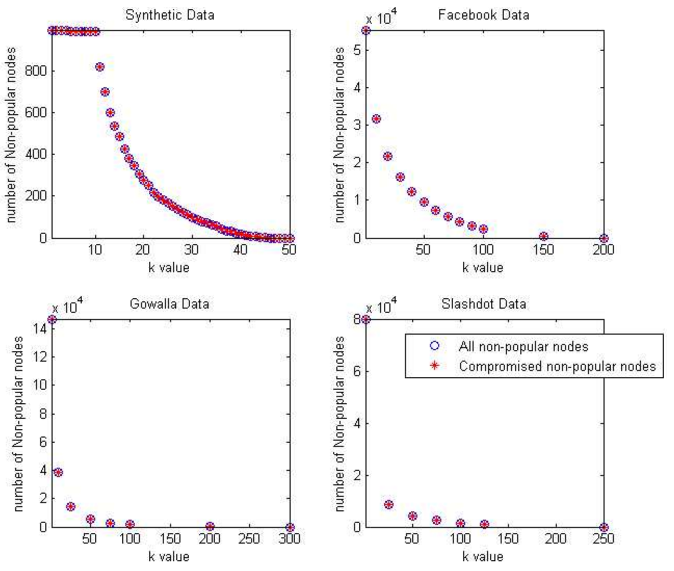

Fig. 7. Proposed attack against non-popular nodes.

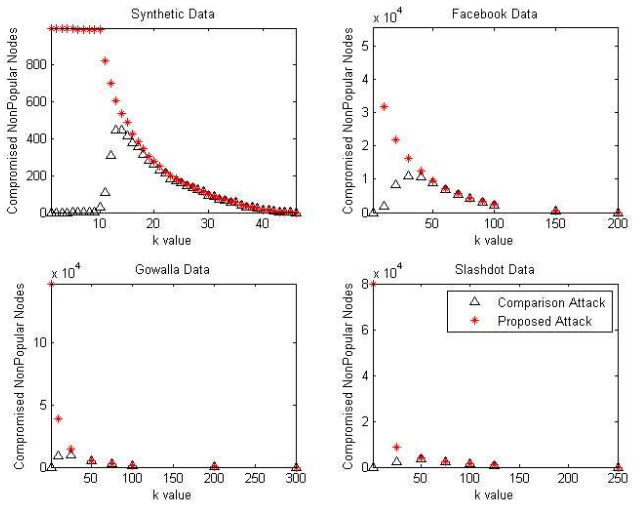

Fig. 8. Attack comparison against non-popular nodes.

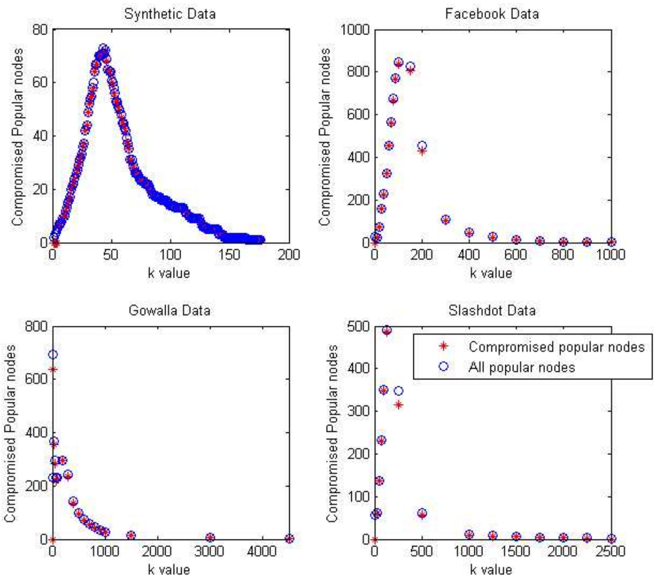

Fig. 9. Proposed attack against popular nodes.

the victim node's privacy (i.e. to find one more friend of the victim node), the direct query attack involves two types of malicious requestors. One is "explorers" who send queries to retrieve more nodes in the OSN, while the other is "violators" who query newly discovered nodes to verify if it is connected with the victim node.

D. Performance Evaluation

In this section, we analyze the effectiveness of launching our proposed attacks against non-popular nodes and popular nodes separately, and compare it to the naive direct attack. In particular, the proposed collusion strategy will only be applied on nodes with degree greater than k, as there is no way to find out more than k friends for a node with degree less than or equal to k.

