When Privacy Meets Usability: Unobtrusive Privacy Permission Recommendation System for Mobile Apps based on Crowdsourcing

Rui Liu, *Student Member, IEEE,* Jiannong Cao, *Fellow, IEEE,* Kehuan Zhang, *Member, IEEE,* Wenyu Gao, Junbin Liang, and Lei Yang, *Member, IEEE*

Abstract— People nowadays almost want everything at their fingertips, from business to entertainment, and meanwhile they do not want to leak their sensitive data. Strong information protection can be a competitive advantage, but preserving privacy is a real challenge when people use the mobile apps in the smartphone. If they are too lax with privacy preserving, important or sensitive information could be lost. If they are too tight with privacy, making users jump through endless hoops to access the data they need to get their work done, productivity can nosedive. Thus, striking a balance between privacy and usability in mobile applications can be difficult. Leveraging the privacy permission settings in mobile operating systems, our basic idea to address this issue is to provide proper recommendations about the settings so that the users can preserve their sensitive information and maintain the usability of apps. In this paper, we propose an unobtrusive recommendation system to implement this idea, which can crowdsource users' privacy permission settings and generate the recommendations for them accordingly. Besides, our system allows users to provide feedback to revise the recommendations for getting better performance and adapting different scenarios. For the evaluation, we collected users' preferences from 382 participants on Amazon Technical Turks and released our system to users in the real world for 10 days. According to the study, our system can make appropriate recommendations which can meet participants' privacy expectation and mobile apps' usability.

Index Terms—Mobile Privacy, Crowdsourcing, Permission, Recommendation

1 Introduction

THE excitement around mobile platform has been encouraged by its unprecedented functionalities. People can get almost everything done with their fingertips nowadays, from business to entertainment, from acquiring information to ordering foods. However, everything comes with a price. For mobile applications, one important cost is user's privacy. Whenever we want a mobile app to do something useful for us, most likely we will have to give it some data about ourselves, and once the data leave our hands, they are out of our control and we totally rely on that app to protect them. As a result, users always need to make trade-offs between privacy controls and apps' functionalities. If the control is too tight, apps may not do anything useful. But if the control is too loose, it may lead to a privacy disaster.

Unfortunately, it is a strikingly difficult task to make a right decision and trade-off. Firstly, many users have not realized the importance of protecting their private data. When you were using a mobile app, have you given a second thought about your privacy? Most people will simply press "Accept" on the screen of permission authorizations without looking at the details when installing apps. Secondly, although there are still lots of people who want to preserve their private information and meanwhile use their mobile apps smoothly [1], few of them have enough background knowledge to make right decisions [2].

More importantly, existing solutions are not useful to address the problem. For example, on Android, users can choose to enable or disable Internet access for an app, but since network connection is one essential part for most apps, it is almost impossible to let users disable it while keeping expected functionality. On iOS, if one permission is disabled, the system will pop up an annoying message box asking users to enable it from time to time.

To balance the privacy and usability of mobile apps and overcome the existing difficulties, our basic idea is to provide recommendations to users about privacy permission settings based on crowdsourced data. Our previous work has proposed a mechanism to generate recommendations for users to mitigate their privacy risk when they want to set their privacy permissions of each mobile app [3]. This paper will pay more attention to the balance between privacy and usability. More specifically, we design and implement a system that can not only generate recommendations for privacy permission settings, but also improve the recommendations by learning from users' feedback.

Our system first collects users' settings of privacy permissions of each mobile app and learns the similarities among users in terms of privacy preferences and privacy

- *Rui Liu is with Department of Computing, The Hong Kong Polytechnic University, and Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong. E-mail: csrliu@comp.polyu.edu.hk*
- *Jiannong Cao is with Department of Computing, The Hong Kong Polytechnic University, Hong Kong. E-mail: csjcao@comp.polyu.edu.hk*
- *Kehuan Zhang is with Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong. E-mail: khzhang@ie.cuhk.edu.hk*
- *Wenyu Gao is with Department of Statistics, Virginia Polytechnic Institute and State University, USA. E-mail: wenyu6@vt.edu*
- *Junbin Liang is with Guangxi key laboratory of multimedia communications and network technology, School of computer and electronics information, Guangxi university, China. E-mail: liangjb@gxu.edu.cn*
- *Lei Yang is with School of Software, South China University of Technology, China. E-mail: sely@scut.edu.cn*

expectations on apps. Then, it calculates appropriate permission settings for users based on such similarities. Finally, our system can ask users' opinions about our recommendations when the apps access the data. Three options, i.e., agree, reject, and agree only this time, will be provided to a user who can choose one based on his/her actual decisions. The rationale behind our method is that: users who share similar preferences on certain private data and/or privacy expectations on apps are more likely to make similar decisions in the related privacy items, and with feedbacks, the system can improve the model of users' privacy preferences and expectations, thus can generate more accurate recommendations.

In short, the work reported herein is an attempt to predict individual user's mobile app permission settings and also study actual permission settings based on individual preferences. We believe that our results, while preliminary, are particularly promising and offer the prospect of significantly reducing user burden while empowering them to effectively control mobile app permission settings. We collected users' preferences from 382 participants on Amazon Technical Turks and released our system to 26 users in the real world for 10 days. According to the evaluation, our system can make proper recommendations which can meet participants' privacy expectations and mobile applications' usability requirements.

2 Privacy vs. Usability

Privacy is by no means a fad of modern society. In 1890, two U.S. lawyers proposed a prevalent definition: private life, habits, act, relations and the right to be alone [4]. With the proliferation of information technology, Wesin proposed that privacy is the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others, and this came to be known as information privacy [5]. These two acknowledged definitions both emphasized that privacy to users should have the ability to express themselves selectively. Moreover, as proposed by Bellotti and Sellen [6], privacy definition is not static and monolithic but should emphasize different aspects due to the coming of new technologies, patterns of use, as well as development of social norms. Thus, privacy protections of mobile apps differ from person to person and from one app to another. For example, some people care more about their contact information, thus they may be loath to provide such data no matter what kind of apps they are using. As the result, one ideal approach seems to ask users for authorization every time when any protected/cared information is accessed. It, however, is almost an impossible mission due to the usability of mobile users.

Usability of a mobile app can be defined as the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use [7]. Effectiveness means the accuracy and completeness with which users achieve specified goals. By efficiency, it means to minimize resources expended in relation to the accuracy and completeness with which users achieve goals. Satisfaction measures the freedom from discomfort, and positive attitudes towards the use of the product. Effectiveness, efficiency, and satisfaction are three factors considered when we measure mobile apps' usability. However, privacy concerns can sometimes be a major obstacle for usability. For example, the reluctance of sharing data could eliminate the effectiveness and efficiency of the apps and the endless hoops to access the data also destroy the users' satisfaction.

So, the core problem here is the conflict between usability and privacy protection, and it becomes extremely important to get a good trade-off. For better understanding, we take Android permission system as an example. To meet privacy requirement of users, Android mediates application access to different data or functions via a permissions mechanism. The ideal situation is the mobile apps request the permissions installation, and they only ask for permission to access data they really need to carry out their functions. That means users have chances to manage their privacy. If users do not want some data to be disclosed, they can change the permission settings. However, preserving the privacy also inevitably leads to some usability problem.

If a user wants to manually control the personal data usage on Android, he/she has to visit the corresponding information so that they can know how to manage, which will cost users a lot of time and contradict with effectiveness. If a user trusts and leaves everything to Android, he/she will have no awareness when some apps access the sensitive information. In any case, privacy protection in the fast-moving eco-system of mobile apps turns out to be a Sisyphean task.

Motivated by the problem discussed above, this paper proposes to build a system that can help users manage privacy settings so that they can achieve optimal trade-offs between usability and privacy, even if they do not have much professional background. More details will be elaborated in subsequent sections.

3 Recommendation Mechanism

Our previous work has already proposed such a mechanism [3], but in this work, we propose a new algorithm that takes advantage of users' demographic information and permission classification to further improve its performance.

3.1 Overview

When you want to set the privacy permissions of a mobile app on your smartphone but you do not know how to set them appropriately, what will you do? Asking others or searching the Internet for suggestions may be an immediate and intuitive idea. This is exactly what we propose to do in this paper: our system will, on your behalf, go and collect opinions from a group of people who share similar backgrounds, privacy concerns and expectations, etc., with you and make the most proper recommendations for you.

A comprehensive investigation about recommendation systems has been conducted [8]. According to the advantages of different recommendation algorithms and the characters of our scenario, we choose collaborative filtering methods to implement our idea. However, the recommendation mechanism was originally designed to attract customers to buy commodities in e-commerce markets, such as Amazon and Taobao. In our case, we do not have customers and commodities; rather we have smartphone users and privacy permission settings. We consider that people with similar backgrounds and habits may have similar privacy preferences. Thus, we map each smartphone user to a customer and each privacy permission setting to a commodity so that the item- and user-based collaborative filtering algorithms can play an expected role in our work. Further, we combine these two algorithms based on conditional probability with considering demographic and permission group information. Such a hybrid algorithm can overcome the intrinsic drawbacks and achieve better performance of item- and user-based collaborative filtering algorithms.

According to our discussion of privacy and the idea of the work, we initialized our recommendation algorithm through crowdsourced users' privacy permission settings rather than some experts' opinions. That is because we believe user expectation should be the key to set the privacy permissions of their mobile apps.

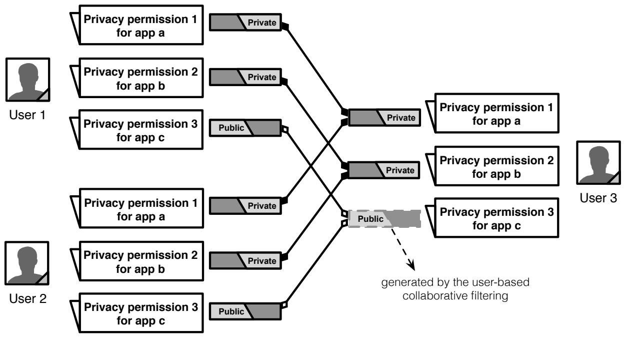
3.2 Collaborative filtering

We assume that there are K users and each user has M apps. Each app holds N data access permissions. $r_{i,a,g}$ is defined as the setting of data permission g of the app a set by the user i. Users are allowed to set the privacy setting by the dichotomous variable $\{0, 1\}$. More specifically, $r_{i,a,g} = 0$ denotes that the users are averse to share the data with anyone, whereas $r_{i,a,g} = 1$ means the participant allows the disclosure of that information. However, the users may not have sufficient understanding to different privacy permissions when they want to make a setting. It is also arduous for them to finish all of the privacy settings. To address this issue, we take advantage of user-based and item-based collaborative filtering algorithms. The following two examples and Fig. 1 further illustrate these two algorithms.

