1 INTRODUCTION

The world is becoming increasingly interconnected, both economically and socially. There has been a noticeable rise over the past few years in the percentage of people in the U.S. who say that they use the internet, own a smartphone, and access social media. For example, roughly three-quarters of Americans (77%) owned a smartphone in early 2017. This number is more than double the level of 2011, when merely 35% reported using smartphones [6]. Similarly, in 2005, only 5% of U.S. adults used social media services, and this share rose to 50% by 2011 and to about 69% in early 2017 [7]. Nowadays, social networks are integrating into our lives in nearly every possible form and corner [1, 9, 22, 23, 27, 36, 39], as people use them to connect, interact, and share with their peers. In particular, more and more people are using social network applications on smart devices, and they tend to use them to help plan, promote, and build excitement for any public events they are hosting.

However, most social network services offer only rudimentary functions for organizing group activities. Facebook, for example, allows users to create public or private events, but the organizer can...
only choose to send invitations one-by-one or to everyone. The service is incapable of automatically sending invitations based on one or more qualifying attributes, and, in the case that there is a party limit, it is filled strictly on a first-come, first-served basis. These services are ill-suited for frequent, small events such as exercise groups: inviting every possible candidate increases the likelihood of a group where no one knows anybody except for the host, yet it is tedious to manually search for a well-acquainted social group that does the same kinds of exercise, at the same time and place. Furthermore, isolated persons with similar interests but no qualified friends would rarely be invited to such groups.

These services also pose problems from the invitees’ perspective. Users might receive lots of invitations to events that they are not willing to attend, since the invitations are usually broadcast based on the friendships of the organizer; the ability of each invitee is not considered. But as the saying goes, birds of a feather flock together. Invitees should accept invitations only when they have the ability to attend. Take group hiking as an example – different hikers have different habits and physical capacities. Some can hike at a fast pace and do not need to take breaks; others move more slowly and take frequent rests. It is not hard to imagine an inferior experience for both these hikers should they be placed in the same hiking trip. To avoid such situations, the organizer needs to selectively send invitations to people with similar habits and activity levels. Our project will design an efficient activity invitation framework which can encourage similar people to attend group activities and thus enhance every attendee’s satisfaction.

Smart wearable devices (a.k.a. wearable technology and wearable computing) have become one of the fastest-growing consumer sectors of the Internet of Things (IoT) [8]. In recent years especially, the number of smart wearable owners has increased rapidly: eMarketer [18] shows that even though penetration among U.S. adults was just 16.0% in 2015, it was a jump of 57.7% over that of 2014. What’s more, eMarketer expects the percentage to double by 2018. Similarly, IDC Research [35] forecasts that the annual worldwide shipment of smart wearable devices will experience a staggering growth of 250% over the period from 2015 to 2019. Since most smart wearable devices are equipped with an array of different sensors such as compasses, proximity sensors, accelerometers, gyroscopes, altimeters, barometers, and GPS [34], their users are involved in a movement toward data and fitness tracking. A GPS sports watch, for instance, can collect various data such as location, route, distance, pace/speed, duration, and elevation changes for different sports activities that the owner attends. These personal data can then be saved and synchronized to authorized mobile devices or laptops. From there, the habits of the device owners, such as their preferred activities, schedule, and location can be easily derived by analyzing the historical data with state-of-the-art mining or learning algorithms. This information can then be used to help the owners find group activities appropriate for them. We propose a framework to use historical data gathered from smart devices to choose invitees for an activity. Furthermore, since information such as habits, age, etc. is sensitive, our framework will also take steps to secure the participants’ privacy.

Our project aims at designing a smart and secure social activity invitation framework based on historical data gathered from smart devices. In particular, we will consider the following scenario. Suppose there is an activity organizer app for different kinds of outdoor events. After registering on this app, users can either organize activities by submitting a request to the server, which will be distributed to the other users, or receive invitations from the app server. Users can also add each other as friends. In order to receive more interesting invitations, app users need to divulge personal information such as age, sex, locational preferences, and historical data from their wearable devices. On the other hand, we assume the server is untrustworthy for several reasons.

