Hindawi
Wireless Communications and Mobile Computing
Volume 2017, Article ID 1393026, 9 pages
https://doi.org/10.1155/2017/1393026

Research Article

An Advanced Private Social Activity Invitation Framework with Friendship Protection

Weitian Tong,[1] Lei Chen,[2] Scott Buglass,[1] Weinan Gao,[3] and Jeffrey Li[1]

[1]*Department of Computer Sciences, Georgia Southern University, Statesboro, GA 30460, USA*
[2]*Department of Information Technology, Georgia Southern University, Statesboro, GA 30460, USA*
[3]*Department of Electrical Engineering, Georgia Southern University, Statesboro, GA 30460, USA*

Correspondence should be addressed to Weitian Tong; wtong@georgiasouthern.edu

Received 31 August 2017; Accepted 25 October 2017; Published 16 November 2017

Academic Editor: Chaokun Wang

Due to the popularity of social networks and human-carried/human-affiliated devices with sensing abilities, like smartphones and smart wearable devices, a novel application was necessitated recently to organize group activities by learning historical data gathered from smart devices and choosing invitees carefully based on their personal interests. We proposed a private and efficient social activity invitation framework. Our main contributions are (1) defining a novel friendship to reduce the communication/update cost within the social network and enhance the privacy guarantee at the same time; (2) designing a strong privacy-preserving algorithm for graph publication, which addresses an open concern proposed recently; (3) presenting an efficient invitee-selection algorithm, which outperforms the existing ones. Our simulation results show that the proposed framework has good performance. In our framework, the server is assumed to be untrustworthy but can nonetheless help users organize group activities intelligently and efficiently. Moreover, the new definition of the friendship allows the social network to be described by a directed graph. To the best of our knowledge, it is the first work to publish a directed graph in a differentially private manner with an untrustworthy server.

1. Introduction

Nowadays, social networks are pervading our lives in nearly every possible form and corner [1–7], as people use them to connect, interact, and share with their peers. In particular, the ubiquity of smart phones and various social network applications have made the global social network flourish over recent years. One common and critical service provided by social networks is organizing group activities. Unfortunately, most social networks offer only rudimentary invitation mechanisms, which send invitations either one-by-one manually or to everyone automatically. Besides, most group activities are filled strictly with a first-come, first-served manner. These services are ill-suited for frequent, small ad hoc events such as outdoor activities: inviting every possible candidate increases the likelihood of a group where few people know anybody else except for the host; however, it is tedious to manually search for a well-acquainted social group that performs the same kinds of exercise, at the same time and place [8]. From the invitees' perspective, they might be overwhelmed by a plethora of different activity invitations that they are not willing to attend since invitations are typically sent out without considering the real interest, ability, and social habit of each invitee.

The popularity of human-carried/human-affiliated devices with sensing abilities, like smartphones and smart wearable devices, has opened up a large resource for sensory data, which has necessitated many novel sophisticated applications. For example, smart watches are usually equipped with an array of different sensors such as compasses, proximity sensors, accelerometers, gyroscopes, altimeters, barometers, and GPS [9]. These can be used to collect various data such as location, route, distance, pace/speed, duration, and elevation changes for different activities attended by the owner. By analyzing these personal data with state-of-the-art mining or learning algorithms, the habits of the device owners, including their preferred activities, schedule, and location, can be easily derived. This

habit information can, in turn, be used to help the owners find group activities appropriate for them.

Based on this observation, Ai et al. [10] first proposed an efficient and personalized group activity organizing framework by learning historical data gathered from smart devices and choosing invitees carefully for an activity. However, they did not consider the risk of the privacy leakage of participants' sensitive information, such as habits, age, and gender. Later, Tong et al. [8] designed a private group activity organizing framework and proposed the adoption of differential privacy to secure participants' personal information. Tong et al. [8] considered a practical scenario, where three parties, including an untrustworthy activity organizer app, current app users, and potential users, are involved. After registering on this app, users can either organize activities by submitting a request to the server or receive invitations from the app server. Users have the capability of adding each other as friends. In order to receive more interesting invitations, app users need to divulge personal information such as age, gender, locational preferences, and historical data from their wearable devices. In particular, Tong et al. [8] assumed that the activity organizer app is untrustworthy, mainly due to the reason that the app developers are motivated by advertising revenue therefore attempting to attract more users by releasing some useful information about current users. The main contribution of Tong et al.'s work [8] is to protect existing users' privacy while satisfying all three parties involved. The primary drawback, however, is that it allows the entire social network to be released to the public after naive sanitization approaches like removing user IDs. This may leave users open to privacy risks, especially reidentification attacks [11, 12].

In our work, based on the same three-party scenario assumption by Tong et al. [8], we designed a new group activity organizing framework with a stronger privacy guarantee and a more efficient invitee-selection algorithm. More precisely, our contributions in this research can be summarized as follows.

(1) *A novel definition of friendship*: we introduced a more flexible definition of the friendship between a pair of users, which asks user "Who do you like doing activities with?" instead of "Who is your friend?" In previous works [8, 10], the friendship is defined mutual. However, a person could enjoy doing activities with another person without having the other person reciprocate the same feeling. This makes sense for the event invitation framework since its purpose is not to keep track of actual mutual friendships, but which users enjoy doing activities with whom. A more accurate term for this relationship would be *"preferred friend"* or *"directed friend."*

Such "directed" friendship notion brings several benefits. First of all, such friendships can be described easily by a directed graph $G = (V, A)$, where the vertex set V represents the user set and the arc set A shows the corresponding directed friendships. That is, if v likes doing activities with u, then $(v, u) \in A$. Second, there is no need for other users to accept a friendship request, meaning two users do not have to directly communicate or have mutual agreement on friendship. This relieves the workload of updating the social network. Last but not least, since friendships are not bidirectional, having one user's report does not compromise information about the remaining users. In other words, such friendships enhance the privacy protection for the users.