Example 1: Two users, i and j, both installed two apps a, b in the smartphone, and each app holds two permissions g, h. User i and j both allow app a to get the corresponding data permissions, by setting $r_{i,a,g} = 1$ & $r_{i,a,h} = 1$ and $r_{j,a,g} = 1$ & $r_{j,a,h} = 1$. In this situation, we consider they may have the similar privacy preferences. If user i set $r_{i,b,g} = 0$ to prohibit the access permission g of app b, user j is likely to have the same choice on this setting.

Example 2: Two apps, a' and b', both are installed in the smartphone carried by user i' and user j'. The apps a' and b' hold the permissions g' and h', respectively. If users i' and j' both reject the data access, namely setting $r_{i',a',g'} = 0$ & $r_{i',b',h'} = 0$ and $r_{j',a',g'} = 0$ & $r_{j',b',h'} = 0$. In this case, permission g' of app a' and permission h' of app b' can be considered as two similar ones because they are both rejected by users i' and j'. The more users do this, the higher similarity the two permissions have. Thus, when newcomers have negative opinions to the privacy permission g' of app a', we will also recommend them to reject the data access of permission h' of app b'.

Examples 1 and 2 illustrate the basic idea of user-based and item-based collaborative filtering approaches, respectively. According to the examples, we make recommendations by finding the users who have similar privacy preferences and finding similar permission settings. Thus,


Privacy permission 1 for app a
Privacy permission 2 for app b
Privacy permission 3 for app c
User 1
User 2
User 3
Private
Public
generated by the user-based collaborative filtering
(a) User-based collaborative filtering algorithm

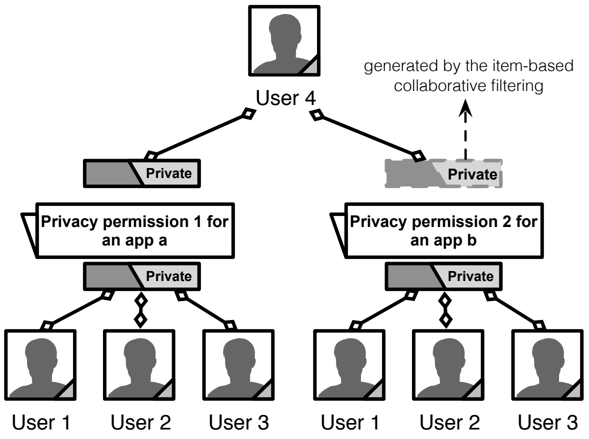

User 4
generated by the item-based collaborative filtering
Private
Privacy permission 1 for an app a
Privacy permission 2 for an app b
User 1
User 2
User 3
(b) Item-based collaborative filtering algorithm

Fig. 1. Generating recommendation of data access permissions for Android apps is based on the user- and item-based collaborative filtering algorithm.

we consider ways to calculate the similarity of users and permissions. $s_u(i,j)$ is defined as the similarity between user i and user j. The similarity reflects how similar the users i and j are, i.e., how many privacy setting permissions the two users have in common. The more such settings, the higher similarity between the users. Thus, we can calculate the similarity $s_u(i,j)$ by calculating the Pearson correlation coefficient between users i and j as shown in Eq 1, judged by their choices of permission settings. The possible similarity values range from -1 to $+1$, where values near $+1$ indicate a strong similarity. We select Pearson correlation coefficient since the empirical analyses show that for user-based recommender systems by far, the Pearson correlation coefficient outperforms other measures, such as the measures based on entropy and mean-squared difference, etc [8]–[11].

$$s_u(i,j) = \frac{\sum\limits_{a\in M}\sum\limits_{g\in N}(r_{i,a,g} - \overline{r}_i)(r_{j,a,g} - \overline{r}_j)}{\sqrt{\sum\limits_{a\in M}\sum\limits_{g\in N}(r_{i,a,g} - \bar{r_i})^2}\sqrt{\sum\limits_{a\in M}\sum\limits_{g\in N}(r_{j,a,g} - \bar{r_j})^2}} \tag{1}$$

We obtain the set of similar users by applying a threshold using $top - Q$ strategy. The $top - Q$ set of similar users to user i, $S_u(i)$, can be generated according to Eq. 2

$$S_u(i) = \{j | rank\ s_u(i,j) \leq Q\} \tag{2}$$

Likewise, we define $s_i(g,h)$ as the similarity between the privacy permission g and h. The similarity is based on the existing users' settings as illustrated in the Example 2. To calculate the similarity, we adopt the adjusted cosine similarity measure, as shown in Eq. 3. We also select $top-Q$ similar items according to Eq. 4.

$$s_i(g,h) = \frac{\sum\limits_{i \in K}\sum\limits_{a \in M}(r_{i,a,g}-\overline{r}_i)(r_{i,a,h}-\overline{r}_i)}{\sqrt{\sum\limits_{i \in K}\sum\limits_{a \in M}(r_{i,a,g}-\overline{r}_i)^2}\sqrt{\sum\limits_{i \in K}\sum\limits_{a \in M}(r_{i,a,h}-\overline{r}_i)^2}} \quad (3)$$

$$S_i(g) = \{h | rank\ s_i(g,h) \leq Q\} \quad (4)$$

Similar to the Pearson correlation coefficient, results for the adjusted cosine measure also range from -1 to $+1$, and $+1$ means strong similarity. In the formula, we subtract $\bar{r_i}$, the average permission setting for user i, to take the differences of permission setting behaviors between different users into account. We adopt the adjusted cosine similarity to calculate the similarity between permission settings because it has been empirically proven that the adjusted cosine similarity consistently outperforms the other metrics in the item-based collaborative filtering approaches [8]–[11], and it eliminates the effect of different behaviors between different users.

3.3 Fusion based on demographic and permission information

We propose a method that labels different users and permissions and then fuses the user- and item-based algorithms based on the labels. It is based on our observation that demographic and permission classification data can provide additional information about one specific user, thus leading to better results in calculating similarities. The additional information helps in reducing the data sparsity problem caused by users' unwillingness in providing some specific answers, which is an obstacle to our work in real-world deployment. Also, the demographic information and classification are getting more widely used in recommendation systems, and the information can also be deployed to fuse the user- and item-based collaborative filtering.

Our labels contain two pieces of information: demographic information from users and classification from permissions. Each user has his/her specific demographic information and each permission has its own classification. Thus, each pair of $< user, item >$ maps to one label in the label space and forms the corresponding triplet $< user, item, label >$. Therefore, considering users, items, and labels jointly, we have a three-dimensional relation $< user, item, label >$. This problem setup is similar to the social tagging systems in the recommendation mechanism. There are some recent works studying social tagging systems that model the three entities together by dimension reduction in a three-order tensor, a generalization of matrix to higher dimension [12]. In our scenario, our three-dimensional relation $< user, item, label >$ can be modelled as a three-order tensor as well. However, the matrix unfolding operations defined in the unified algorithm are not appropriate here. The matrix unfolding operations consider all the combinations of the elements in the remaining dimensions when unfolding one dimension. For example, a three-order tensor $\mathcal{A} \in R^{I_1 \times I_2 \times I_3}$ has its 1-mode matrix unfolding $A_1 \in R^{I_1 \times I_2 I_3}$. Unfortunately, this is not the optima in our scenario. On the one hand, the labels in our scenario are combinations of the demographic information and permission classifications. They cannot be combined with arbitrary items to predict users, and vice versa. On the other hand, if we force those impossible combinations to zero, the model sparsity will increase significantly.

As a result, to make it more fit to our scenario, we projected the three-dimensionalities to three two-dimensional relations, $< user, item >$, $< user, label >$, and $< item, label >$. In the projection, we separate our labels to a set of user labels L_k and a set of permission labels L_n. The user labels are generated according to demographic information, such as age, gender, occupation, and activity of mobile apps. These kinds of information can be regarded as items in a sense. If two users have similar demographic information, then we consider them having similar privacy preferences just like they are similar on items. In this way, when we calculate the similarities of users, items and labels will play the same role. That is, the number of items N is extended to $N' = N + L_k$. Likewise, a set of permission labels L_n can be treated as users who only have interests on some particular permissions as well. Therefore, as shown in Fig. 2, the new set of users can be extended by item labels, $K' = K + L_n$ and the new set of items are extended by user labels, $N' = N + L_k$. Thus, the new matrix for recomputing the similarity $s_u(i,j)$ using user-based collaborative filtering is represented in a $K \times N'$ matrix, and the new matrix for recalculating the $s_i(g,h)$ using item-based collaborative filtering is denoted by a $K' \times N$ matrix. By using projection rather than matrix unfolding operations, we are using summation instead of multiplication for the extended dimension. On the one hand, we are easy to separate the two pieces of information contained in the labels and put them to the 'right' position. On the other hand, we are creating fewer 'blanks' for the new information to fill in. That is, the model is more dense.

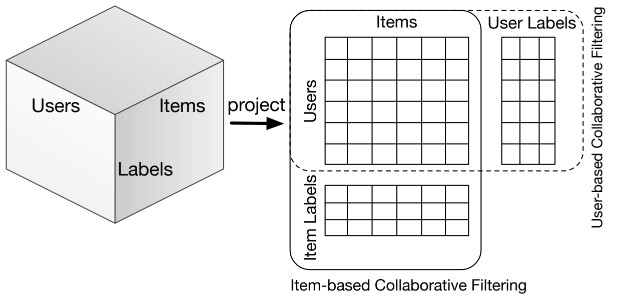

Users
Items
Labels
project
Items
User Labels
Users
Item Labels
User-based Collaborative Filtering
Item-based Collaborative Filtering

Fig. 2. The three-dimensional matrix *user-item-label* is projected as three two-dimensional matrixes, *user-item*, *user-label*, and *item-label*.

We fuse the similarities $s_u(i,j)$ and $s_i(g,h)$ based on probability to generate a more robust similarity and overcome the data sparsity problem. More specifically, we provide different weights to the two similarities $s_u(i,j)$ and $s_i(g,h)$ and form a unified similarity. In this case, the user-based and item-based collaborative filtering approaches are only two special cases in the unified form.

Assume we want to make a recommendation for user x about the privacy setting of permission z of app y, namely calculating $r_{x,y,z}$. In light of the previous illustration, a user-based collaborative filtering approach only considers privacy settings provided by the users who have similar privacy preferences. Thus, we define the existing privacy settings for calculating $r_{x,y,z}$ as a set, US, $US_{x,y,z} = \{r_{i,y,z} | i \in S_u(x)\}$. Likewise, an item-based collaborative filtering approach considers privacy settings to items. We also define a set, IS, $IS_{x,y,z} = \{r_{x,a,g} | g \in S_i(z)\}$. The similarity fusion algorithm considers these two sets jointly, i.e., UIS, $UIS_{x,y,z} = \{r_{i,a,g} | i \in S_u(x), g \in S_i(z)\}$.