- First, server curators have full access to users’ data, and can thus browse and even leak user data to other parties.
- Second, malicious attackers can reach the server and steal user data. The server needs to apply many security measures, such as encrypting data in storage.
- Third, users’ data is vulnerable to “man-in-the-middle” attacks [38] when it travels from device to server, because attackers can eavesdrop on the transmission channel.
- Fourth, since app developers are incentivized to make money via advertisements and more users implies more profits, the app may attempt to attract users by releasing some histogram-based information about current users or answering queries from potential new users. However, providing aggregate statistical information about the data may cause privacy leakage and de-anonymize current users, as we have seen from classic examples such as the Netflix Prize dataset [32], the Massachusetts Group Insurance Commission (GIC) medical encounter database [10], and the Metadata and Mobility databases [11].

However, designing an efficient and secure activity invitation framework is quite challenging.

1. How does the framework make the utmost effort to benefit all three parties (the current users, potential new users, and app developers), who have different goals? In particular, current users want to receive the fittest invitations; potential new users query for useful information to decide whether to register; and the app developers want to entice more users and provide the best possible services to maintain existing users. Since we know it is impossible to satisfy every party, designing the system to benefit as many participants as possible becomes a complicated optimization problem.

2. How is the framework to protect existing users’ privacy? Considering the complicated relationship between existing users and an untrustworthy server, protecting existing users’ privacy while allowing the server to provide quality service is an additional challenge.

3. Judiciously motivating less active or isolated users to participate in more group activities without upsetting existing users is yet another challenge.

In this paper, we follow a recent work by Ai et al. [2] and propose a novel activity invitation framework to address these issues, privacy in particular. First, our proposed framework collects sensed data from smart devices to generate a profile for each device owner. Then, using a state-of-the-art privacy concept, differential privacy,
we will design privacy-preserving algorithms to perturb users’ personal data before it is reported to the server. When a user submits a request to organize an activity, the server employs a k-core graph algorithm [2] to send out invitations such that each receiver has at least k friends who received invitations as well. In order to encourage new and less active or isolated users to participate in group activities, the server will adopt a novel method to perturb the k-core graph by fairly including more such users. Differential privacy will also allow the server to offer aggregate statistical information to attract more users without jeopardizing current users’ privacy.

The main contribution of our work is to solve the trilemma among existing users’ privacy; quality services to both existing and potential users; and app developers’ profits. In other words, our work will protect existing users’ privacy while satisfying all three parties involved. Lastly, our framework will provide a novel mechanism to encourage less active and isolated users. 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

Due to the popularity of smartphones, the internet, and social media, lots of social media platforms/websites, such as Facebook, Twitter, Plancast, Meetup, Yahoo! Upcoming, and Eventbrite provide services for people to organize and distribute social events. There are also many local outdoor or special activity clubs such as the Atlanta Outdoor Club [2]. However, most of these social medias offer only rudimentary functions for organizing group activities. Facebook, for instance, allows users to create public or private events, but the organizer can only choose to send invitations one-by-one or to everyone. In Plancast, users may follow event calendars of other members. Usually, an organizer simply broadcasts the invitations to the public or to his/her friends, and potential invitees either choose to subscribe to events of all or some categories, or browse events on websites. These invitation-disseminating mechanisms are ideal for neither organizers nor invitees, as group experiences can be spoiled by having too many unqualified attendees. When invitations are indiscriminately broadcast, invitees may also be overwhelmed by a plethora of different activity invitations. Furthermore, isolated persons with similar interests but no qualified friends would rarely be invited to events where invitations are sent according to friendships.

There is plenty of research in the literature on social networks; the following are the ones most related to our work. Liu et al. [29] investigated event-based social networks and discovered heavy-tailed degree distributions and strong locality among social interactions by analyzing data collected from Meetup. The authors also studied the event recommendation problem, where a simulation would evaluate users’ response rates without considering user event satisfaction. Kim et al. explored the use of k-core and inverse k-core graphs to solve the issue of selecting biased survey respondents [26], and deriving an effective decision making group for a society [25]. Ai et al. [3, 4] presented efficient algorithms to maintain as many stable partnerships as possible. Li et al. [28] and Han et al. [20, 21] proposed propagation models for the influence maximization problem in social events.

Most recently, Ai et al. [2] studied the exact same problem as our paper and proposed an efficient social event invitation framework based on Historical Data of Smart Devices. Their work, however, assumes the existence of a trusted and altruistic server and uses an invitation framework comprised of only two parties – the existing users and a server. Ai et al. [2] also handle less active or isolated users by simply assigning them a higher priority, whereas our paradigm implements a more flexible and fair mechanism to encourage these users. In addition, their framework only uses overall group statistical data or ranges instead of specific values to substitute real data. Because such simple privacy protection methods cannot guarantee the safety of every user’s personal information, our proposed framework considers a more realistic situation in which the server is selfish and possibly untrustworthy. We concentrate more on the privacy issue such that existing users will be sufficiently protected while simultaneously satisfying all three of our framework’s parties.