(2) *Stronger privacy guarantee*: we added an efficient algorithm to make the graph satisfy a strong privacy guarantee, differential privacy, and thus allow the app server to release the underlying graph of the entire social network without jeopardizing users' privacy. Differential privacy requires no computational/informational assumptions about attackers, data type-agnosticity, composability, and so on [13]. Since the app server is untrustworthy, we need to hide structure information before it is uploaded to the server. We applied the *Randomized Response Technique* (RRT) [14] to all vertices (or users). That is, each user's friendships will be perturbed before being reported to the server. For example, a user will report the true (fake, resp.) relationship with a probability $1 - p$ (p, resp.), where the parameter $p \in (0, 1]$ is usually a small number. Such a randomized response strategy ensures the existence of connection from one user to the other to be hidden in the output graph while keeping the low distortion of the graph and preserving the most useful information about the graph. To the best of our knowledge, this is the pioneer work that this technique is applied in a directed graph under the existence of an untrusted server.

(3) *A more efficient invitation sending mechanism*: in order to select appropriate candidates as invitees, Ai et al. [10] proposed a greedy algorithm, K-CORE, based on the k-core (undirected) graph theory. Our work designed a novel greedy algorithm, named as *advanced k-core* (ADV-K-CORE), to improve the K-CORE algorithm. The K-CORE algorithm starts with the original graph, sets $k = 1$, and then iteratively deletes all vertices with a degree less than k in the current graph. k gradually increases and the algorithm terminates when the size of the remaining graph reaches a lower bound. Our ADV-K-CORE deletes vertices more carefully by assigning higher priority to the vertex with the least impact on other vertices.

(4) *Experimental validation*: in order to evaluate the performance of our activity invitation framework, we simulated an outdoor activity invitation system, where at most 1,000 users are created with different profiles, including age, gender, free time schedules, activity types, activity levels, and locational ranges. Then, at most 5,000 different activity events are generated, each of which requires a specific age range, time range, activity type, activity level, and location. Our experiments show that the privacy-preserving algorithm protects the structure of the social network

effectively and the Adv-k-core algorithm improves the original k-core algorithm extensively.

The rest of the paper is organized as follows. Section 2 reviews related works; the proposed framework is introduced in Section 3; Section 4 shows the simulation results; and Section 5 concludes our paper.

2. Related Work

Organizing group activities via social media, such as *Facebook*, *Twitter*, *Plancast*, *Meetup*, *Yahoo! Upcoming*, and *Eventbrite*, are quite popular in the era of "Internet of Everything." However, most of these social media offer only rudimentary functions for organizing group activities [8]. Take *Facebook* as an example; it allows users to create public or private events, but the organizer can only choose to send invitations one-by-one or to everyone.

There is plenty of research in the literature on social networks; the following two are the ones most related to our work. Ai et al. [10] first made the proposal to design the social event invitation framework based on historical data of smart devices. They also presented two greedy invitation-disseminating algorithms. Their framework, however, is impractical as it assumes the existence of a trusted and altruistic server. Besides, few privacy protection approaches were applied to guarantee the security or confidentiality of users' personal information. Recently, Tong et al. [8] considered a more realistic scenario in which the server is selfish and possibly untrustworthy. They concentrated more on the privacy issue such that existing users will be sufficiently protected while satisfying all involved parties simultaneously. Nevertheless, Tong et al. [8] only protected personal data such as age, gender, free time schedules, activity types, activity levels, and locational ranges, while leaving the underlying graph structure of the entire social network open to privacy risks, especially reidentification attacks [11, 12].

Differential privacy [14–18] is a strictly provable and security-controlled privacy model to provide a very strong privacy guarantee. It can quantify the extent to which individuals' privacy in a data set is preserved, while maintaining the usefulness of the data set. Differential privacy has proven to be extremely successful since its inception. The most popular differential privacy mechanisms include the Laplace mechanism [14], exponential mechanism [19], geometric mechanism [20], and Gaussian mechanism [17, 21].

The problem of graph publication under differential privacy has been well investigated. Generally speaking, there are two main techniques: *direct publication* and *model-based publication*. By direct publication, the output graph is constructed by directly adding noise to each edge or vertex, followed by a postprocessing step (probably a rounding step). For example, given an undirected graph and assuming edges are independent, adding Laplace noise to each cell of the adjacency matrix and then rounding each cell to 1's or 0's is a trivial Laplace mechanism to preserve the privacy. However, such an approach may severely deteriorate the graph structure. Recently, there are two differential privacy algorithms, TmF [22] and EdgeFlip [23], in this category for undirected graph publication. The algorithms for model-based publication inject noise to some intermediary quantities or structures, such as graph spectral, instead of directly to the original graph. The output graph will be regenerated from these noisy intermediary structures. Popular algorithms in this category include 1K-series, 2K-series [24, 25], Kronecker graph model [13], graph spectral analysis [26], DER [27], HRG-MCMC [28], and ERGM [29]. Most existing privacy-preserving algorithms for graph publication assume the graph is undirected and published by a trusted and altruistic server.

3. Privacy-Enhanced Activity Invitation Framework

In this section, we introduce our novel privacy-enhanced activity invitation framework (refer to Figure 1). Following Tong et al.'s [8] design, our framework also involves three parties: a central server controlled by the app developers, the existing app users, and potential new members. Compared with Tong et al.'s [8] framework, our framework enhances the users' privacy by defining a "directed" friendship and protecting the underlying graph structure of the social network under the differential privacy model. Furthermore, our framework employs a novel and significantly more effective invitation-disseminating algorithm.