When we scrutinize the privacy settings provided by users, we find they have totally different preferences to the same permission. Some users always have the sensitive attitude to the data usage, while others rarely shut down the permission due to their intrinsic traits. Some permissions are always shut down, simply because they have been set by some sensitive users. To eliminate such effects, we normalize the collected privacy settings by removing the average value, as shown in Eq. 5.

$$p_{x,y,z}(r_{i,a,g}) = r_{i,a,g} - (\overline{r}_i - \overline{r}_x) - (\overline{r}_{a,g} - \overline{r}_{y,z}) \tag{5}$$

$p_{x,y,z}(r_{i,a,g})$ serves as a normalizing function of the privacy setting of the permission z of the app y set by the user x, based on the existing crowdsourced privacy setting $r_{i,a,g}$. $\overline{r}_{a,g}$ and $\overline{r}_{y,z}$ are the mean of the privacy setting of permission g of app a and the privacy setting of permission z of app y, respectively. The sample space of the privacy permission settings should be defined as $\Phi_r = \{\emptyset, 0, 1, 2, ..., r\}$. In our case, there are actually three options, i.e., $\Phi_r = \{\emptyset, 0, 1\}$. $\emptyset$ means that the privacy settings have not been set so far, 0 expresses that users regard the information as private, and 1 represents that users allow the disclosure of this information. Therefore, $r_{i,a,g}$ denotes a privacy setting of permission g of app a, which is provided by user i, over the sample space Φ_r. Then, given a set of normalized settings, $\Omega_{x,y,z}$, we will can calculate the probability of $r_{x,y,z}$ with the condition $\mathbb{P}(r_{x,y,z}|\Omega_{x,y,z})$, where Ω is given in Eq. 6.

$$\Omega_{x,y,z} = \{p_{x,y,z}(r_{i,a,g}) | r_{i,a,g} \neq \emptyset\} \tag{6}$$

Now taking both user- and item-based recommendation algorithms into consideration, i.e $r_{i,a,g} \in (US, IS)$, we get the conditional probability presented in Eq. 7. That is, if we know $r_{i,a,g} \in (US, IS)$, we can eventually obtain the conditional probability of $r_{x,y,z}$, conditioning on the set Ω.

$$\mathbb{P}(r_{x,y,z}|\Omega_{x,y,z}) = \mathbb{P}(r_{x,y,z}|\{p_{x,y,z}(r_{i,a,g}) | r_{i,a,g} \in US \cup IS\}) \tag{7}$$

Eq. 7 indicates that the probability of $r_{x,y,z}$ depends only on $r_{i,a,g}$. Thus we can write Eq. 7 for short as $P(r_{x,y,z}|\Omega_{x,y,z}) = P(r_{x,y,z}|r_{i,a,g} \in US \cup IS)$. We introduce two independent binary indicators I_1 and I_2 to present the dependency of $r_{i,a,g}$ on set US and IS. That is, $I_1 = 1$ corresponds to dependency on the set US while $I_1 = 0$ indicates independency. Likewise, $I_2 = 1$ states $r_{i,a,g}$ depends on the set IS while $I_2 = 0$ indicates $r_{i,a,g}$ is independent of IS. Therefore, given the two sets US and IS, we can derive Eq. 8 based on the indicators I_1 and I_2.

According to the definition of indicators I_1, I_2, $r_{i,a,g}$ is independent to US if $I_1 = 0$ and is irrelevant to IS when $I_2 = 0$. Thus, $\mathbb{P}(r_{x,y,z}|I_1 = 1, I_2 = 0, US, IS) = \mathbb{P}(r_{x,y,z}|US)$, and $\mathbb{P}(r_{x,y,z}|I_1 = 0, I_2 = 1, US, IS) = \mathbb{P}(r_{x,y,z}|IS)$. Obviously, we cannot generate any recommendation without the sets US and IS, which means $\mathbb{P}(r_{x,y,z}|I_1 = 0, I_2 = 0, US, IS) = 0$. When we consider the sets US and IS jointly, these two sets can be regarded as a new set UIS. Namely, $\mathbb{P}(r_{x,y,z}|I_1 = 1, I_2 = 1, US, IS) = \mathbb{P}(r_{x,y,z}|UIS)$. Thus, we can obtain Eq. 9.

$$\begin{aligned}
&\mathbb{P}(r_{x,y,z}|US, IS) \\
&= \sum_{I_1}\sum_{I_2} \mathbb{P}(r_{x,y,z}|I_1, I_2, US, IS)\mathbb{P}(I_1, I_2|US, IS) \\
&= \mathbb{P}(r_{x,y,z}|I_1 = 0, I_2 = 0, US, IS)\mathbb{P}(I_1 = 0, I_2 = 0|US, IS) \\
&+ \mathbb{P}(r_{x,y,z}|I_1 = 1, I_2 = 0, US, IS)\mathbb{P}(I_1 = 1, I_2 = 0|US, IS) \\
&+ \mathbb{P}(r_{x,y,z}|I_1 = 0, I_2 = 1, US, IS)\mathbb{P}(I_1 = 0, I_2 = 1|US, IS) \\
&+ \mathbb{P}(r_{x,y,z}|I_1 = 1, I_2 = 1, US, IS)\mathbb{P}(I_1 = 1, I_2 = 1|US, IS)
\end{aligned} \tag{8}$$

$$\begin{aligned}
\mathbb{P}(r_{x,y,z}|US, IS) = &\mathbb{P}(r_{x,y,z}|US)\mathbb{P}(I_1 = 1, I_2 = 0|US, IS) \\
&+ \mathbb{P}(r_{x,y,z}|IS)\mathbb{P}(I_1 = 0, I_2 = 1|US, IS) \\
&+ \mathbb{P}(r_{x,y,z}|UIS)\mathbb{P}(I_1 = 1, I_2 = 1|US, IS)
\end{aligned} \tag{9}$$

For easy computation, we use two parameters λ and δ in Eq. 10, assuming $\mathbb{P}(I_1 = 1|US, IS) = \lambda$ and $\mathbb{P}(I_2 = 1|US, IS) = \delta$. According to Eq. 10, $r_{i,a,g}$ depends on both sets US and IS, i.e., UIS, when $\lambda = 1$ and $\delta = 1$. Likewise, $r_{i,a,g}$ has 0.5 probability dependent on US, if $\lambda = 0.5$; the set IS also can play a half role when δ is 0.5.

$$\begin{aligned}
\mathbb{P}(r_{x,y,z}|US, IS) = &\mathbb{P}(r_{x,y,z}|US)\lambda(1-\delta) \\
&+ \mathbb{P}(r_{x,y,z}|IS)(1-\lambda)\delta \\
&+ \mathbb{P}(r_{x,y,z}|UIS)\lambda\delta
\end{aligned} \tag{10}$$

Afterwards, we can get the estimated privacy settings $r_{x,y,z}$, as presented in Eq. 11. We can determine the parameters λ and δ through iterations in the experiments.

$$\begin{aligned}
\widehat{r}_{x,y,z} &= \sum_{t=1}^{\Phi_r} t\mathbb{P}(r_{x,y,z} = t|US, IS) \\
&= (\sum_{t=1}^{\Phi_r} t\mathbb{P}(r_{x,y,z} = t|UIS)\lambda\delta) \\
&+ (\sum_{t=1}^{\Phi_r} t\mathbb{P}(r_{x,y,z} = t|US)\lambda(1-\delta)) \\
&+ (\sum_{t=1}^{\Phi_r} t\mathbb{P}(r_{x,y,z} = t|IS)(1-\lambda)\delta)
\end{aligned} \tag{11}$$

Now we need to estimate the conditional probability in Eq. 11, namely, $\mathbb{P}(r_{x,y,z} = t|UIS)$, $\mathbb{P}(r_{x,y,z} = t|US)$, and $\mathbb{P}(r_{x,y,z} = t|IS)$. The basic idea of the estimation is to calculate the likelihood of $r_{x,y,z}$ to be similar with $r_{i,a,g}$ based on the sets US, UI, and UIS. Hence, we make use of the similarity between users to calculate the likelihood based on US, as shown in Eq. 12. Likewise, the similarity function $s_i(.)$ is used to compute the likelihood based on the set IS, as presented in Eq. 13.

$$\mathbb{P}(r_{x,y,z} = t|US) = \frac{\sum_{\forall r_{i,a,g}:(r_{i,a,g} \in US) \wedge (r_{x,y,z} = t)} s_u(i, x)}{\sum_{\forall r_{i,a,g}:r_{i,a,g} \in US} s_u(i, x)} \tag{12}$$

$$\mathbb{P}(r_{x,y,z} = t|IS) = \frac{\sum_{\forall r_{i,a,g}:(r_{i,a,g} \in IS) \wedge (r_{x,y,z} = t)} s_i(g, z)}{\sum_{\forall r_{i,a,g}:r_{i,a,g} \in IS} s_i(g, z)} \tag{13}$$

Calculating the likelihood based on UIS is a little tricky. We consider the probability estimation as a combination of the similarity function $s_u(.)$ and $s_i(.)$. More specifically, we use Euclidean distance to produce the similarity function, as illustrated in Eq. 15.

$$\mathbb{P}(r_{x,y,z} = t|UIS) = \frac{\sum_{\forall r_{i,a,g}:(r_{i,a,g} \in UIS) \wedge (r_{x,y,z}=t)} s_{ui}(r_{i,a,g}, r_{x,y,z})}{\sum_{\forall r_{i,a,g}:r_{i,a,g} \in UIS} s_{ui}(r_{i,a,g}, r_{x,y,z})} \quad (14)$$

$$s_{ui}(r_{i,a,g}, r_{x,y,z}) = \frac{1}{\sqrt{(\frac{1}{s_u(i,x)})^2 + (\frac{1}{s_i(g,z)})^2}} \quad (15)$$

Now, we can get the results,

$$\widehat{r}_{x,y,z} = \sum_{r_{i,a,g}} p_{x,y,z}(r_{i,a,g}) W^{i,a,g}_{x,y,z} \quad (16)$$

where

$$W^{i,a,g}_{x,y,z} = \begin{cases} \frac{s_u(i,x)}{\sum_{r_{i,a,g} \in US} s_u(i,x)} \lambda(1-\delta) & r_{i,a,g} \in US \\ \frac{s_i(g,z)}{\sum_{r_{i,a,g} \in IS} s_i(g,z)} (1-\lambda)\delta & r_{i,a,g} \in IS \\ \frac{s_{ui}(r_{i,a,g}, r_{x,y,z})}{\sum_{r_{i,a,g} \in UIS} s_{ui}(r_{i,a,g}, r_{x,y,z})} \lambda\delta & r_{i,a,g} \in UIS \end{cases} \quad (17)$$

So far, we have elaborated the process of recommendation based on the crowdsourced privacy settings. The only thing is to determine the parameters λ and δ. When we deploy the system in the real world, we find these two parameters, λ and δ, reaching their optima at 0.7 and 0.5, respectively. According to the illustration of the algorithm, the parameters are determined by the dataset, which means they are adaptive. More details are presented in Section 5.3.