Differential privacy [5, 12–15] is a recent theoretical privacy model used to quantify the extent to which individuals’ privacy in a statistical data set is maintained, while preserving the usefulness of a dataset’s aggregate information. In other words, differential privacy is a strictly provable and security-controlled method. Since its inception, this concept has proven to be extremely successful. Several widely used differential privacy mechanisms, such as the Laplace mechanism [12], exponential mechanism [30], geometric mechanism [19], and Gaussian mechanism [14, 33] have been proposed. Differential privacy also has a composition property [13, 17, 24, 31], which allows for more sophisticated differentially private algorithms by combining several simpler ones.

3 SMART AND PRIVATE ACTIVITY INVITATION FRAMEWORK

In this section, we introduce our proposed smart and private activity invitation framework (refer to Figure 1).

Our framework involves three parties: a central server controlled by the app developers, the existing app users, and potential new members. The server accepts a request for organizing an activity from an existing user and then distributes the invitation to appropriate candidates such that all of them meet the group activity requirements and have a high chance to attend. The server profits in this model by advertising to its existing users. While it strives to provide quality services to maintain current members, the server also aims at attracting more users to bolster its income. It may try to entice new users by releasing some aggregate statistical information about current users, and by providing online querying services. For instance, a potential user may query for the number of current users who like hiking. If the server can guarantee many other hiking lovers, then the potential user is much more likely to join. Our framework will also occasionally group strangers together to encourage a new bond over their mutual interest. Then, if such strangers are able to befriend each other, their new friendship can be added to the system.

Existing users may have trouble deciding whether to report their personal information honestly. Candid information helps the server
compute users’ habits more accurately, which in turn leads to better services, but the server itself is not trustworthy, and users should be worried by the possibility of having their personal information leaked. To palliate this concern, the front-end, user-side app will automatically obfuscate the given personal information when reporting data to the server so that the conditions for differential privacy are satisfied. Since differential privacy is a strong privacy method, any privacy-preserving algorithm that satisfies differential privacy will protect the individual’s information regardless of the adversary’s background information, and give the demarcation of adversary’s ability.

3.1 Differential privacy

Differential privacy is a formal framework for releasing useful information about a given dataset without compromising its members’ individual privacy. Intuitively, differential privacy works by adding artificial noise to disclosed data in order to hide whether an individual’s information in a dataset has changed. In other words, given two datasets where only one participant’s data is changed, the probability distribution of outputs for a statistical analysis of one dataset should be nearly identical to the distribution of the other’s.

To formalize this, let \( x \in X^n \) and \( x' \in X^n \) be two data sets. The distance between the two datasets, denoted as \( d(x, x') \), is the minimum number of sample changes that are required to change \( x \) into \( x' \). If \( d(x, x') = 1 \), that is, if \( x \) and \( x' \) differ by at most one individual, then we say that \( x \) and \( x' \) are neighbors.

**Definition 3.1** ((\( \epsilon, \delta \))-differential privacy). A mechanism or randomized function \( M : X^n \rightarrow \mathbb{R} \) provides \((\epsilon, \delta)\)-differential privacy [12–14] if and only if for all pairs of neighboring data sets \( x \) and \( x' \), and all subset \( S \subset \text{Range}(M) \), it holds that:

\[
\Pr[M(x) \in S] \leq e^\epsilon \Pr[M(x') \in S] + \delta.
\]

The two parameters \( \epsilon \) and \( \delta \) control the level of privacy. Intuitively speaking, \( \epsilon \), usually deemed privacy budget, is an upper bound on the amount of influence that an individual’s record has on the mechanism’s output; \( \delta \) represents the probability that the mechanism’s output varies by more than a factor of \( e^\epsilon \) when applied to a data set and any one of its neighbors. Therefore, as \( \epsilon \) and \( \delta \) decrease, \( \Pr[M(x) \in S] \) and \( \Pr[M(x') \in S] \) become closer, and privacy protection is increased. Usually, the values of \( \epsilon \) and \( \delta \) are small. For instance, \( \epsilon \in (0, 1] \) and \( \delta \leq 10^{-4} \). When \( \delta = 0 \), \((\epsilon, \delta)\)-differential privacy becomes \( \epsilon \)-differential privacy.