As introduced, we make a realistic assumption that the server is untrustworthy, given that it is motivated by advertising to its existing users and gaining profits. In order to bolster its income, the server will strive to provide quality services to maintain current members and also try to entice new users by releasing some statistical information about current users and providing online querying services. As a result, existing users or new registers may have trouble deciding whether to report their personal information honestly, including age, gender, and "Who I like doing activities with." On the one hand, the server will definitely learn users' habits more accurately if users could provide candid information, which in turn leads to better services. On the other hand, users should be worried by the possibility of having their personal information leaked.

The following shows how our design works in detail. Once a person registers on the app, the server will create and maintain a profile for him/her until he/she wants to destroy the account. If the user is a smart wearable device owner, the front-end app will seek authorization to access his/her historical data which contains records pertaining to activities. Otherwise, users need to fill their own profiles manually based on their understanding and estimation of their abilities. Whenever a user needs to update or report his/her personal information to the server, the front-end, user-side app will automatically obfuscate the given personal information before being transferred to the server so that the information is protected by differential privacy. If a user wants to organize an activity, a request will be first sent to the server. Then the server will analyze users' historical data and estimate the users' abilities or levels for each type of activity; the routine times they are free; and a locational range, indicating the rough area in which he/she is willing or able

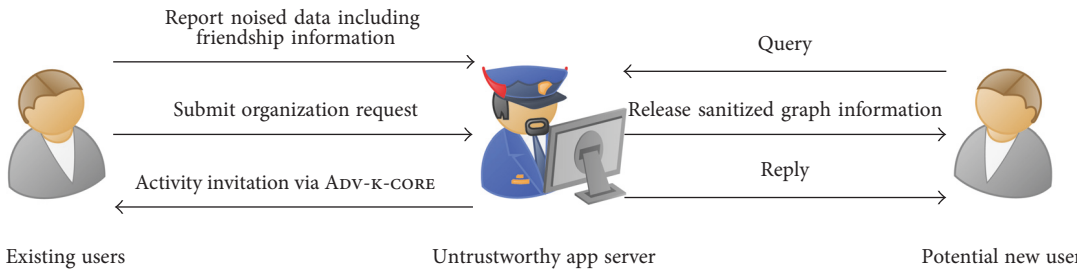

Figure 1: Our privacy-enhanced activity invitation framework.

to travel in order to participate in the activity. Based on the above estimated habits about existing users, the server will disseminate the invitations to appropriate candidates via the Adv-k-core algorithm such that all of the invitees meet the group activity requirements and have a high chance to attend the activities. Since any privacy-preserving algorithm that satisfies differential privacy will protect the individual's information regardless of the adversary's background information [13], the server can release the statistical information about the current users safely to the public.

Our framework does not need to keep track of actual mutual friendships, but which users enjoy doing activities with whom. To depict such relationship among the users, we first define the concept of *directed friendship* and then use a directed graph to simulate the entire social network.

Definition 1 (directed friendship). For any two users A and B, if A likes attending activities together with B, one says B is A's directed friend.

While the traditional friendship is a symmetric relation, our definition implies an asymmetric relation between users. Let $G = (V, A)$ represent the underlying directed graph, where a vertex $v \in V$ denotes a user. An arc from v to u means that user v likes attending activities together with u. Such a definition allows each user to update his/her neighbors independently, which not only reduces workload but also enhances the privacy guarantee for the users.

3.1. Graph Publication via Differential Privacy

3.1.1. Preliminary. Differential privacy [14, 16, 17] is a privacy model that offers strong privacy guarantees under the assumption of a powerful adversary. In particular, the adversary could have nearly unlimited background knowledge. The model works by injecting artificial noise to the disclosed data set such that no one can tell whether an entry in the data set has been changed or not. On the other hand, differential privacy guarantees the released information is still useful. Formally, given two datasets where only one entry is altered, the probability distribution of the outputs for a statistical analysis of one data set should be nearly identical to the distribution of the other's.

Let $\mathbf{x} \in \mathcal{X}^n$ and $\mathbf{x}' \in \mathcal{X}^n$ be two data sets. The *distance between the two datasets*, denoted as $d(\mathbf{x}, \mathbf{x}')$, is the minimum number of sample changes that are required to change $\mathbf{x}$ into $\mathbf{x}'$. If $d(\mathbf{x}, \mathbf{x}') = 1$, that is, if $\mathbf{x}$ and $\mathbf{x}'$ differ by at most one entry, then we say that $\mathbf{x}$ and $\mathbf{x}'$ are *neighbors*.

Definition 2 (edge-neighboring graphs). One says two directed graphs $G_1 = (V_1, A_1)$ and $G_2 = (V_2, A_2)$ are *edge-neighboring graphs* if $V_1 = V_2$, $A_1 \subset A_2$, $|A_2| = |A_1| + 1$.

Definition 3 (vertex-neighboring graphs). One says two directed graphs $G_1 = (V_1, A_1)$ and $G_2 = (V_2, A_2)$ are *vertex-neighboring graphs* if $V_1 = V_2 - \{v\}$, $A_1 \subset A_2$, $A_2 \backslash A_1 \subset N(v)$. Here $N(v)$ denotes the set of incident incoming and outgoing arcs on v.

A *query* f is a function whose domain is the collection of data sets. The output of the query f is usually denoted as $f(\mathbf{x})$. The *global sensitivity* Δf of the given query f is defined as

$$\Delta f = \max_{d(\mathbf{x},\mathbf{x}')=1} \left\| f(\mathbf{x}) - f\left(\mathbf{x}'\right) \right\|, \tag{1}$$

where $\| \cdot \|$ is a norm function. Our proposed framework is trying to hide the true friendship information for each user against queries like "how many neighbors does a user have?" It is not difficult to check that the sensitivity is 1 under the edge-neighboring notion and at most $n - 1$ in the worst case under the vertex-neighboring notion. We adopt the edge-neighboring notion in our work for the sake of low sensitivity.