1) Compromising Non-Popular Nodes: In Figure 7, we demonstrate the effectiveness of the proposed attack against non-popular nodes when k value changes. As the the number of non-popular nodes becomes zero when k exceeds the peak k value, we only demonstrate the k values below the peak value. Specifically, the blue circles and the red stars represent the total number of non-popular nodes, and the number of compromised non-popular nodes, respectively. From this figure, we can make two observations. First, the number of non-popular nodes decreases when k increases. It matches our intuition that when k increases, it is easier for a node to be included in the top k list of its friends. Second, the two charts in each subfigure exactly overlap, validating our discussions in Section IV-D that the proposed attack strategy can ensure to violate privacy of all non-popular nodes.

In addition, we compare the proposed attack to the naive direct attack in Figure 8, where red stars and black triangles represent the number of non-popular nodes compromised by the proposed attack and the naive direct attack, respectively. It is obvious that the proposed attack has done a much better job than the comparison scheme when compromising non-popular nodes. In addition, when the k value is small, the comparison scheme can compromise very few nodes as small k values dramatically restrict the number of friends that can be revealed by the comparison scheme. However, the effectiveness of our proposed attack is not influenced by the small k value.

2) Compromising Popular Nodes: Next, we evaluate the effectiveness of the proposed attack strategy against popular nodes when k value changes in Figure 9. In particular, the blue circles and the red stars represent the total number of popular nodes, and the number of compromised popular nodes, respectively.

From Figure 9, we first make an interesting observation that when k gradually increases, the number of popular nodes increases first and then decreases. The reason is as follows. In the beginning, when k value is very small, such as 10, it is extremely difficult for a node to appear in the top 10 friend list

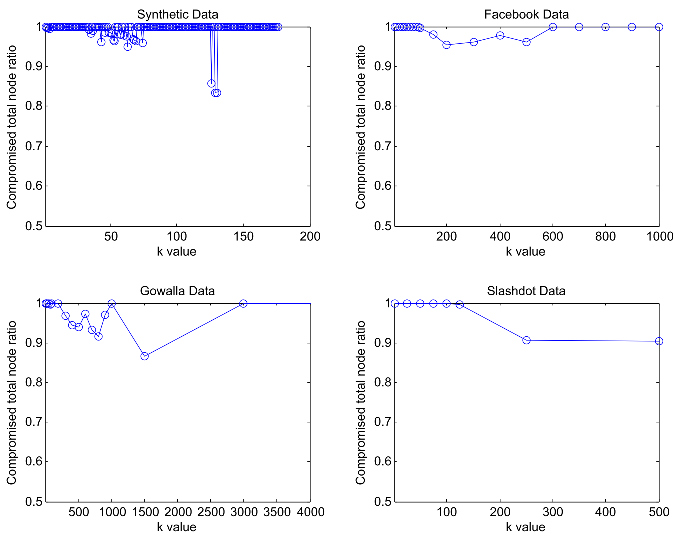

Fig. 10. Percentage of total compromised nodes.

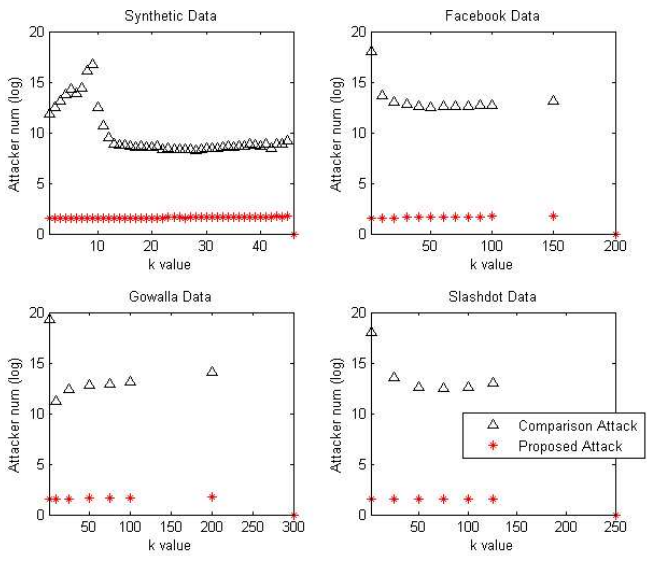

Fig. 11. Average malicious requestor number against non-popular nodes (log).