3.4 Revision based on feedback

The aforementioned method can generate the initial recommendations for each user. Our system also allows users to provide their feedbacks and then re-generates the recommendations accordingly. The feedbacks in our system include approval, rejection, and temporal approval. For example, we recommend user i to shut down privacy permission g of app a, i.e., $r_{i,a,g} = 0$. If the user holds the approval or rejection opinion, PriVs can update that users' privacy preference with $r_{i,a,g} = 0$ or $r_{i,a,g} = 1$ respectively since the user already presents clear feedback. When a user chooses temporal approval as his/her opinion to a specific recommendation, we will count the times this user chooses this option. Then, the tricky part is how this number will impact the value of $r_{i,a,g}$.

We map the times to a value ranging from 0 to 1. The idea is based on our observations when our system was used by some voluntary users: 1) the users have little understanding about the privacy settings and corresponding recommendations when they select temporal approval for the first or second time; 2) the users become confirmed about their privacy settings and the recommendations after they select temporal approval for just several times; 3) the users eventually steady themselves. These observations also match the common sense. Therefore, the weight should be small at the very beginning, increasing rapidly along with times growth, and stable eventually. The growth rate should be increasing first and then decreasing, and the highest rate should be achieved in the middle. This is because at the beginning, as users become familiar with our app, their decisions are becoming more trustable. However, after some trials, we have already gained much information about that user. As he/she makes more decisions, our additional information gained from him/her is decreasing and finally goes to zero as the user steadies his/her choice. Therefore, the mapping relation should be an 'S' shape. Thus, we take advantage of logistic function to calculate the weight as logistic function describes the 'S' shape well. A typical application of the logistic equation is a common model of population growth. The population growth can be mapped to the weight increase. As shown in Eq. 18. x_0 is the x-value of the sigmoid's midpoint, L is the curve's maximum value, and k is the steepness of the curve.

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \quad (18)$$

Therefore, the whole process of revising recommendations is illustrated as follows:

(1) To provide an unobtrusive recommendation for users' privacy permission settings, a user-interface in PriVs will be invoked to show the requesting permissions when the app was touched to launch by users. For example, when a user opens a popular game app, all the permissions are requested by this app and corresponding recommendations will be listed in the user-interface in our prototype system, as shown in Fig. 4(g).

(2) The users will figure out the permissions in use and the corresponding recommendations made by PriVs. They can provide their opinions about the recommendations, by approving, rejecting, or temporal approving. Taking the same example, the users can click AGREE button to approve the recommended permission settings of coarse location information as feedback, as shown in Fig. 4(g) as well.

(3) We revise the corresponding user-permission matrix directly when a user chooses to approve or reject. We apply the logistic function to calculate the weight value based on the times of selecting the temporal approval and revise the matrix accordingly. In the example, a user have already approved our recommendation of coarse location information three times. In the fourth time PriVs will ask for a formal grant, as shown in Fig. 4(h).

(4) The user-interface will not be popped up any more after stablilization but PriVs allows users to keep revising the recommendations for their privacy permission settings to fit different scenarios and get better performances.

So far, we have elaborated on the process of revising the recommendations based on the crowdsourced privacy settings. We need to determine the parameters L, k and x_0. According to the scenario, L should be 1 since we are mapping the times to a score between 0 and 1. When we deploy the system in the real world, we find the remaining two parameters k and x_0 reach their optima at 1 and 3.5, respectively. More details are presented in Section 5.3.

4 System Design and Implementation

In this section, we describe the design and the implementation of PriVs, including both smartphone side (i.e., app side) and server side.

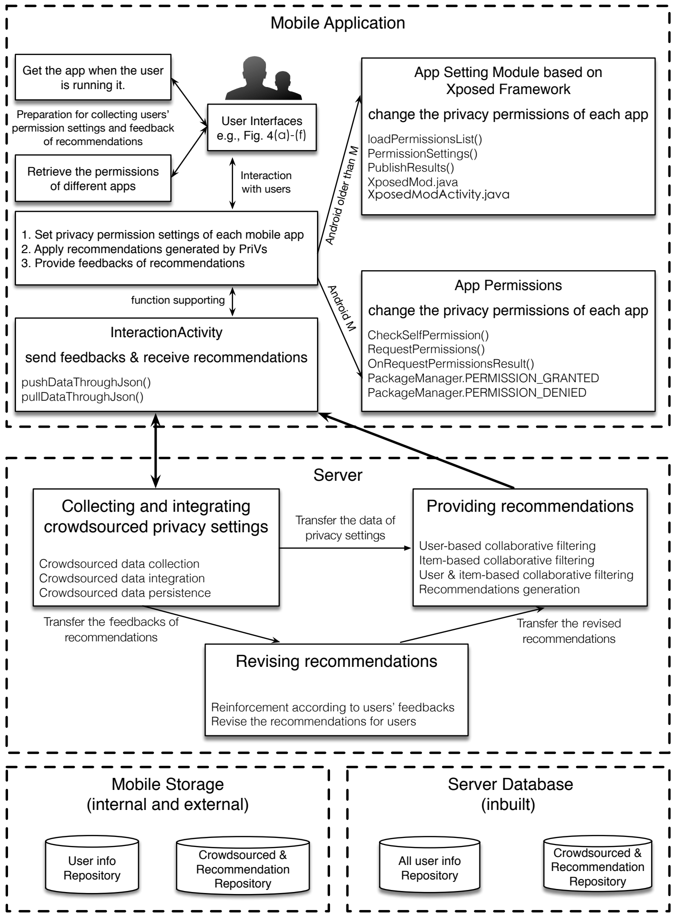

Mobile Application
Get the app when the user is running it.
Preparation for collecting users' permission settings and feedback of recommendations
Retrieve the permissions of different apps
User Interfaces e.g., Fig. 4(a)-(f)
Interaction with users
1. Set privacy permission settings of each mobile app
2. Apply recommendations generated by PriVs
3. Provide feedbacks of recommendations
function supporting
InteractionActivity
send feedbacks & receive recommendations
pushDataThroughJson()
pullDataThroughJson()
Android older than M
Android M
App Setting Module based on Xposed Framework
change the privacy permissions of each app
loadPermissionsList()
PermissionSettings()
PublishResults()
XposedMod.java
XposedModActivity.java
App Permissions
change the privacy permissions of each app
CheckSelfPermission()
RequestPermissions()
OnRequestPermissionsResult()
PackageManager.PERMISSION_GRANTED
PackageManager.PERMISSION_DENIED
Server
Collecting and integrating crowdsourced privacy settings
Crowdsourced data collection
Crowdsourced data integration
Crowdsourced data persistence
Transfer the data of privacy settings
Providing recommendations
User-based collaborative filtering
Item-based collaborative filtering
User & item-based collaborative filtering
Recommendations generation
Transfer the feedbacks of recommendations
Transfer the revised recommendations
Revising recommendations
Reinforcement according to users' feedbacks
Revise the recommendations for users
Mobile Storage (internal and external)
User info Repository
Crowdsourced & Recommendation Repository
Server Database (inbuilt)
All user info Repository
Crowdsourced & Recommendation Repository

Fig. 3. The implementation architecture of PriVs

4.1 System Framework

We have three goals when designing the system. First, it should be able to help users to make proper decisions about privacy settings. Second, the recommendation algorithm can evolve and improve its suggestions by leveraging user feedback. Third, the system should help the users to make such decisions unobtrusively, i.e., without breaking users' old habit or keeping interrupting users' normal activities.

To interact with a user, a mobile app needs to be deployed on his/her smartphone, which has following features. Firstly, it will automatically scan all apps installed on that smartphone and identify each app with its full name. Secondly, the app provides an interface that allows users to view, set or change permission settings. Thirdly, the app can receive recommendations from server and apply them automatically, and if it is preferred, allow user to express their feedback on accepting or overriding such recommendations. Finally, our system should not pose any privacy threat by itself, since its ultimate goal is to protect users' privacy.

The server side of our system has three key components. The first component aims to pre-process the data collected from user, e.g., data validation and classification. The second part focuses on generating recommendations, and the third one is responsible to receive users' feedback and leverage them to generate more user-specific (thus more accurate) recommendations. All information, including the raw crowd-sourced data, processed results, user feedback, is kept in a database at server side. According the design of mobile app and server side, we implement PriVs, as presented in Fig. 3

4.2 PriVs App

As mentioned in Section 4.1, the app running on users' smartphones should have the following four features. First, it should allow users to set and change privacy permission settings which are among the data sources used to calculate recommendations. Second, it can receive recommendations from server and apply them on users' smartphones. Third, it should provide an interface to collect user feedback to improve future recommendations. Last but not least, this app should work in an unobtrusive way, i.e., without interrupting users' normal operations or disturbing users too frequently.