Definition 4 (ϵ-differential privacy [14, 30]). A mechanism or randomized function $\mathcal{M} : \mathcal{X}^n \to \mathcal{R}$ provides *ϵ-differential privacy if and only if* for all pairs of neighboring data sets $\mathbf{x}$ and $\mathbf{x}'$, and all subset $S \subset \text{Range}(\mathcal{M})$, it holds that

$$\Pr\left[\mathcal{M}(\mathbf{x}) \in S\right] \leq e^{\epsilon} \Pr\left[\mathcal{M}\left(\mathbf{x}'\right) \in S\right]. \tag{2}$$

The parameter ϵ, deemed *privacy budget*, controls the level of privacy. Usually, the value of ϵ is small; say $\epsilon \in (0, 1]$. Intuitively speaking, the parameter ϵ gives the upper bound on the output difference when the mechanism is applied to a data set and any one of its neighbors. From inequality (2), $\Pr[\mathcal{M}(\mathbf{x}) \in S]$ and $\Pr[\mathcal{M}(\mathbf{x}') \in S]$ become closer when ϵ decreases, implying more effort to distinguish the neighboring data sets and therefore indicating a stronger privacy guarantee.

The *Laplacian mechanism* [17] and *exponential mechanism* [19] are two of the most popular ϵ-differentially private mechanisms. Generally speaking, the Laplace mechanism is typically used when the output is numerical, whereas the exponential mechanism is applied to nonnumerical outputs. In particular, the exponential mechanism is more suited for situations where we need to select the "optimal" response but adding noise directly to $f(\mathbf{x})$ can completely destroy its value.

Definition 5 (Laplacian mechanism [17]). Given a query $f : \mathcal{X}^n \to \mathbb{R}^k$, the Laplacian mechanism is defined as

$$\mathcal{M}_L(\mathbf{x}) = f(\mathbf{x}) + (Y_1, \ldots, Y_k), \tag{3}$$

where Y_i are *i.i.d. (independent and identically distributed)* random variables drawn from $\text{Lap}(\Delta f/\epsilon)$. Here, $\text{Lap}(b)$ denotes a *Laplace distribution* (centered at 0) with scale b and its probability density function is $\text{Lap}(x \mid b) = (1/2b)\exp(-|x|/b)$.

Definition 6 (exponential mechanism [19]). The exponential mechanism $\mathcal{M}_E$ selects and outputs an element $r \in \text{Range}(f)$ with probability proportional to $\exp(\epsilon u(\mathbf{x}, r)/2\Delta u)$, where

$$u : \mathcal{X}^n \times \text{Range}(f) \longrightarrow \mathbb{R} \tag{4}$$

is a *utility function* that maps data set/output pairs to utility scores, and the sensitivity of u is defined as

$$\Delta u = \max_{r \in \text{Range}(f)} \max_{d(\mathbf{x},\mathbf{x}')=1} \left\| u(\mathbf{x}, r) - u(\mathbf{x}', r) \right\|. \tag{5}$$

3.1.2. Our Differential Privacy Mechanism. There are two main types of noise injection strategies: *output perturbation* and *input perturbation*. Namely, the ϵ-differentially private mechanisms are usually designed by either perturbing the output of the query or adding noise to the input data set. Obviously, the output perturbation requires a trusted server to hold the authentic data sets while the input perturbation is more flexible as the data can be perturbed before being transferred to the server. Our framework assumes an untrustworthy server, and therefore an input perturbation strategy will be adopted.

Both the *Laplacian* and *exponential* mechanisms mentioned in Section 3.1.1 can be modified to perturb the input rather than output. These two mechanisms can be applied to obfuscate different types of users' raw data, such as age, activity types, or activity ranges [8]. Since our work concentrates on the protection of users' friendships, we add a novel privacy-preserving mechanism, named as Pert, in the random response manner (refer to Algorithm 1). More precisely, each user reports his/her real friendship information with a probability $1 - p$, where $p \in (0, 1]$. The larger p is, the more arcs in the graph are randomized.

Theorem 7. *Our graph perturbation algorithm Pert guarantees ϵ-differential privacy.*

Proof. Suppose $G_1 = (V, A_1)$ and $G_2 = (V, A_2)$ are two edge-neighboring graphs. Assume $A_2 = A_1 \cup \{(v, u)\}$. Let G_1' and G_2' represent the perturbed version of G_1 and G_2, respectively. Note that V represents the same set of users in both graphs. The probability that two edge-neighboring graphs are perturbed to the same graph is determined by the value assigned to the differing arc (v, u). According to the algorithm Pert, an arc in the input graph maintains its original value with a probability $1 - p$ and flips its value with a probability p. For any G', depending on whether $(v, u) \in G'$, we have

$$\begin{aligned}
&\frac{\Pr\{G_1' = G'\}}{\Pr\{G_2' = G'\}} \\
&= \begin{cases} \dfrac{\Pr\{G_1' = G' \mid (v,u) \in G'\}}{\Pr\{G_2' = G' \mid (v,u) \in G'\}}, & (v,u) \in G' \\ \dfrac{\Pr\{G_1' = G' \mid (v,u) \notin G'\}}{\Pr\{G_2' = G' \mid (v,u) \notin G'\}}, & (v,u) \notin G' \end{cases} \\
&= \begin{cases} \dfrac{p}{1-p}, & (v,u) \in G' \\ \dfrac{1-p}{p}, & (v,u) \notin G' \end{cases} \\
&\le e^{\epsilon},
\end{aligned} \tag{6}$$

where the last inequality is due to the value of $p = e^{\epsilon}/(1 + e^{\epsilon})$. This proves the theorem according to the definition of ϵ-differential privacy. □

Theorem 8 (composition theorem [30]). *Let $\mathcal{M}_i$, $i \in \{1, 2, \ldots, n\}$, be ϵ_i-differentially private algorithms. Suppose*

$$\mathcal{M}_{[n]}(\mathbf{x}) = (\mathcal{M}_1(\mathbf{x}), \mathcal{M}_2(\mathbf{x}), \ldots, \mathcal{M}_n(\mathbf{x})) \tag{7}$$

is the combination of these n algorithms.