A query \( f \) is a function that takes a data set as an input, and the answer to the query \( f \) is denoted as \( f(x) \). For example, if \( x \) is a university’s dataset, then the question "How many students were enrolled in this Fall?" is a query; it takes \( x \) as an input and then outputs a number. Given a query \( f \) and a norm function \( \| \cdot \| \) over the range of \( f \), the sensitivity of \( \Delta f \) is defined as

\[
\Delta f = \max_{d(x, x') = 1} \| f(x) - f(x') \|.
\]

The norm function \( \| \cdot \| \) is either \( L_1 \) or \( L_2 \) norm.

\( \epsilon \)-differentially private mechanisms are usually designed by adding random noise to the output of the query, that is, \( M(x) = f(x) + \text{noise} \). The Laplacian mechanism [14] and exponential mechanism [30] are two of the most popular \( \epsilon \)-differentially private mechanisms.

**Definition 3.2** (Laplacian Mechanism). Given a query \( f : X^n \rightarrow \mathbb{R}^k \), the Laplacian mechanism is defined as

\[
M_L(x) = f(x) + (Y_1, \ldots, Y_k),
\]

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 (center at 0) with scale \( b \) and its probability density function is

\[
\text{Lap}(x \mid b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right).
\]

From figure 2 we can see that increasing \( b \) increases the noise added to the output, while also flattening \( \epsilon \) values.
The exponential mechanism was designed for situations where we need to select the "optimal" response but adding noise directly to \( f(x) \) can completely destroy its value. An example would be setting a price at an auction [16]. Generally speaking, the Laplace mechanism is typically used when the output is numerical, whereas the exponential mechanism is applied to non-numerical outputs. If we define a utility function \( u : X^n \times \text{Range}(f) \to \mathbb{R} \) to map data set/output pairs to utility scores, the sensitivity of \( u \) is defined as

\[
\Delta u = \max_{r \in \text{Range}(f)} \max_{x, x'} \|u(x, r) - u(x', r)\|.
\]

Intuitively, the exponential mechanism guarantees that the change in the output of \( u \) is bounded by the sensitivity of \( u \), which allows us to simulate a social network where each user has a vertex and an edge set \( E \), connecting vertex \( u \) to \( v \) if they are friends. Let \( \Delta v \) denote a user's profile. Each field \( x \) denotes one property about this user, such as age, activity types, or age ranges. We assume that a query only concentrates on one data field, and that all pair of queries on different data field are independent. For instance, the queries "How many users are between the ages of 20 to 30?" and "What is the number of users who like hiking?" are independent, as the former concerns the age field and the latter focuses on the activity type field. With this assumption, we add noise to each field using different \( \epsilon_i \)-differentially private algorithms \( M_1, \ldots, M_n \). In other words, the noised data will be

\[
\hat{x} = (x_1 + M_1, \ldots, x_n + M_n).
\]

This data will be optimized by post-processing, using, for example, the least squares method to enhance the query’s accuracy by the optimized data \( \hat{x} \). Finally, the noised data \( \hat{x} \) will be disclosed to the server. If we choose uniform privacy budget \( \epsilon \), our perturbation will guarantee \( \max \epsilon_i = \epsilon \)-differential privacy due to the Composition theorem.

### 3.3 Selecting invitees

After a user submits a request to organize a group activity, the server will choose candidates to invite. First, the server must guarantee that all participants meet the event’s requirements, in order to maximize each participant’s satisfaction. Then, in accordance with Ai et al. [2], we assume that having friends attend an activity will improve a person’s overall experience. Therefore, the server tries to ensure that for each invitee, a number of friends will also be invited.

To accomplish this, we adopt the concept of \( k \)-cores from graph theory in order to simulate a social network where each user has at least \( k \) friends. Let \( G = (V, E) \) represent a graph with a vertex set \( V = V(G) \) and an edge set \( E = E(G) \). A vertex \( v \in V \) denotes a user who has satisfied all the requirements for an event, or are otherwise disqualified. An edge \( e = (u, v) \in E \) connects vertex \( u \) and \( v \) indicates that \( u \) and \( v \) are friends. We say a graph \( G = (V, E) \) is a \( k \)-core graph if each vertex \( v \in V \) has at least \( k \) neighbors. Let \( N_G(v) \) be the set of neighbors of
v in graph G, and let \(|N_G(v)|\) denote its cardinality, or the degree of v in G. 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.