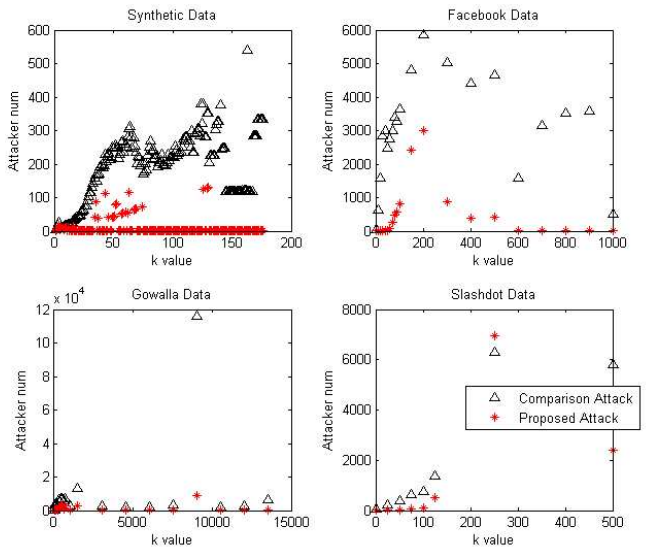

Fig. 12. Average malicious requestor number against popular nodes.

of its top 10 friends, leading to a limited number of popular nodes. When k gradually increases, it is easier for a node to be in the top k list of its top k friends, which leads to an increase of the popular node number. When k continues increasing and becomes very large, as the network follows power law degree distribution, the nodes with degree greater than k become less and less. As a result, the number of popular nodes will also decrease.

In addition, in each subfigure, we can make the following observations. (1) When k value is 1, the number of popular node in different data sets equals to 2, 30, 232, 56, respectively. However, none of them can be compromised by the proposed attack. By further tracking these nodes, we find that these popular nodes actually form pairs, where each node is the top one friend of its pairing node. In other words, these popular nodes form black cliques. As discussed in Section IV-B, their privacy cannot be violated by the proposed collusion strategy. (2) When $k > 1$, the proposed collusion strategy works effectively, as the two charts almost overlap with each other.

To demonstrate the overall performance of the proposed attack, we present the overall percentage of compromised nodes in Figure 10. Specifically, we observe that regardless of the k value, the proposed scheme can always compromise more than 80% of nodes in the network. Please note that for some large k values, there may be a few fluctuations in the compromise ratio. This is mainly caused by the limited number of nodes available. For example, for the Gowalla data, when $k = 1500$, there are only 15 nodes with degree greater than k. Among these nodes, the proposed attack is able to compromise 13 of them, resulting in a compromise ratio as 87%.

In summary, the proposed attack is very effective against both non-popular and popular nodes. In addition, it also outperforms the comparison attack on all types of nodes, especially the non-popular nodes.

3) Number of Required Malicious Requestors: In this section, we investigate the cost of the proposed attack in terms of the average number of required malicious requestors to compromise a node, and compare it to that of the naive direct attack.

We first demonstrate the average number of malicious requestors involved to compromise each non-popular node in Figure 11, where red stars and black triangles represent the proposed attack and the comparison attack, respectively. Please note that as k exceeds the peak k value, there are no more non-popular nodes. Therefore, the average number of required malicious requestor becomes zero. As the malicious requestor numbers differ too much, we take log_2 of the requestor numbers for better illustration.

From Figure 11, we can observe that to violate a non-popular node's privacy, the proposed attack only requires less than 4 malicious requestors regardless of the k value, which is very efficient. However, to compromise one non-popular node, the comparison attack may require upto

107174, 273033, 659232, 266540 (i.e. $2^{16} \sim 2^{20}$) malicious requestors, for the four data sets respectively, significantly worse than the proposed attack.

Next, we demonstrate the average number of malicious requestors involved to compromise a popular node's privacy in Figure 12. Similar to Figure 12, red stars and black triangles represent the proposed attack and the comparison attack, respectively. We can observe that compared to violating privacy of a non-popular node, violating a popular one requires more effort. Nevertheless, the proposed collusion strategy still outperforms the comparison attack by using much less number of malicious requestors.

VI. Conclusion and Discussions

In this paper, we have proposed an advanced collusion attack strategy where multiple attackers with very limited initial knowledge (i.e. only the victim node) can successfully penetrate the defense and violate victim node's privacy settings on friend search engine. In particular, we start this study with a simple and small social clique model, aiming to deeply understand users' friendship types and reveal the fundamental reasons why collusion attacks can be done successfully. Based on observations made from this model, we further propose to classify social network users into non-popular users and popular users; develop different attack strategies against them; and illustrate the attack effectiveness in a general social network through different scenarios. Experiment results show that our proposed collusion attack strategy has achieved high success rate by using limited number of malicious requestors.