(i) *If all $\mathcal{M}_i$ are defined on the same data set, then $\mathcal{M}_{[n]}$ is $(\sum_{i=1}^n \epsilon_i)$-differentially private.*

(ii) *If all $\mathcal{M}_i$ are defined on different data sets, then $\mathcal{M}_{[n]}$ is $(\max\{\epsilon_i\})$-differentially private.*

According to the Composition Theorem, combining several differentially private algorithms results in a new differentially private algorithm at a cost of linearly increasing privacy budget in the worst case. For each user, his/her profile can be described by a tuple, where each dimension represents one type of data. Injecting noises to different data field with different ϵ_i-differentially private algorithms $\mathcal{M}_1, \ldots, \mathcal{M}_n$, $(\max\{\epsilon_i\})$-differential privacy will be guaranteed if data fields are independent; otherwise, $(\sum_{i=1}^n \epsilon_i)$-differential privacy will be guaranteed.

3.2. Improved k-Core Algorithm. The server's main job is to select invitees to meet the request of organizing a group activity from some user. Following Ai et al. [10] and Tong et al. [8], we assume that having friends attend an activity

Input: A directed graph $G = (V, A)$ and $p = e^{\varepsilon}/(1 + e^{\varepsilon})$
Output: A perturbed graph $G' = (V, A')$
(1) Let $A' = \emptyset$
(2) **for** each $v \in V$ **do**
(3) **if** $(v, u) \in A$ **then**
(4) Add (v, u) to A' with probability $1 - p$
(5) **else**
(6) Add (v, u) to A' with probability p
(7) **return** the resultant graph $G' = (V, A')$

Algorithm 1: Graph perturbation algorithm Pert.

Input: A directed k-core graph H and a group size m
Output: A list L of invitees
(1) Let $L = V(H)$
(2) Let $k = 1$
(3) **while** $|L| > m$ **do**
(4) **if** $v \in V(H)$ such that $|N^+(v)| \leq k$ **then**
(5) Pick the v such that after its deletion
(6) resulting in miminum number of vertices
(7) with degree $\leq k$
(8) Delete v
(9) **else**
(10) $k = k + 1$
(11) **return** the remaining list L

Algorithm 2: Improved k-core algorithm Adv-k-core.

will improve participants' overall experience. Therefore, the server needs to ensure that a number of friends will also be invited for each invitee.

We adopt the concept of k-core graph to simulate a qualified social network where each user has at least k friends. Suppose H is a subgraph of G such that users in H satisfy all the requirements for an activity. Let $V(H)$ and $A(H)$ denote the vertex and arc set of H, respectively. We say H is a k-core graph if each vertex $v \in V(H)$ has at least k directed friends. Let $N_H^+(v) = \{u \mid \exists(v, u) \in A(H)\}$ be the set of neighbors of v in graph H, and let $|N_H^+(v)|$ denote its cardinality, or the degree of v in H. Suppose a group activity has a limited capacity m, and r is the statistical response rate for similar past activities. The task then becomes choosing $m + m/r$ invitees such that each person also has k friends invited.

Ai et al. [10] presented a greedy invitee-selection algorithm, k-core. The k-core algorithm starts with the original graph and sets $k = 1$; and then it iteratively deletes all vertices with a degree less than k in the current graph. When deleting the vertices, the highest priority will be assigned to the vertex with the minimum degree. As k gradually increases, the algorithm terminates when the size of the remaining graph is $(m + m/r)$. We propose an improved k-core algorithm, denoted as Adv-k-core (refer to Algorithm 2). Adv-k-core works very similar to k-core with the exception of the vertex deletion step. We scan through the whole graph and find the vertex with the least impact on other vertices, in respect to the number of vertices with degree less than current k by the deletion.

4. Experiments

Two experiments were designed to evaluate the performance of our activity invitation framework. In these experiments, an outdoor activity invitation system is simulated, where at most 1000 users are created with different profiles, including friendships, age, gender, free time schedules, activity types, activity levels, and locational ranges. Then, at most 5,000 different activity events are generated, each of which requires a specific age range, time range, activity type, activity level, and location. As previously mentioned, each participant must satisfy all of the event's requirements. A random response rate $r \in [0.6, 1)$ is generated uniformly for each user in advance. When a user receives an invitation, another random number $\text{re} \in [0, 1)$ is generated. If $\text{re} < r$, he/she accepts the invitation; otherwise, there will be no response. All experiments were implemented with Java and conducted under OS X EL Capitan with processor, 3.5 GHz Intel Core i5, and memory, 16 GB 1600 MHz DDR3.

4.1. Experiment 1. As shown in Section 3.1, users' sensitive information has been theoretically secured by our differential privacy algorithms. In particular, the graph structure of the social network can be protected by the algorithm Pert. Since the algorithm Pert hides users' friendship by perturbing the arcs, the graph structure can be changed, which might affect users' usage experience. For example, suppose a user originally has 5 friends in the social network and the number may decrease to 0 after the Pert algorithm is applied, which excludes this user from the invitee pool.