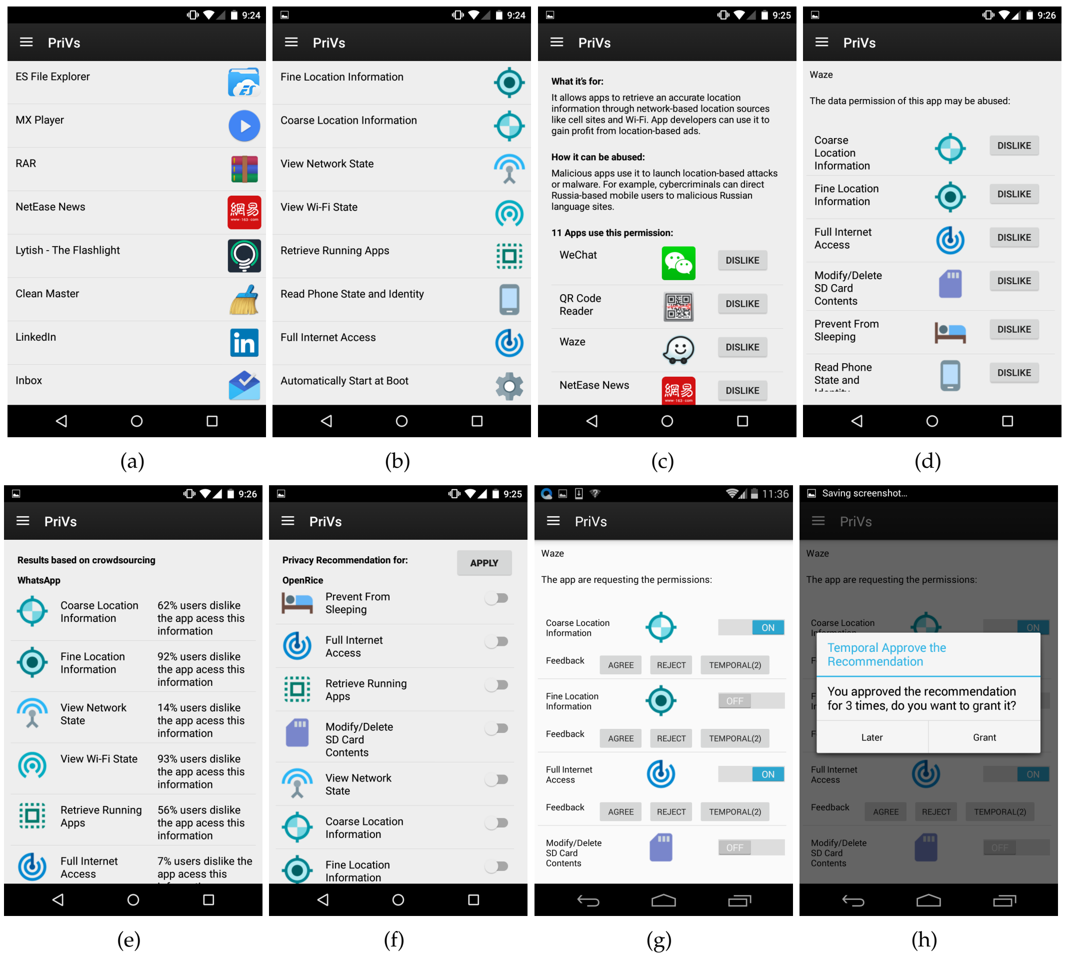
We achieve all goals with a mobile app design depicted in Figure 3. Browsing of apps and permissions is built with standard APIs in official Android SDKs. For example, function *PackageManager.getInstalledPackages(0)* can retrieve installed apps in the smartphone. Function *PackageInfo.requestedPermissions* can scrutinize the privacy permissions of each app. Such a method will return all data usages of the app. To get current foreground app, we invoke the function *ActivityManager.getRunningTasks*. According to the develop document of Android, this function was deprecated in/after Android 5.0. The alternative method is to use *ActivityManager.RunningAppProcessInfo* and *UsageStatsManager.queryUsageStats* to obtain the current app.

Since it is arduous for users to read all of these permission details, we have done some optimizations to organize permissions into groups and only focus on those having been mostly abused. With information from various sources, including Android permission groups [13], a survey given in [14], some online technical comments [15], [16] and research paper [2], [17]–[20], we summarized thirteen groups of mostly abused permissions and sensitive data, as well as their security implications, as shown in Table 1. Each group has a number of permissions. For example, the network group may include the permissions of accessing the cellular network, Wifi network information, and full internet.

In order to change permission settings of other apps, our PriVs app needs to run at a system level process, which is a reasonable assumption considering that device manufactures like Google or Samsung generally have such privilege and they may integrate our solutions into their products. For evaluation purpose, we achieve this goal on a "rooted" Android phone. For security reasons, we do not advocate users to root their smartphones because that could lead to other more serious security problems.

There are two different methods to change permission settings, depending on Android versions. For latest smartphones coming with Android M or later, PriVs app can leverage existing mechanisms to change and apply recommended permission settings directly. For devices with OS versions earlier than Android M, there is a framework called Xposed Framework [21] with which PriVs app can apply recommended permission settings.

The key functions of PriVs mobile app are depicted in Fig. 4, which is composed by snapshots of the app in Nexus 4. When users first launch PriVs, they are allowed to set privacy permission settings through user-interfaces like Fig.



9:24 PriVs
ES File Explorer
MX Player
RAR
NetEase News
Lytish - The Flashlight
Clean Master
LinkedIn
Inbox

(a)

9:24 PriVs
Fine Location Information
Coarse Location Information
View Network State
View Wi-Fi State
Retrieve Running Apps
Read Phone State and Identity
Full Internet Access
Automatically Start at Boot

(b)

9:25 PriVs
What it's for:
It allows apps to retrieve an accurate location information through network-based location sources like cell sites and Wi-Fi. App developers can use it to gain profit from location-based ads.
How it can be abused:
Malicious apps use it to launch location-based attacks or malware. For example, cybercriminals can direct Russia-based mobile users to malicious Russian language sites.
11 Apps use this permission:

WeChat	DISLIKE
QR Code Reader	DISLIKE
Waze	DISLIKE
NetEase News	DISLIKE

(c)

9:26 PriVs
Waze
The data permission of this app may be abused:

Coarse Location Information	DISLIKE
Fine Location Information	DISLIKE
Full Internet Access	DISLIKE
Modify/Delete SD Card Contents	DISLIKE
Prevent From Sleeping	DISLIKE
Read Phone State and	DISLIKE

(d)

9:26 PriVs
Results based on crowdsourcing
WhatsApp

Coarse Location Information	62% users dislike the app acess this information
Fine Location Information	92% users dislike the app acess this information
View Network State	14% users dislike the app acess this information
View Wi-Fi State	93% users dislike the app acess this information
Retrieve Running Apps	56% users dislike the app acess this information
Full Internet Access	7% users dislike the app acess this

(e)

9:25 PriVs
Privacy Recommendation for: APPLY
OpenRice
Prevent From Sleeping
Full Internet Access
Retrieve Running Apps
Modify/Delete SD Card Contents
View Network State
Coarse Location Information
Fine Location Information

(f)

11:36 PriVs
Waze
The app are requesting the permissions:

Coarse Location Information	ON		
Feedback	AGREE	REJECT	TEMPORAL(2)
Fine Location Information	OFF		
Feedback	AGREE	REJECT	TEMPORAL(2)
Full Internet Access	ON		
Feedback	AGREE	REJECT	TEMPORAL(2)
Modify/Delete SD Card Contents	OFF		

(g)

Saving screenshot... PriVs
Waze
The app are requesting the permissions:
Temporal Approve the Recommendation
You approved the recommendation for 3 times, do you want to grant it?
Later Grant
Full Internet Access ON
Feedback AGREE REJECT TEMPORAL(2)
Modify/Delete SD Card Contents OFF

(h)

Fig. 4. PriVs app runs in Android platform. (a) PriVs can scan various apps installed in a smartphone; (b) PriVs will list all the permissions of different apps (the permissions are based on our summary); (c) PriVs allows users to find out how many apps use a particular permission and express their privacy preferences; (d) PriVs also can list the permissions usage of different apps; (e) The statistical results are presented to the participants, which can be taken as a reference for their privacy preferences; (f) PriVs can receive and apply the recommendations generated by our algorithm. (g) When an app is requesting various permissions, PriVs will pop up a interface for users to collect their feedbacks with regarding to these permissions; and (h) A dialog will pop up to remind users when they choose to approve our recommendation temporarily.

TABLE 1
Summary of most abused permissions and data

Classification of the Most Abused Permissions & Data
• **Location** (The permissions that allow accessing the device location, such as fine and coarse location. It can lead location-based attacks or malware, or sending location-based ads.)
• **Phone** (The permissions that are associated with telephony features, e.g., phone states information and IMEI. The abuse of these permissions will lead privacy risk of mobile apps.)
• **Contacts** (The permissions are related to user's contacts. The abuse of these permission may disclose the contacts information.)
• **Calendar** (The permissions are related to user's calendar. The permission may release users' schedule.)
• **SMS** (The permissions are related to user's SMS messages. Sending text messages without users' awareness for subscribe additional services may leave users with unexpected charges.)
• **Sensors** (The permissions are associated with accessing camera, capturing images/video from the device or accessing vibrator function. They can stop the functions to prevent users' awareness.)
• **Network** (The permissions are associated with accessing various kinds of network. These permissions can disclose information by network and drain smartphones' battery.)
• **Apps Running** (The permissions are associated with the running apps. The abuse of these permissions will lead the information for running apps and allow malicious apps boot automatically.)
• **Storage** (The permissions related to the shared external storage, e.g., permission of modification internal and external storage. Apps steal information or save data on internal and external storage.)

4(a) - 4(d). After this initialization, the recommendations can be generated, revised and applied automatically, which can avoid endless users' intervention. The results are based on crowdsourced users' settings, as shown in Fig. 4(e). In Fig. 4(e), users can apply the recommendations generated by PriVs by only touching the "APPLY" button. Fig. 4(g) and 4(h) present the way to collect feedback.

4.3 PriVs Server

The server is designed to analyze data collected from users, then generate and return recommended settings back. As shown in Fig. 3, the server contains three key components that are used for collecting crowdsourced data, generating recommendations, and revising recommendations respectively. In the first part, the server mainly focuses on collecting and preprocessing the crowdsourced users' settings. In generating recommendation part, the server takes input from previous stage and applies methods depicted in section 3. The output of this step will be sent back to PriVs app. In the recommendation revising part, the server gets feedback from users and updates the existing recommendations using the method illustrated in section 3.4.

The server system is deployed in an IBM server and built as three-tier architecture which is composed of an application tier, a domain logic tier, and a data persistence tier. More specifically, the application tier is a web-front which is implemented with HTML, JavaScript and third-party libraries and provides a user friendly interface. The domain logic tier is implemented with Java EE and Enterprise Beans framework to analyze collected data. To improve robustness and configurability, the web application is built with mature frameworks including Spring, Struts and Hibernate. The recommendation algorithm is also deployed at this tier to generate recommendations. At data persistence tier, all data are persisted in a MySQL database.

5 Evaluation and Findings

5.1 Experiment Design and Data Collection

The evaluations and findings are based on crowdsourced data collected from our experiments on Amazon Mechanical Turk and 10-days user study. We published a task on the Amazon Mechanical Turk[1] for three weeks, and 382 participants completed our task. In the task, we asked the participants to answer a questionnaire to indicate their privacy preferences about various types of mobile apps. In order to get a better understanding, we prepared two questionnaires, survey A and survey B. Survey A was used to get the privacy preferences of participants towards various apps widely, while Survey B was used to collect fine-grained participants' preferences on certain privacy permission requests from some particular mobile apps. 200 participants completed survey A and 182 participants finished survey B.

We have performed some statistical analyses on the background of all participants, and found that they are more or less evenly distributed in terms of age, gender, work/professional background, which shows that the data are unbiased and the analysis results should be convincing. Among all the participants, 243 participants are male, and 139 participants are female. 226 participants are 20-29 years old, and 115 participants are 30-39 years old. The remainder of the participants are either 10-19 or above 40. All of the participants came from various backgrounds, such as energy, materials, consumer staples, health care, finance, information technology, etc. More information about the distribution of the participants in survey A and survey B is shown in the Table 2.