Theoretically, this work has simplified the complex privacy attack problem as a small social clique model, classified users' friendship into three categories, and revealed the fundamental reason of the collusion attack's success as the defense scheme's asymetric responses on the symmetric friendship. Furthermore, theoretical model analysis has been conducted on the success probability as well as the total number of malicious requestors needed for the proposed collusion attack when a random victim user is chosen.

Technically, the observations made from this work also shed light on future design of advanced privacy preserving schemes. Specifically, OSNs may utilize the large-large friendship, so that one friendship can only be released in a symmetric way if both friends are on each other's top k lists. It leads to the intuitive idea as that if a node is not so influential, not releasing its friendship with its top influential friend could help protecting the privacy of both nodes. However, such symmetric design may significantly influence the OSNs' sociability as a node's most influential friends may not be displayed when the node itself is not that influential. In addition, as different nodes may set different k values, it is extremely challenging to always find the symmetric k friends to display for all the nodes. Consequently, advanced design is required to better balance friendship privacy and network sociability.

Practically, for OSNs, as people in reality often ignores that their privacy settings may significantly influence their friends' privacy, it is critical to educate people be aware of not only protecting their own information privacy but also be careful when release their friendship to third parties. In addition, as compromising a non-popular node is much easier than compromising a popular one, an individual node may enhance its privacy by making itself a popular node through connecting with more nodes, contributing more to the OSN, or setting an appropriate k value. Specifically, when the k value is too small, to make itself a popular node, the node has to be ranked in the top k lists of all its top k friends, which is very difficult. As the k value increases, it becomes easier for a node to be in the top k lists. However, a too large k value indicates that the node will release many friends anyway, which also hurts its privacy. Therefore, it could be helpful for an individual node to choose its k value based on its own situations.

References

[1] *Friendlist API*. Accessed: 2016. [Online]. Available: https://developers.facebook.com/docs/reference/fql/

[2] J. Bonneau, J. Anderson, F. Stajano, and R. Anderson, "Eight friends are enough: Social graph approximation via public listings," in *Proc. ACM SNS*, 2009, pp. 13–18.

[3] A. Yamada, T. H.-J. Kim, and A. Perrig, "Exploiting privacy policy conflicts in online social networks," CyLab, Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep., Feb. 2012, pp. 1–9.

[4] N. Li, "Privacy-aware display strategy in friend search," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 951–956.

[5] R. Fogues, J. M. Such, A. Espinosa, and A. Garcia-Fornes, "Open challenges in relationship-based privacy mechanisms for social network services," *Int. J. Hum.-Comput. Interact.*, vol. 31, no. 5, pp. 350–370, 2015.

[6] L. Guo, C. Zhang, and Y. Fang, "A trust-based privacy-preserving friend recommendation scheme for online social networks," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 4, pp. 413–427, Jul. 2015.

[7] Z. Feng, H. Tan, and H. Shen, "Relationship privacy protection for mobile social network," in *Proc. Int. Conf. Adv. Cloud Big Data (CBD)*, Aug. 2016, pp. 215–220.

[8] N. Li, N. Zhang, and S. K. Das, "Relationship privacy preservation in publishing online social networks," in *Proc. 3rd IEEE Int. Conf. Social Comput. (SocialCom)*, Oct. 2011, pp. 443–450.

[9] B. Zhou and J. Pei, "Preserving privacy in social networks against neighborhood attacks," in *Proc. 24th IEEE Int. Conf. Data Eng. (ICDE)*, Apr. 2008, pp. 506–515.

[10] J. Cheng, A. W.-C. Fu, and J. Liu, "K-isomorphism: Privacy preserving network publication against structural attacks," in *Proc. SIGMOD*, 2010, pp. 459–470.

[11] S. Das, O. Egecioglu, and A. El Abbadi, "Anonymizing edge-weighted social network graphs," Dept. Comput. Sci., Univ. California, Santa Barbara, Santa Barbara, CA, USA, Tech. Rep. CS-2009-03, 2009.

[12] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis, "Resisting structural re-identification in anonymized social networks," *Proc. VLDB Endowment*, vol. 1, no. 1, pp. 102–114, 2008.

[13] B. Zhou, J. Pei, and W. Luk, "A brief survey on anonymization techniques for privacy preserving publishing of social network data," *ACM SIGKDD Explorations Newslett.*, vol. 10, no. 2, pp. 12–22, Dec. 2008.