Our first experiment is to investigate whether existing users will receive worse services if they report noisy friendships to the server. We define the utility for each existing user as the ratio of accepted invitations in the original graph to the number of accepted invitations in the perturbed graph. Denote this ratio by γ. That is,

$$\gamma = \frac{\sharp \text{ of accepted invitations in } G}{\sharp \text{ of accepted invitations in } G'}. \tag{8}$$

The quantity $\gamma_0 = |\gamma - 1|$ tending to 0 indicates that our framework could still provide qualified servers to existing users despite users reporting noisy information to the server. For simplicity, we still name γ_0 as the utility.

In this experiment, we set privacy budget $\epsilon \in \{0.05 : 0.05 : 1\}$, where the notation $\{\ell : \Delta : u\}$ denotes an arithmetic sequence of numbers with lower bound ℓ, upper bound u, and constant difference Δ between the consecutive terms. For example, $\{1 : 2 : 10\} = \{1, 3, 5, 7, 9\}$. To figure out how the utility γ_0 behaves as the privacy budget varies, the average utility γ_0 was calculated for each privacy budget ϵ. Additionally, we tested 4 scenarios aiming at investigating the scalability of the algorithm Pert. More precisely, we revoked the Pert algorithm to inject noises to the outdoor activity invitation systems with the following settings:

(i) 200 users and 1000 invitations;

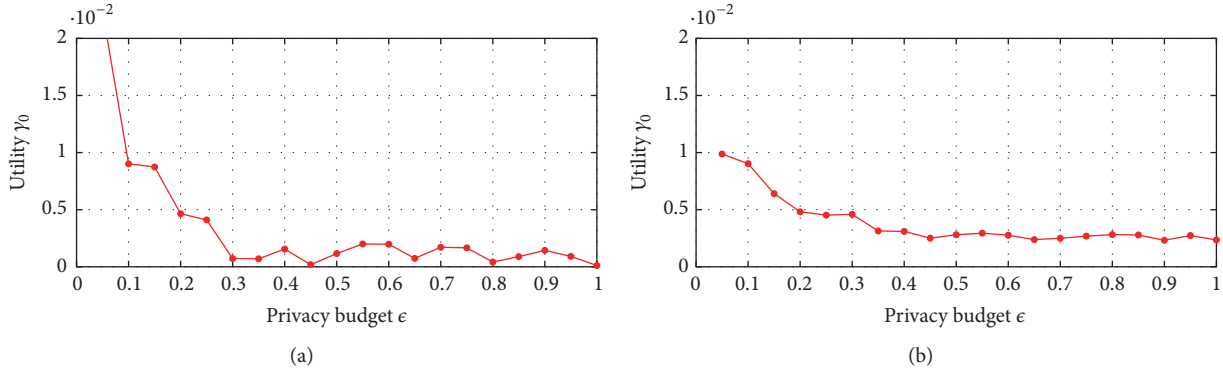

FIGURE 2: Average utility γ_0 for existing users under PERT and ADV-K-CORE. Here, the outdoor activity invitation system involves 200 users. (a, b) show the average utility γ_0 after 1000 and 5000 activities created, respectively.

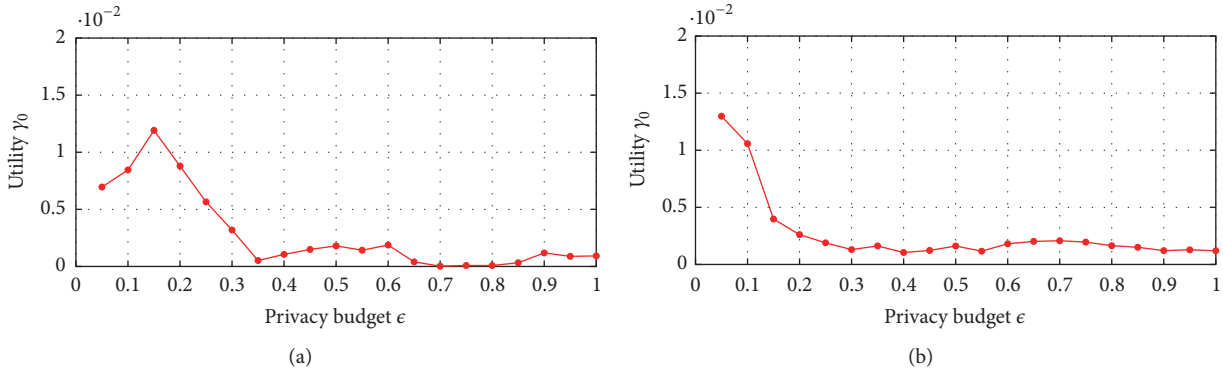

FIGURE 3: Average utility γ_0 for existing users under PERT and ADV-K-CORE. Here, the outdoor activity invitation system involves 500 users. (a, b) show the average utility γ_0 after 1000 and 5000 activities created, respectively.

(ii) 200 users and 5000 invitations;

(iii) 500 users and 1000 invitations;

(iv) 500 users and 5000 invitations.

To calculate the average utility γ_0, our ADV-K-CORE algorithm was used to select invitees. The results for the size-200 and size-500 outdoor activity invitation systems are shown in Figures 2 and 3, respectively.

Figures 2 and 3 show the average utility γ_0 is relatively small, that is, $\gamma_0 \leq 0.02$, in most cases. This demonstrates the service quality for existing users is not jeopardized severely even if they report noisy friendships to the server. Besides, the experiment results show the excellent scalability of our PERT algorithm. As the privacy budget ϵ increases, the privacy guarantee becomes weaker according to the definition of differential privacy, resulting in better services received by the existing users. Consequently, γ_0 should decrease towards 0 along the ϵ axis, which is verified by Figures 2 and 3. Actually, when $\epsilon = 0.35$, our PERT algorithm has already achieved a satisfying utility.