In Ai et al. [2]’s paper, the proposed invitee-selection algorithms prefer users with more friends. They realize, however, that this preference makes the less active or isolated users even more isolated over time. To fix this, the authors implement a rule: if a user participates in less events than 80% of all members, then they are automatically invited whenever they qualify for an activity group. This means that a few invitation slots are reserved for a privileged group, resulting in a policy that may seem unfair and arbitrary to members of the framework.

Instead of reserving invitations for isolated users, we propose using a novel algorithm to inject an appropriate amount of noise to perturb friendships in a social network such that isolated users are more likely to be selected as invitees. We first construct a \(k'\)-core graph \(G' = (V', E')\), and let \(V''\) be the set of qualified candidates not included in \(G'\). We perturb the graph \((V'' \cup V''', E''')\) by negating the existence of edges independently with a small probability \(p > 0\). The perturbed graph is now denoted as \(\hat{G} = (V, E)\). That is, \(\hat{G}\) has the vertex set \(V'' \cup V''\) but it contains edge e with probability \(pe\), where \(pe = 1 - p\) if \(e \in E'\) or otherwise \(pe = p\). After deleting all isolated vertices from \(\hat{G}\), let \(G = (V, E)\) be the resultant graph. The graph perturbation algorithm is described in Figure 3.

Thus, the expected number of newly included less active and isolated users is at most
\[
\frac{n - (n - 1)p}{1 - p} < \frac{n - p}{1 - p}.
\]

By choosing appropriate values for p, we can allow a controlled number of less active or isolated users to be invited to group activities. Since this is all done randomly, we minimize the impression that certain users receive preferential treatment.

**Figure 3**: A high-level description of our graph perturbation algorithm

<table>
<thead>
<tr>
<th>Input: A (k')-core graph (G' = (V', E'))</th>
<th>Output: A (k)-core graph (G = (V, E))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Let (V'') be the set of qualified candidates not in (G').</td>
<td></td>
</tr>
<tr>
<td>2. Let (G = (V, E)) with (V = V'' \cup V'') and (E = \emptyset).</td>
<td></td>
</tr>
<tr>
<td>3. For each ((u, v) \in V \times V)</td>
<td></td>
</tr>
<tr>
<td>4. If ((u, v) \in E')</td>
<td></td>
</tr>
<tr>
<td>5. Add ((u, v)) to (E) with probability (1 - p)</td>
<td></td>
</tr>
<tr>
<td>6. Else</td>
<td></td>
</tr>
<tr>
<td>7. Add ((u, v)) to (E) with probability (p)</td>
<td></td>
</tr>
<tr>
<td>8. End if</td>
<td></td>
</tr>
<tr>
<td>9. End for</td>
<td></td>
</tr>
<tr>
<td>10. Delete all isolated vertices from (G)</td>
<td></td>
</tr>
<tr>
<td>11. Return the resultant graph (G)</td>
<td></td>
</tr>
</tbody>
</table>

4 EXPERIMENTS

In order to evaluate the performance of our activity invitation framework, we design three experiments. In these experiments, an outdoor activity invitation system is simulated, where 1000 users are created with different profiles, including age, sex, free time schedules, activity types, activity levels, and locational ranges. Then, 10,000 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 \(r \in \{0, 0.1\}\) is generated uniformly for each user in advance. When a user receives an invitation, another random number \(re \in [0, 1]\) is generated. If \(re < r\), he/she accepts the invitation; otherwise, there will be no response. All experiments are implemented with...
Smart and Private Social Activity Invitation Framework

4.1 Experiment 1

Since the personal information of existing users has theoretically been secured by differential privacy mechanisms, our Experiment 1 is to investigate whether existing users will receive worse services if they report noisy data to the server. We define the utility for each existing user as the ratio of accepted invitations vs. the number of received invitations. The higher the average ratio is, the better our framework serves the existing users. In this experiment, we test privacy budget $\epsilon \in \{0.05 : 0.05 : 1\}$, and concentrate on the age and location range fields. In particular, to better test the efficiency of our system framework, we investigate three cases: 1) only injecting noise to the age field; 2) only injecting noise to the location range field; 3) injecting noise to both of the age and location range fields.