5.2 Accuracy

During the evaluation, we will use a metric defined in Eq. 19 to measure the accuracy (or effectiveness) of the proposed recommendation algorithm, where R_p denotes all the privacy permission settings the participants have chosen in the Amazon Mechanical Turk, and R_i represents the recommendations of the corresponding privacy permission settings provided by PriVs.

$$Accuracy(i) = \frac{R_p \cap R_i}{R_i} \tag{19}$$

To evaluate the accuracy of recommendations produced by PriVs, we followed the standard practice by splitting the data into two sets: one for training and the other for testing. This is done at multiple granularity levels in order to get more complete results that can be cross-verified. For example, we split data from survey A into two sets and perform the tests, and do the same on data from survey B. We also use the data from survey A as training set and data from survey B as test set, and vice versa.

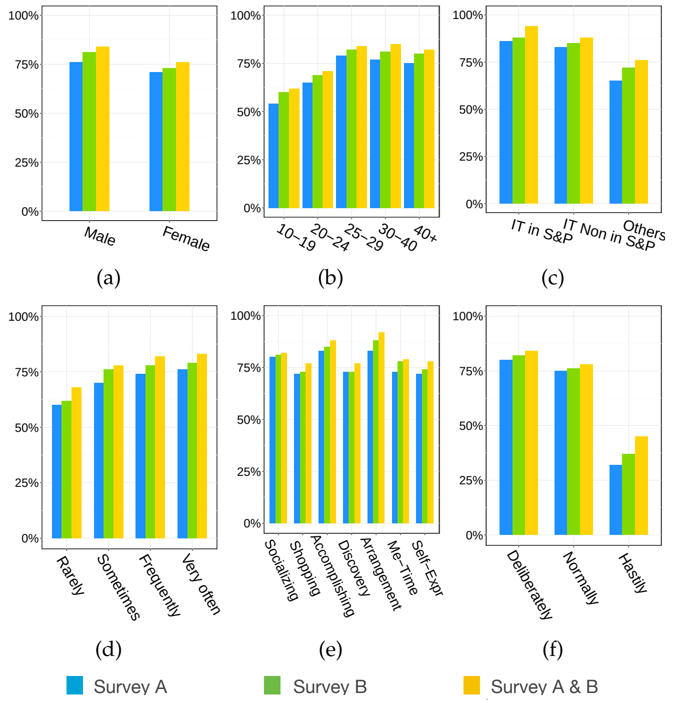


Fig. 5. The accuracy of recommendation generated by PriVs based on the participants' feedbacks from Amazon Mechanical Turk. The results are presented according to (a) the participants' genders (b) the participants' ages (c) the participants' backgrounds (d) the time participants spent on the smartphone (e) the most frequent activities of participants and (f) the attitudes of participants

All the results are demonstrated in Fig. 5. The overall accuracy of the recommendations made by PriVs is about 78%, which means most recommendations are accurate and appropriate and thus have been accepted by users. It also shows that the results with combined data sets both from survey A and survey B are better than those when each data set is used independently, which means PriVs can give more accurate recommendations when the data set is larger.

We presented the results according to participants' gender, age, background, time spent on smartphones, favourite activity on smartphones and attitude to the survey, as shown in Fig. 5(a)-5(f). Fig. 5(a) demonstrates the recommendations provided by PriVs for male participants can achieve slightly higher accuracy than those for females. There is no obvious evidence to support that males have better understanding on the privacy permission of mobile apps. However, what we found is that female participants' most frequent activities on smartphones are shopping and socializing. It may suggest that female users do not pay enough attentions to personal information on the smartphone since shopping and socializing always request personal information for functioning. Another finding is that accuracy rises gradually with the increase of participants' ages. One potential explanation is that some young people have ambiguous perceptions about their privacy permissions of their mobile

1. https://www.mturk.com/mturk/preview?groupId=3PBTVBPQ8T1PENG33V3IMPSHIB9LG1

TABLE 2
Statistics of participants in Amazon Mechanical Turk

Participants	Numbers in Survey A	Percentage in Survey A	Numbers in Survey B	Percentage in Survey B	Remarks
Male	133	66.5%	110	60.4%	We believe gender is an important factor.
Female	67	33.5%	72	39.6%	
10-19	4	2%	6	3.3%	There are two groups 20-24 and 25-29, since we think the people of 20-29 are more diverse and we separate them
20-24	45	22.5%	43	23.6%	
25-29	69	34.5%	70	38.5%	
30-40	64	32%	51	28%	
40+	18	9%	12	6.6%	
Energy	9	4.5%	6	3.3%	This taxonomy is based on Global Industry Classification Standard (GICS). The people from different vocation will have different privacy preferences.
Materials	4	2%	6	3.3%	
Industrials	19	9.5%	22	12.1%	
Consumer Discretionary	13	6.5%	7	3.9%	
Consumer Staples	12	6%	17	9.3%	
Health Care	24	12%	17	9.3%	
Finance	28	14%	21	11.5%	
IT in Security & Privacy	27	13.5%	25	13.7%	
IT in non Security & Privacy	40	20%	39	21.4%	
Tele Services	15	7.5%	19	10.4%	
Utilities	9	4.5%	3	1.7%	
Rarely (0-1hr)	7	3.5%	9	4.9%	This time indicates the participants' habits in smartphone in a sense.
Sometimes (1-2hr)	49	24.5%	49	26.9%	
Frequently (2-4 hr)	79	39.5%	56	30.8%	
Very often (4+ hr)	65	32.5%	68	37.4%	
Socializing	78	39%	59	32.4%	This taxonomy is based on *Seven Shades of Mobile* study, conducted by InsightsNow for AOL and BBDO, 2012. More then 1000 US smartphone users are involved.
Shopping	23	11.5%	16	8.8%	
Accomplishing	10	5%	14	7.7%	
Arrangement	11	5.5%	13	7.1%	
Discovery	25	12.5%	22	12.1%	
Me Time	41	20.5%	35	19.2%	
Self-expression	12	6%	23	12.6%	
Seriously completed	113	56.5%	119	65.4%	The participants who hastily completed our task make no contribution to the results
Normally completed	80	40%	61	33.5%	
Hastily completed	7	3.5%	2	1.1%	

apps. We investigated the participants with background in information technology with a focus on privacy & security and other related areas. The recommendation accuracy (around 90%) for the participants with information security background is higher than all of the others, which results from their better understandings about the privacy permission settings in smartphones. Due to the same reason, users from other areas have lower accuracy of recommendations. Fig. 5(d) indicates that PriVs does not provide very proper advice to people spending less time on smartphones. They may have inadequate knowledge about smartphone apps since they do not spend much time on them. As shown in Fig. 5(e), people who like to use some accomplishing (e.g., managing finances, health and productivity) or arrangement (planning for upcoming events) apps will get more accurate recommendations from PriVs due to their existing and crowdsourced permission settings. In the last subfigure Fig. 5(f), we can see that people who completed our task in Amazon Mechanical Turk in a rush cannot get accurate recommendations for their privacy permission settings since they just finished the task without any attention.

5.3 Parameter Estimation

There are two parameters, λ and δ, in Eq. 17. Since the exact values of both parameters are calculated from data from real world deployments, it is important to know their stability and scalability. By stability, we mean the optimal value of a parameter will not change greatly with data in different sets. By scalability, we mean the optimal value of a parameter will not change greatly with the dataset sample size. The optimal value here is defined as the value that will lead to minimum *Mean Absolute Error* (MAE) [22] shown in Equation 20 where L denotes the total number of predicted permission settings. The basic idea of MAE is to calculate the average absolute deviation of predictions to the ground truth data. In our study, several sub-datasets with different size and content were generated randomly from original testing dataset, and we computed the mean absolute error of our recommendation results to the actual selections of the participants for each sub-dataset under different parameter values. In order to remove impact of the other parameter, we first tested λ by setting δ to zero, and later nail down λ to test δ. The results are shown in Fig. 7.

$$MAE = \frac{\sum_{x,y,z} |r_{x,y,z} - \hat{r}_{x,y,z}|}{L} \tag{20}$$

Fig. 7(a) presents MAE of recommendation results by varying λ from zero to one under multiple datasets with different sizes (i.e., with 5, 20, 50 and 80 participants respectively). It shows that our recommendation algorithm can achieve minimal mean deviation error when λ falls range between 0.4 and 0.6, and further calculations will output 0.5 as the optimal value for λ. The data in Fig. 7(a) also show that the optimal value of λ is fixed around 0.5 under different datasets and different sample sizes, thus such an optimal value is stable and can scale well with different datasets.

The results of parameter δ are given in Fig. 7(b), which shows that the optimal value of δ is also pretty stable and scales well. More specifically, its optimal value is 0.7, because under all cases, our recommendation algorithm will always achieve minimal MAE when δ equals to 0.7, even though the datasets contents and sizes have changed dramatically.

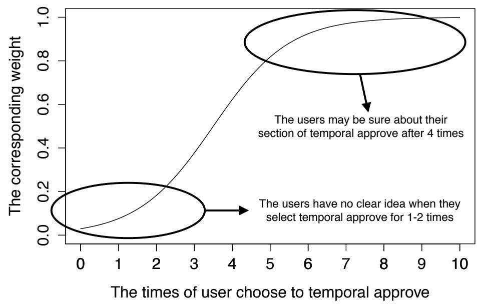


Fig. 6. The curve of logistics function and the meaning of different part. The users have no clear idea when they select temporal approve for 1-2 times but they may be sure about their section of temporal approve after 4 times

There are two parameters k and x_0 in Eq. 18 in Section 3.4. Recall that x_0 is the x-value of the sigmoid's midpoint and k is the steepness of the curve, so we calculate the average times of each users choosing the temporal approval as their opinions. According to the 10-days user study, the average value is 3.5 times. Therefore, we set $x_0 = 3.5$. Due to the same reason, we think the users who choose temporal approval for more than 6 times to a particular permission would like to approve the PriVs's recommendation. So, we set the parameter $k = 1$ to fulfil the requirement of this situation. The plot based on these parameters are shown in Fig 6. In the figure, the value is low at the very beginning since the users have little understanding to their opinions. The value is gradually rising as increase of times. The value is close to 1 when the times are larger than 6, which meets the results of the 10-days user study.

5.4 Scalability

To better understand the scalability of both parameters, i.e., how the number of participants would impact the optimal value of λ and δ, some additional experiments were done and the results are given in Fig. 7(c). It shows that: when the dataset size is small, the optimal values of both parameters will change greatly. However, when the dataset size is larger than 50, their optimal values become very stable and will stick to 0.5 and 0.7 respectively. This means: we need only to learn the optimal parameter values once with a big initial dataset, and such learned optimal values can be effective for a later real world deployment in large scale.