4.2. Experiment 2. Our second experiment is to study the efficiency of our invitation-selection algorithm ADV-K-CORE. Suppose k_1 and a_1 are the value of k and the number of remaining arcs after the algorithm K-CORE terminates. Similarly, let k_2 and a_2 be the value of k and the number of remaining arcs after the algorithm ADV-K-CORE stops. Then define two measures

$$\alpha = \frac{k_1}{k_2}, \qquad \beta = \frac{a_1}{a_2}. \tag{9}$$

The smaller values of α and β mean more average neighbors in the resulting graph after the application of ADV-K-CORE, compared with the one obtained by the employment of K-CORE. Therefore, they further indicate a closer related invitee pool, which implies invitees have higher chance to accept the invitation.

To study how the values of α and β change as the graph size changes, we applied both algorithms ADV-K-CORE and K-CORE in graphs with multiple sizes. For each size, a number of graphs of the same size were generated and then the average values of α and β were obtained over these graphs, which was designed to show how stable our algorithm ADV-K-CORE could improve the algorithm K-CORE. In our experiment setting, let the set of graph sizes be $\{100 : 100 : 1000\}$ and we calculated the average values of α and β over N graphs, where $N \in \{50, 100, 200, 500\}$. The results are shown in Figure 4.

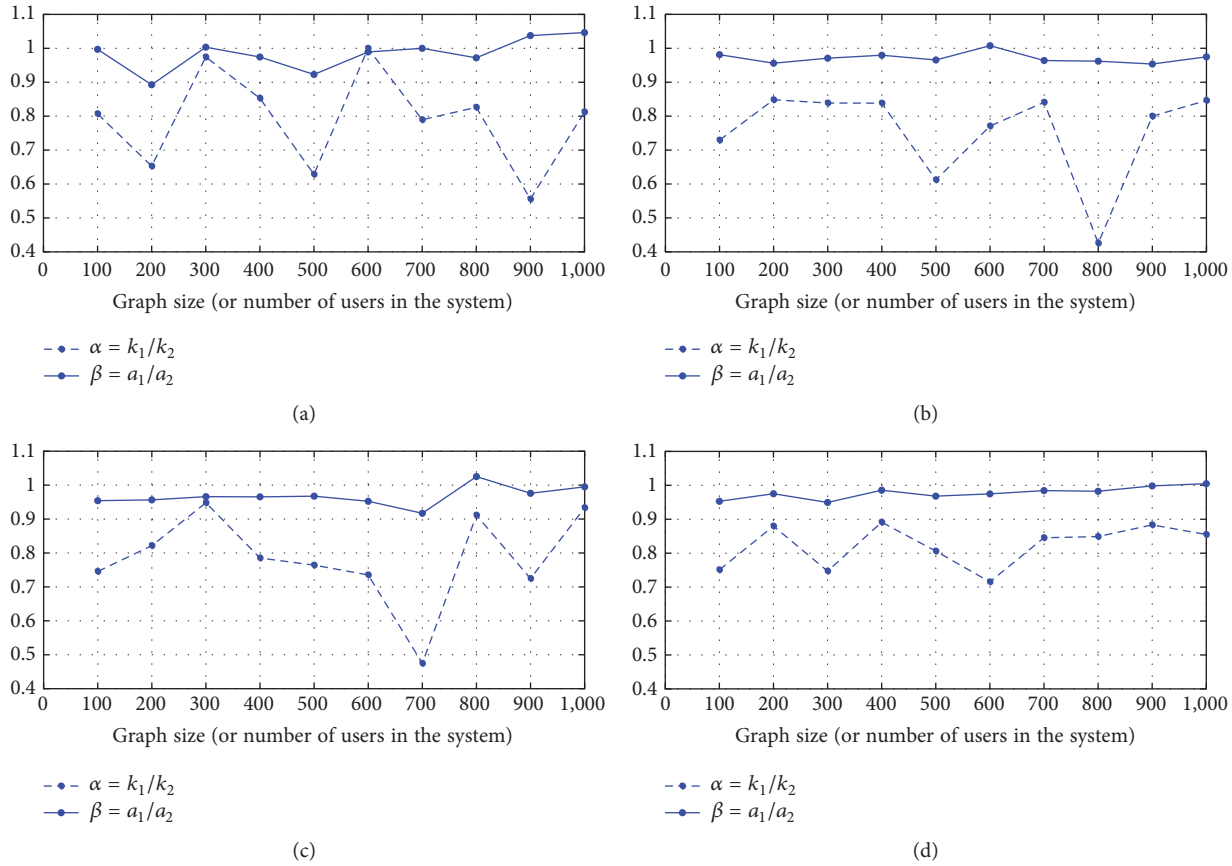

FIGURE 4: A description of how values of α and β change as the graph size changes. (a, b, c, d) were obtained by 50, 100, 200, and 500 repetitions, respectively.

From Figure 4, we can observe that $k_2 \geq k_1$ and $a_2 \geq a_1$ almost hold for all graph sizes in the experiment, which implies our algorithm ADV-K-CORE indeed produces a closer related invitee pool. Moreover, we find β is always smaller than α. This indicates the original K-CORE algorithm generates a less "consistent" adjacency in the sense that both high degree users and low degree users can be selected, which in turn results in a smaller k value. In other words, in the resultant graph after the application of our ADV-K-CORE algorithm, the variance of the numbers of neighbors is relatively smaller. Besides, we can claim that our ADV-K-CORE algorithm improves the K-CORE algorithm steadily as the values of α and β are quite stable when the graph size increases.