The experiments results are shown in Figure 4, 5, 9, 8, 6 and 7. From these figures, we can observe that when the privacy budget is small, average utility is relatively low; and it increases along the increment of $\epsilon$. Thus, by choosing an appropriately small privacy budget, existing users will enjoy qualified services with their privacy protected very well. In particular, the average utility ratio for existing users is as large as 0.95 when $\epsilon \geq 0.3$ ($\epsilon \geq 0.05$, respectively), if we perturb only the age field (location range field, respectively). Figure 4 and 5, compares the utility for existing users under only age perturbation and under only location range perturbation. The comparison result shows age information is more sensitive than the location information. Besides, the Greedy algorithm is more robust than the K-core algorithm, as their performances are similar for location range perturbation while the Greedy algorithm performs slightly better for the more sensitive age field perturbation.

By injecting noise to both the age field and location range field with privacy budget $\epsilon_1$ and $\epsilon_2$ respectively, the total privacy budget will become $\epsilon_1 + \epsilon_2$, according to the Composition Theorem. Indeed, in Figure 6 and 7, the average utility for existing users decreases regardless of the choice of selection algorithm.

4.2 Experiment 2

Experiment 2 demonstrates that the noised data from existing users is still able to provide informative replies to queries from potential new users, who want information about existing users before joining up. Assume new users only care about the age information of the existing users in the database, because they only want to participate in activities with those of a similar age to themselves. In
our simulation, a user’s age can fall in a range between 15 and 70. The 1,000 users in our database have various ages. We then construct 11 queries, of the format: “How many users in the database are between the age of a and b”, where a and b are both integers divisible by 5. To evaluate whether the replies are informative, we use the mean absolute percentage error (MAPE) as a measure,

\[ \text{MAPE} = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{r_i - a_i}{a_i} \right|, \]

where \( N \) is the total number of trials, \( a_i \) is the true answer to the \( i \)-th query, and \( r_i \) is the reply from the server to the \( i \)-th query. In this experiment, we test privacy budget \( \epsilon \in [0 : 0.05 : 1] \). The experiment results is shown in Figure 10.

From Figure 10, we can observe that the MAPE decreases dramatically as the privacy budget increases, which indicate that the system can provide very informative and accurate statistical information to all queries. On the other hand, when the privacy budget \( \epsilon \) is small, say \( \epsilon < 0.3 \), any querist or potential new user is not able to learn the existence of a specific person in the current database, according to the strong mathematical privacy guarantee of differential privacy. Therefore, our framework indeed ameliorates the conflict between the existing users’ concerns about personal privacy and the app developers’ agenda.

4.3 Experiment 3

...
Experiment 3 studies the relationship between the graph perturbation parameter \( p \) and the number of newly included less active or isolated users. The k-core graph is perturbed by parameter \( p \in [0 : 0.001 : 0.01] \). After the graph has been fully perturbed, we send the database a series of 10,000 pre-generated events and keep a count of the number of invited isolated users selected by the k-core and Greedy algorithms. After all events have been simulated, the system prints the number of isolated users divided by the total number of invited users, to provide a ratio indicating the effectiveness of the perturbation from that parameter.

According to Figure 11, both k-core and Greedy algorithms tend to include more inactive or isolated users as the graph perturbation parameter \( p \) increases, which verifies our analysis result in Section 3.3. We can also observe that the k-core algorithm is more stable than the Greedy algorithm. By controlling the value of \( p \), we can limit the number of less active or isolated users to be invited. This will help to guarantee most attendees’ satisfied experience of the group activities.

![Figure 11: the relationship between the number of included isolated users and the graph perturbation parameter \( p \)](image)

### 5 CONCLUSION

This paper summarizes the mechanisms for a smart and private social activity invitation framework using historical data collected from smart devices. Our main contribution is in solving the trilemma among existing users’ privacy; quality services to both existing and potential users; and an app developers’ profits. We create a model where the server is assumed to be untrustworthy, but can nonetheless help users organize group activities intelligently and efficiently. In addition, the proposed framework helps less active or isolated members participate via a new method based on perturbed graphs. Our simulation results show that our proposed framework has good performance. In our current research, we only consider very simple queries; each query only focuses on one data field. We also assume any pair of queries are independent. In the future, we will consider the correlation among queries. Another research direction is to consider more complicated queries which may involve several data fields. Moreover, we will also explore ways to protect the friendships in the social network structure, and design more efficient invitee selection algorithms.

### ACKNOWLEDGMENTS

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

### REFERENCES