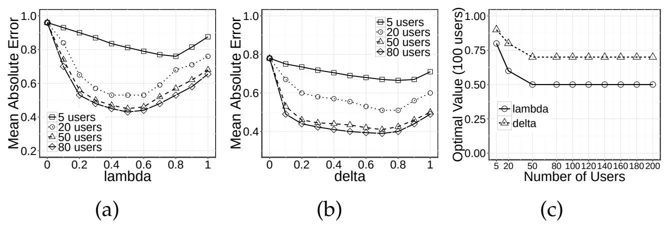


Fig. 7. Parameters estimation of the recommendation algorithm. (a) the impact of lambda (b) the impact of delta (c) the impact of size of participants

5.5 Usability

To improve the usability of PriVs, the PriVs app can be a proxy tool, which can accept and apply the recommendations generated by PriVs automatically on behalf of the users. This proxy function can be activated or deactivated at any time, which at least implicitly indicates the level of usability of PriVs. We separate the 10-days user study into three round: 1-3 day is the first round, 4-6 day is the second round, and the 7-10 day is the third round. We monitor the users' activities about proxy activation and deactivation during the user study. Fig. 8 shows the results of the proxy activation recordings of the second and third round. More than 50% of the participants activated the proxy function in the second round. This means they trust the results generated by PriVs and embrace its usability after using it for a while. Further, more participants enabled the proxy function in the third round than disabled it. Approximately 18% of the participants did not use the function during the study. Finally, after the study, we asked the participants to answer a questionnaire on how they feel with regard to their privacy in participatory sensing. Eleven participants responded.

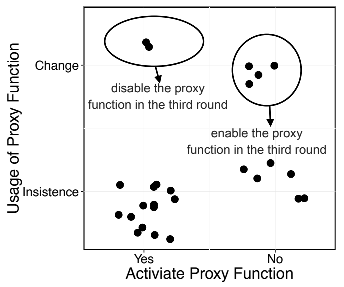


Fig. 8. Scatter plot showing the distribution of participants using the proxy function. Two participants disable the proxy function and four participants enable the proxy function in the third round.

Many participants noted that PriVs increased their awareness for their privacy concerns of their mobile apps. As examples, we would like to share the following two characteristic comments from participants:

"I never know about how to set my privacy permission since I use my mobile phone. PriVs really can help me to handle this."

"The work is well done cause how to use my apps perfectly and keep the my information safety is a problem bothers me a lot."

6 Related Work

We review some related works from two perspectives, technique-centric and human-centric, and conduct a taxonomy of the representative ones.

6.1 Technique-centric privacy preserving

Technique-centric methods mainly focus on preserving privacy using technique according to different context for various goals. Before protecting privacy, we need to understand what caused the privacy risk. Thus, the methods about detecting and analyzing the potential privacy risk of mobile apps have emerged, such as static analysis and dynamic analysis. The former one analyzes the source code of mobile apps to generate a control flow graph (CFG) rather than actually executing the apps, while the latter one monitors the mobile apps when the apps are running. LeakMiner is tool to detect disclosure of sensitive information on Android based on static analysis [23]. AndroidLeaks is a static analysis framework for finding potential leaks of sensitive information in Android apps on a large scale [24]. Static analysis for android permissions can figure out the flaws when the apps are granted more permissions than they actually need [25]. TaintDroid is a dynamic taint tacking and analysis system, which involves some aforementioned methods to simultaneously track multiple sources of sensitive data [26]. It can provide realtime analysis by leveraging Android's virtualized execution environment and detect the predefined nine situations of the information as taint.

After detecting the privacy risks, some technologies and mechanisms are proposed to mitigate or even eliminate the risks. The existing works involve permission removal, access control and data mock. Permission removal has been proposed to mitigate the privacy leak in Android smartphone [20]. It is a kind of reverse engineering process which aims to remove an app's permissions to a resource when the permissions are unrelated to the application. The repackaged app can run in the smartphone again. Access control provides a different perspective of detecting and protecting privacy in smartphone. FlaskDroid provides mandatory access control on Android's middleware and kernel layers to prevent information disclosure [27]. AppIntent provides a framework which tries to control data transmission to prevent Android applications from stealing sensitive data, meanwhile identify if transmission is from users' intentions [28]. TrustDroid is designed to isolate data and communication at different layers of the Android software stack, including the middleware layer, kernel layer and network layer [29]. AppFence is a method which aims to empower users to protect their data from exfiltration by permission-hungry applications [30]. Data mock also plays an important role in preserving privacy since some applications cannot run without accessing specific information. TISSA [31] and MockDroid [32] can provide artificial data instead of real one to the applications such that they can still function. In this case, there is no risk for users because the data are fake. However, due to the same reason, applications cannot provide competent services to users.

6.2 Human-centric privacy preserving

Human-centric methods focus on finding the balance between privacy and usability, which have the similar objective to our work. For this, we first need to understand users' privacy and then help them to preserve their privacy.

Understanding users' privacy is the key of human-centric methods, which is a claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others [33]. Therefore, privacy of mobile apps should emphasize that users should have adequate awareness and understanding to their personal and sensitive information. According to a recent survey [34], Android users hold quite different viewpoints due to their demographic characteristics, privacy awareness, and reported behaviors when installing mobile apps. It is probably surprised for users to realize data collection and distribution activities of smartphone apps [35]. Thus, it is challenging to recognize users' perceptions of whether a given action is legitimate, or how the action makes them feel with respect to privacy. A model, privacy as expectations, is proposed to capture people's expectations of privacy [36]. Appprofiler shows an approach to provide users with knowledge for decision-making about Android apps through analyzing privacy-related behaviors of apps and users' opinions [37]. After understanding user's perceptions, it is also important to assess privacy risk and predict user's privacy preferences so that we can help users to preserve their privacy. An approach is proposed for assessing the privacy risk of Android users based on impact valuation from users and their profiles [38]. Super-Ego is a crowdsourcing framework which can predict the users' privacy preferences for different locations on the basis of the general user population [39].

6.3 Taxonomy and Comparison

Table 3 gives a taxonomy of some representative existing works, as well as ours, with regard to the *perspective, objective, summary*. Considering *perspective*, we categorized all the existing works into two groups, technique-centric and human-centric. Taking *objective* into account, there are three main categories, privacy detection and analysis, privacy protection and understanding users' privacy. In *summary*, we abstract the work and highlight the main contribution.

We also add our work, PriVs, into this taxonomy as a comparison. According to this table, we illustrate our work is unique so far since we are the only one focusing on both understanding users' privacy and privacy protection. That means our work is the only one which aims to protect users' privacy based on their own opinions, preferences, attitudes.

7 Discussion

In this section, we discuss some potential limitations in our work, which may be argued.

Firstly, we initialize the recommendation mechanism according to the collected users' privacy permission settings rather than the experts' opinions. This is something about our motivation, which is elaborated in Section 2 as well, i.e, seeking a balance between usability and privacy. We believe so far there is no absolute right answer for the people who want to set their privacy permission, even some answers provided by experts because individual opinion differs from person to person [40]. Thus, in our system, we initialize our recommendation mechanism based on settings collected from each individual instead of experts. However, this does not necessary mean that experts' opinions are totally excluded. Actually there are two ways to incorporate them into our system. First, experts are also users, thus their opinions can enter system in the same way as other regular users and impact other users. Second, it is possible to set up a pool of expert users whose opinions may have relatively larger weights when generating recommendations. We will leave the latter approach as our future work.

Secondly, we discuss the parameters in the recommendation approach in Section 5.3. We determined the parameters according to the MAE of recommendations. Also, according to our illustration, the number of participants also influences the performances of recommendation algorithms. The

TABLE 3
Comparison with existing works

	Perspective	Objective	Summary
Leakminer [23]	Technique-centric	privacy detection and analysis	It is an automatic and static taint analysis method. After analyzing 1750 apps, it can identify 145 real leakages in this app set.
AndroidLeaks [24]	Technique-centric	Privacy detection and analysis	24350 Android apps were examined, 57299 potential privacy leaks in 7414 Android apps were found.
Taintdroid [26]	Technique-centric	Privacy detection and analysis	30 popular Android apps were examined, 68 instances of potential misuse of users' privacy were found across 20 apps.
FlaskDroid [27]	Technique-centric	Privacy detection and analysis	It provides mandatory access control simultaneously on both Android's middleware and kernel layers. Empirical testing is based on the security models, testbed of known malware and synthetic attacks.
AppIntent [28]	Human-centric	Privacy detection and analysis, Privacy protection	It is an analysis framework, which can provide a sequence of GUI manipulations corresponding to the sequence of events to determine if the data transmission is user intended or not.
MockDroid [32]	Technique-centric Human-centric	Privacy protection	It is a modified version of the Android which allows a user to provide artificial data instead of real one to the apps such that they can still function (possibly with reduced functionality).
Privacy as expectations [36]	Human-centric	Understanding users' privacy	It is a system which captures users' expectations of what sensitive resources mobile apps use through crowdsourcing. It found that both users' expectation and the purpose of sensitive resources can affect users' feelings and their decisions.
Appprofiler [37]	Human-centric	Understanding users' privacy	It can make informed decisions about the applications they install, which creates a knowledge base of mappings between API calls and fine-grained privacy-related behaviors to generate high-level behavior profiles.
PriVs	**Human-centric**	**Understanding users' privacy, Privacy protection**	**It is an unobtrusive system, which can crowdsource users' privacy permission settings and generate recommendations for them accordingly so that they can preserve their sensitive data and maintain the apps' usability**.

research issue about participant selection for generating recommendation algorithm is proposed, which is out of scope of this article and will be the future work.

Thirdly, there are more then 400 participants involved in our work to help us conduct the experiments and improve our research. Intuitively, the more people participate, the better the results will be. However, we cannot recruit as many participants as possible due to the time and resource limitation. Even though, we try our best to get more users involved. All the information about the participants are shown in Table 2. We avoid the statistical bias of the population, which can make our results more convincing. Also according to the our finding in Section 5.3, our current sample size is adequate to obtain good results.

Finally, we have two data sources of our experiments as shown in Section 5. One is based on Amazon Mechanical Turks, while the other is based on the real deployment. In nature, both of them are based on the real users in the world. In the Amazon Mechanical Turks, we can get more participants in easily, which is significant to our work. In the real deployment, people will use our app and provide more feedback to us since we can have the face-to-face survey, which also can help us to improve our work. That is the reason why we conduct two sorts of evaluations.