5. Conclusion

This paper follows the recent works by Ai et al. [10] and Tong et al. [8]. We presented a private and efficient social activity invitation framework where the server is assumed to be untrustworthy but can nonetheless help users organize group activities intelligently and efficiently. Our main contributions are (1) a novel definition of friendship to reduce the communication/update cost among the network while simultaneously enhancing data security and user confidence; (2) a strong privacy-preserving algorithm for graph publication, which addresses the concern proposed by Tong et al. [8]; (3) an efficient invitee-selection algorithm. Our simulation results show that our proposed framework has good performance. In our current research, we assumed each data field is independent with each other and queries from the adversary are also independent. In the future, we will consider more complicated queries and the correlation among data fields.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Buglass, Chen, and Tong were supported in part by the 2016 Allen E. Paulson College of Engineering & Information Technology Faculty Research Seed Grant (CEIT-FRSG) Award from CEIT, Georgia Southern University. Gao was supported in part by funds from the Office of the Vice President for Research & Economic Development at Georgia Southern University.

References

[1] L. Adamic and E. Adar, "How to search a social network," *Social Networks*, vol. 27, no. 3, pp. 187–203, 2005.

[2] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1082–1090, ACM, August 2011.

[3] Z. He, Z. Cai, Q. Han, W. Tong, L. Sun, and Y. Li, "An energy efficient privacy-preserving content sharing scheme in mobile social networks," *Personal and Ubiquitous Computing*, vol. 20, no. 5, pp. 833–846, 2016.

[4] Z. He, Z. Cai, and X. Wang, "Modeling propagation dynamics and developing optimized countermeasures for rumor spreading in online social networks," in *Proceedings of the 35th IEEE International Conference on Distributed Computing Systems (ICDCS '15)*, pp. 205–214, July 2015.

[5] R. Kumar, J. Novak, and A. Tomkins, "Structure and evolution of online social networks," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*, pp. 611–617, August 2006.

[6] M. Szell, R. Lambiotte, and S. Thurner, "Multirelational organization of large-scale social networks in an online world," *Proceedings of the National Acadamy of Sciences of the United States of America*, vol. 107, no. 31, pp. 13636–13641, 2010.

[7] R. Xiang, J. Neville, and M. Rogati, "Modeling relationship strength in online social networks," in *Proceedings of the 19th International Conference on World Wide Web*, pp. 981–990, ACM, April 2010.

[8] W. Tong, S. Buglass, J. Li, L. Chen, and C. Ai, "Smart and private social activity invitation framework based on historical data from smart devices," in *Proceedings of the 10th EAI International Conference on Mobile Multimedia Communications (MOBIMEDIA '17)*, pp. 1–10, 2017.

[9] S. Poslad, *Ubiquitous Computing: Smart Devices, Environments and Interactions*, John Wiley & Sons, 2011.

[10] C. Ai, M. Han, J. Wang, and M. Yan, "An efficient social event invitation framework based on historical data of smart devices," in *Proceedings of the IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, pp. 229–236, Atlanta, Ga, USA, October 2016.

[11] L. Backstrom, C. Dwork, and J. Kleinberg, "Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography," in *Proceedings of the 16th International World Wide Web Conference (WWW '07)*, pp. 181–190, Alberta, Canada, May 2007.

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

[13] D. Mir and R. N. Wright, "A differentially private estimator for the stochastic Kronecker graph model," in *Proceedings of the Joint EDBT/ICDT Workshops*, pp. 167–176, March 2012.

[14] C. Dwork, "Differential privacy," in *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP '06)*, pp. 1–12, 2006.

[15] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical privacy: the SulQ framework," in *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2005*, pp. 128–138, June 2005.

[16] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: privacy via distributed noise generation," in *EUROCRYPT*, pp. 486–503, 2006.

[17] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proceedings of the Theory of Cryptography Conference*, pp. 265–284, 2006.

[18] C. Dwork and K. Nissim, "Privacy-preserving datamining on vertically partitioned databases," in *Advances in Cryptology—CRYPTO 2004*, pp. 528–544, 2004.

[19] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Proceedings of the 48th Annual Symposium on Foundations of Computer Science (FOCS '07)*, pp. 94–103, Providence, RI, USA, October 2007.

[20] A. Ghosh, T. Roughgarden, and M. Sundararajan, "Universally utility-maximizing privacy mechanisms," *SIAM Journal on Computing*, vol. 41, no. 6, pp. 1673–1693, 2012.

[21] A. Nikolov, K. Talwar, and L. Zhang, "The geometry of differential privacy: The sparse and approximate cases," in *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC '13)*, pp. 351–360, June 2013.

[22] H. H. Nguyen, A. Imine, and M. Rusinowitch, "Differentially private publication of social graphs at linear cost," in *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '15)*, pp. 596–599, August 2015.

[23] Y. Mülle, C. Clifton, and K. Böhm, "Privacy-integrated graph clustering through differential privacy," in *Proceedings of the Workshops of the EDBT/ICDT Joint Conference (EDBT/ICDT)*, pp. 247–254, 2015.

[24] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao, "Sharing graphs using differentially private graph models," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC '11)*, pp. 81–97, Berlin, Germany, November 2011.

[25] Y. Wang and X. Wu, "Preserving differential privacy in degree-correlation based graph generation," *Transactions on Data Privacy*, vol. 6, no. 2, pp. 127–145, 2013.

[26] Y. Wang, X. Wu, and L. Wu, "Differential privacy preserving spectral graph analysis," in *Proceedings of 17th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD '13)*, pp. 329–340, 2013.

[27] R. Chen, B. C. M. Fung, P. S. Yu, and B. C. Desai, "Correlated network data publication via differential privacy," *The VLDB Journal*, vol. 23, no. 4, pp. 653–676, 2014.

[28] Q. Xiao, R. Chen, and K.-L. Tan, "Differentially private network data release via structural inference," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*, pp. 911–920, New York, NY, USA, August 2014.

[29] W. Lu and G. Miklau, "Exponential random graph estimation under differential privacy," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*, pp. 921–930, August 2014.

[30] C. Dwork and A. Roth, *The Algorithmic Foundations of Differential Privacy*, Now Publishers, 2014.