[14] E. Zheleva and L. Getoor, "Preserving the privacy of sensitive relationships in graph data," in *Proc. 1st ACM SIGKDD Int. Workshop Privacy, Secur., Trust KDD (PinKDD)*, San Jose, CA, USA, 2007, pp. 153–171.

[15] L. Liu, J. Wang, J. Liu, and J. Zhang, "Privacy preserving in social networks against sensitive edge disclosure," Dept. Comput. Sci., Univ. Kentucky, Lexington, KY, USA, Tech. Rep. CMIDA-HiPSCCS 006-08, 2008.

[16] L. Zou, L. Chen, and M. T. Özsu, "K-automorphism: A general framework for privacy preserving network publication," *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 946–957, Aug. 2009.

[17] K. Liu, K. Das, T. Grandison, and H. Kargupta, "Privacy-preserving data analysis on graphs and social networks," in *Next Generation of Data Mining*, H. Kargupta, J. Han, P. Yu, R. Motwani, and V. Kumar, Eds. Boca Raton, FL, USA: CRC Press, 2008.

[18] *13 Million U.S. Facebook Users Don't Change Privacy Settings*. Accessed: May 3, 2012. [Online]. Available: https://www.zdnet.com/article/13-million-usfacebook-users-dont-change-privacy-settings/

[19] *Social Capital*. Accessed: 2018. [Online]. Available: https://en.wikipedia.org/wiki/Social_capital

[20] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, “SybilGuard: Defending against sybil attacks via social networks,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 267–278, 2006.

[21] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, “SybilLimit: A near-optimal social network defense against sybil attacks,” in *Proc. IEEE Symp. Secur. Privacy*, May 2008, pp. 3–17.

[22] H. Yu, C. Shi, M. Kaminsky, P. B. Gibbons, and F. Xiao, “DSybil: Optimal sybil-resistance for recommendation systems,” in *Proc. IEEE Symp. Secur. Privacy*, May 2009, pp. 283–298.

[23] G. Danezis and P. Mittal, “Sybilinfer: Detecting sybil nodes using social networks,” in *Proc. NDSS*, 2009, pp. 1–15.

[24] J. Sun, X. Zhu, and Y. Fang, “A privacy-preserving scheme for online social networks with efficient revocation,” in *Proc. 29th Conf. Inf. Commun.*, Mar. 2010, pp. 2516–2524.

[25] R. D. Alba, “A graph-theoretic definition of a sociometric clique,” *J. Math. Sociol.*, vol. 3, no. 2, pp. 113–126, 1973.

[26] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[27] *Stanford Large Network Dataset Collection*. [Online]. Available: http://snap.stanford.edu/data/

[28] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, “On the evolution of user interaction in Facebook,” in *Proc. 2nd ACM Workshop Online Social Netw.*, 2009, pp. 37–42.

[29] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, “Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters,” *Internet Math.*, vol. 6, no. 1, pp. 29–123, 2009.

[30] E. Cho, S. A. Myers, and J. Leskovec, “Friendship and mobility: User movement in location-based social networks,” in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 1082–1090.

Yuhong Liu received the B.S. and M.S. degrees from the Beijing University of Posts and Telecommunications in 2004 and 2007, respectively, and the Ph.D. degree from the University of Rhode Island in 2012. She is currently an Assistant Professor with the Department of Computer Engineering, Santa Clara University. With expertise in trustworthy computing and cyber security, her research interests include developing trust models and applying them on emerging applications, such as online social media, cyber-physical systems, and cloud computing. She was a recipient of the 2013 University of Rhode Island Graduate School Excellence in Doctoral Research Award and the Best Paper Award at the 9th International Conference on Ubi-Media Computing (UMEDIA 2016). Her work on securing online reputation systems received the Best Paper Award at the IEEE International Conference on Social Computing 2010 (acceptance rate = 13%).

Na Li received the Ph.D. degree in computer science from the University of Texas at Arlington in 2012. She has been an Assistant Professor with Prairie View A&M University (PVAMU) since fall 2015. Prior to PVAMU, she was an Assistant Professor with the Northwest Missouri State University. She has been dedicated to cybersecurity research and education. Particularly, her interests include but are not limited to security and privacy in wireless sensor networks, online social networks, and Internet of Things. She is also interested in computer science education, particularly focusing on underrepresented minorities. Her projects have been funded by the National Security Agency and the National Science Foundation.