8 Conclusion and Future Work

In this paper, we presented an unobtrusive recommendation system for smartphone users to set their privacy permissions. Based on our idea about privacy and usability in mobile apps, the system can provide recommendations to users according to their preferences through crowdsourcing, and then revise the recommendations due to their feedback so that the system can find the balance between the privacy and usability. To evaluate our work, we published tasks on the Amazon Mechanical Turk and collected the feedbacks of 382 participants. Further, we implemented and deployed the system for 26 people usage during 10 days as a case study. The evaluation is based on the feedback from the Amazon Mechanical Turk and the case study shows that our system can provide proper recommendations that fit the user's individual perception of privacy, and is accepted by the users as a convenient tool due to the usability. In the future work, we plan to consider experts' opinions into our existing system and deploy the system in a large scale, like potential open dataset and app store.

Acknowledgments

This work is supported by HK RGC GRF Grant PolyU152244/15E, 14217816, and NSFC Key Project No. 61332004, NSFC Grant No. 61572415, 61502312.

References

[1] B. Liu, J. Lin, and N. Sadeh, "Reconciling mobile app privacy and usability on smartphones: could user privacy profiles help?" in *Proceedings of the 23rd international conference on World Wide Web*. ACM, 2014, pp. 201–212.

[2] P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. Sadeh, and D. Wetherall, "A conundrum of permissions: installing applications on an android smartphone," in *Financial Cryptography and Data Security*, 2012, pp. 68–79.

[3] R. Liu, J. Cao, L. Yang, and K. Zhang, "PriWe: Recommendation for privacy settings of mobile apps based on crowdsourced users' expectations," in *IEEE International Conference on Mobile Services*, 2015, pp. 150–157.

[4] S. D. Warren and L. D. Brandeis, "The right to privacy," *Harvard law review*, vol. 4, no. 5, pp. 193–220, 1890.

[5] A. F. Westin, "Privacy and freedom," *Washington and Lee Law Review*, vol. 25, no. 1, p. 166, 1968.

[6] V. Bellotti and A. Sellen, "Design for privacy in ubiquitous computing environments," in *3rd European Conference on Computer-Supported Cooperative Work 13–17 September 1993, Milan, Italy ECSCW'93*. Springer, pp. 77–92.

[7] C. Baber, P. Smith, M. Butler, J. Cross, and J. Hunter, "Mobile technology for crime scene examination," *International Journal of Human-Computer Studies*, vol. 67, no. 5, pp. 464–474, 2009.

[8] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender systems: an introduction*. Cambridge University Press, 2010.

[9] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 1999, pp. 230–237.

[10] P. Melville and V. Sindhwani, "Recommender systems," in *Encyclopedia of machine learning*. Springer, 2011, pp. 829–838.

[11] C. C. Aggarwal, *Recommender Systems - The Textbook*. Springer, 2016.

[12] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "A unified framework for providing recommendations in social tagging systems based on ternary semantic analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 2, pp. 179–192, 2010.

[13] "Android System Permissions Group." http://developer.android.com/reference/android/Manifest.permission_group.html.

[14] X. Jiang and Y. Zhou, "A survey of android malware," in *Android Malware*. Springer, 2013, pp. 3–20.

[15] "12 Most Abused Android App Permissions." http://about-threats.trendmicro.com/us/library/image-gallery/12-most-abused-android-app-permissions, 2013.

[16] "92% of top 500 android apps carry security or privacy risk." http://www.infosecurity-magazine.com/news/92-of-top-500-android-apps-carry-security-or/, 2014.

[17] P. Gerber, M. Volkamer, and K. Renaud, "Usability versus privacy instead of usable privacy: Google's balancing act between usability and privacy," *ACM SIGCAS Computers and Society*, vol. 45, no. 1, pp. 16–21, 2015.

[18] C. Orthacker, P. Teufl, S. Kraxberger, G. Lackner, M. Gissing, A. Marsalek, J. Leibetseder, and O. Prevenhueber, "Android security permissions–can we trust them?" in *Security and Privacy in Mobile Information and Communication Systems*, 2012, pp. 40–51.

[19] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: User attention, comprehension, and behavior," in *Proceedings of the Eighth Symposium on Usable Privacy and Security*. ACM, 2012, p. 3.

[20] Q. Do, B. Martini, and K.-K. R. Choo, "Enhancing user privacy on android mobile devices via permissions removal," in *2014 47th Hawaii International Conference on System Sciences (HICSS)*. IEEE, 2014, pp. 5070–5079.

[21] "Xposed Module Repository." http://repo.xposed.info/.

[22] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen, "Scalable collaborative filtering using cluster-based smoothing," in *the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, 2005, pp. 114–121.

[23] Z. Yang and M. Yang, "Leakminer: Detect information leakage on android with static taint analysis," in *Third World Congress on Software Engineering (WCSE)*. IEEE, 2012, pp. 101–104.

[24] C. Gibler, J. Crussell, J. Erickson, and H. Chen, "Androidleaks: automatically detecting potential privacy leaks in android applications on a large scale," in *International Conference on Trust and Trustworthy Computing*. Springer, 2012, pp. 291–307.

[25] J. Klein, M. Monperrus, A. Bartel, and Y. Le Traon, "Static analysis for extracting permission checks of a large scale framework: The challenges and solutions for analyzing android," *IEEE Transactions on Software Engineering*, p. 1, 2014.

[26] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: an information flow tracking system for real-time privacy monitoring on smartphones," *Communications of the ACM*, vol. 57, no. 3, pp. 99–106, 2014.

[27] S. Bugiel, S. Heuser, and A.-R. Sadeghi, "Flexible and fine-grained mandatory access control on android for diverse security and privacy policies," in *Usenix security*, 2013, pp. 131–146.

[28] Z. Yang, M. Yang, Y. Zhang, G. Gu, P. Ning, and X. S. Wang, "Appintent: Analyzing sensitive data transmission in android for privacy leakage detection," in *the 2013 ACM SIGSAC conference on Computer & Communications Security*, 2013, pp. 1043–1054.

[29] Z. Zhao and F. C. C. Osono, "Trustdroid: Preventing the use of smartphones for information leaking in corporate networks through the used of static analysis taint tracking," in *7th International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 2012, pp. 135–143.

[30] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall, "These aren't the droids you're looking for: retrofitting android to protect data from imperious applications," in *the 18th ACM conference on Computer and Communications Security*, 2011, pp. 639–652.

[31] Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh, "Taming information-stealing smartphone applications (on android)," in *Trust and Trustworthy Computing*. Springer, 2011, pp. 93–107.

[32] A. R. Beresford, A. Rice, N. Skehin, and R. Sohan, "Mockdroid: trading privacy for application functionality on smartphones," in *12th Workshop on Mobile Computing Systems and Applications*. ACM, 2011, pp. 49–54.

[33] V. M. García-Barrios, "User-centric privacy framework: Integrating legal, technological and human aspects into user-adapting systems," in *International Conference on Computational Science and Engineering*, vol. 3, 2009, pp. 176–181.

[34] Z. Benenson, F. Gassmann, and L. Reinfelder, "Android and ios users' differences concerning security and privacy," in *ACM CHI'13 Extended Abstracts on Human Factors in Computing Systems*, 2013, pp. 817–822.

[35] I. Shklovski, S. D. Mainwaring, H. H. Skúladóttir, and H. Borgthorsson, "Leakiness and creepiness in app space: Perceptions of privacy and mobile app use," in *the 32nd annual ACM conference on Human factors in computing systems*, 2014, pp. 2347–2356.

[36] J. Lin, S. Amini, J. I. Hong, N. Sadeh, J. Lindqvist, and J. Zhang, "Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, 2012, pp. 501–510.

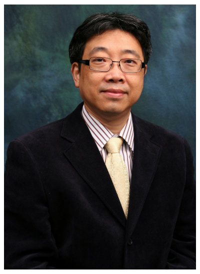
[37] S. Rosen, Z. Qian, and Z. M. Mao, "Appprofiler: a flexible method of exposing privacy-related behavior in android applications to end users," in *Proceedings of the third ACM conference on Data and application security and privacy*. ACM, 2013, pp. 221–232.

[38] A. Mylonas, M. Theoharidou, and D. Gritzalis, "Assessing privacy risks in android: A user-centric approach," in *Risk Assessment and Risk-Driven Testing*. Springer, 2014, pp. 21–37.

[39] E. Toch, "Crowdsourcing privacy preferences in context-aware applications," *Personal and ubiquitous computing*, vol. 18, no. 1, pp. 129–141, 2014.

[40] R. Liu, J. Cao, S. VanSyckel, and W. Gao, "Prime: Human-centric privacy measurement based on user preferences towards data sharing in mobile participatory sensing systems," in *2016 IEEE International Conference on Pervasive Computing and Communications, PerCom 2016, Sydney, Australia, March 14-19, 2016*, pp. 1–8.

Rui Liu is currently a research assistant in the Department of Computing at The Hong Kong Polytechnic University. He received MPhil degree from The Hong Kong Polytechnic University and the BSc degree from Northeastern University, China. He is an Outstanding Graduate in Liaoning Province and a recipient of the Google Excellence Scholarship. His research interests include ubiquitous computing, mobile computing, and crowdsourcing. He is a student member of IEEE.



Jiannong Cao is currently a chair professor and the head of the Department of Computing at Hong Kong Polytechnic University. He received the BSc degree from Nanjing University, China, and the MSc and PhD degrees from Washington State University, USA, all in computer science. His research interests include parallel and distributed computing, computer networks, mobile and pervasive computing, fault tolerance, and middleware. He co-authored 4 books, coedited 9 books, and published more than 300 technical papers in major international journals and conference proceedings. He is a fellow of IEEE, a member of ACM, and a senior member of China Computer Federation.

Kehuang Zhang is an Assistant Professor with the Information Engineering Department, The Chinese University of Hong Kong. He received the PhD degree in Informatics from Indiana University at Bloomington in 2012. His current research focuses on system and software security, including mobile computing security, cloud computing, embedded system security. He is a member of the IEEE and ACM.

Wenyu Gao is currently a PhD candidate in statistics department of Virginia Polytechnic Institute and State University. She received the Bsc degree (double major in statistics and finance) from the University of Hong Kong in 2013, the MA degree in statistics from Columbia University in 2014. Her research interests include machine learning, human-computer interaction and bayesian statistics.

Junbin Liang is currently a professor in Guangxi University. He received BSc and MSc degrees in Computer Science from the Guangxi University in 2000 and 2005, respectively. He received his PhD degree in Central South University of China in 2010. His research interests include mobile ad hoc networks, wireless sensor networks and real-time programming language design.

Lei Yang is currently an associate professor at School of Software, South China University of Technology. He received the BSc degree from Wuhan University, in 2007, the MSc degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2010, and the PhD degree from the Department of Computing, Hong Kong Polytechnic University, in 2014. During 2014 to 2015, he has been a postdoctoral fellow at Department of Computing, Hong Kong Polytechnic University. His research interest includes mobile cloud computing, Internet of Things, and data analytics.