Reconciling Mobile App Privacy and Usability on Smartphones: Could User Privacy Profiles Help?

Bin Liu[2] **Jialiu Lin**[1] **Norman Sadeh**[2]

December 2013
CMU-CS-13-128
CMU-ISR-13-144

[1]Computer Science Department, [2]Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This paper has been accepted for publication in the proceedings of the 23rd International World Wide Web Conference (WWW2014).

This research was supported in part by the National Science Foundation under grants CNS-1012763 and CNS-1330596 and by Google in the form of an unrestricted grant to the Mobile Commerce Laboratory. The authors would like to thank LBE Privacy Guard for sharing with them the data analyzed as part of this study.

Keywords

Mobile Security & Privacy, Android Permissions, Personalization

ABSTRACT

As they compete for developers, mobile app ecosystems have been exposing a growing number of APIs through their software development kits. Many of these APIs involve accessing sensitive functionality and/or user data and require approval by users. Android for instance allows developers to select from over 130 possible permissions. Expecting users to review and possibly adjust settings related to these permissions has proven unrealistic.

In this paper, we report on the results of a study analyzing people's privacy preferences when it comes to granting permissions to different mobile apps. Our results suggest that, while people's mobile app privacy preferences are diverse, a relatively small number of profiles can be identified that offer the promise of significantly simplifying the decisions mobile users have to make.

Specifically, our results are based on the analysis of settings of 4.8 million smartphone users of a mobile security and privacy platform. The platform relies on a rooted version of Android where users are allowed to choose between "granting", "denying" or "requesting to be dynamically prompted" when it comes to granting 12 different Android permissions to mobile apps they have downloaded.

1. INTRODUCTION

The past five years have seen an explosion in the number of smartphone users. At the time of writing, nearly 80% of cellphones sold in the US are smartphones. An important driver behind the adoption of smartphones is the emergence of app stores, where third party developers publish mobile apps users can download on their devices. Two competing ecosystems have emerged: the Apple iTunes store and a number of app stores based on the Android platform. Both ecosystems have competed in part based on the number of APIs they expose to developers. As more APIs are made available, developers are able to add new functionality and develop more sophisticated, and hopefully more appealing apps. Along with the explosion in APIs, both Apple's iOS and Google's Android platforms have found it necessary to also expose a greater number of settings to users. This is because not all users may necessarily feel comfortable allowing different apps to access potentially sensitive information or functionality on their smartphones such as their location, contacts lists, photos, calendar, and more. Historically, Android has relied on an approach where, as users download new apps on their smartphones, they are presented with a screen listing the data and functionality requested by the app, in the form of "permissions" they need to agree to grant to the app. The user is expected to be able to evaluate the permissions requested by an app and determine whether he or she feels comfortable granting them. Research by Kelley et al. [19] as well as by others [14], has shown that this approach leaves a lot to be desired, as most users generally do not understand the implications of their decisions and are not given a chance to revisit them later on. Apple's iOS environment initially focused on informing users about applications requesting their location, enabling them to selectively decide which app they were willing to grant access to their location and also giving them some real-time visibility into whether their location was being accessed or had been accessed over the past 24 hours. With the introduction of iOS6, this approach was extended to encompass the ability to dynamically review and revise permissions to access one's location, calendar, reminders, photos, contacts list and more. While this approach provides more control to users, it overwhelms them with options they cannot realistically be expected to manage. This situation reflects a fundamental tension between usability and privacy, with greater privacy arguing for users being given a greater number of controls or settings, and usability arguing for keeping a tractable number of decisions for users to make. Most recently, with the introduction of Android 4.3 (Jelly Bean), Android has introduced a permission manager somewhat similar to that used in iOS, enabling users to toggle permissions on and off for individual apps, effectively creating a situation where the two dominant mobile app platforms, Android and iOS, both give rise to similar tensions between usability and privacy. The fact that both platforms are moving towards making a greater number of settings available to users reflects the increasing breadth of sensitive functionality and data mobile apps can access and the difficulty of identifying default privacy settings likely to satisfy everyone. But no one so far has really tried to understand how diverse people's mobile app privacy preferences really are. Are there app permissions where a single default setting could do the trick? If a single default cannot be identified, is it possible to define a relatively small number of profiles that would enable us to capture and simplify many of the privacy decisions both Android and iOS users are expected to make today?

In this article, we look at LBE, a rooted version of the Android platform that has been in use by several million people and that has allowed its users to manually configure 12 particularly sensitive Android permissions well before the introduction of Android Jelly Bean 4.3. Specifically, we analyze a corpus of data capturing the settings selected by 4.8 million LBE users, looking at how they configured these 12 settings. Our analysis differentiates between users who have passively accepted default settings selected by LBE on their behalf and those more active users that went through the trouble of modifying these settings. While our results confirm our intuition that people's mobile privacy settings can be fairly diverse, they also strongly suggest that a relatively small number of privacy profiles could probably capture the vast majority of people's privacy preferences. These results offer the prospect of significantly simplifying the decisions users have to make without reducing the level of control they have over their privacy.

The remainder of this article is organized as follows. In Section 2, we briefly review related work in this area. In Section 3, we provide some background information about the LBE platform, the corpus of data used in this study and how this corpus was pre-processed for the purpose of our study. In Section 4, we present results of our analysis, looking at both diversity and commonality in people's mobile app privacy preferences as captured through user settings. These results are further discussed in Section 5 along with their likely implications and future possible work.

2. RELATED WORK

Mobile app privacy is getting more and more attention. A significant body of work has focused on the type of sensitive data and functionality (or "resources") accessed by mobile apps [30, 10, 12, 15, 17, 18, 7] and how users respond to existing practices [8, 14, 13, 22]. Below we provide a summary of the most relevant research.

2.1 Permission Interfaces

By default, Android apps can only access sensitive resources if they declare the corresponding permissions in their manifest files and obtain authorization from users to use them at installation time. For instance, on the Google Play store, before installing an app, a user is shown a permission screen that lists the resources the app wants to access. In order to proceed with the installation, the user needs to grant the app all the requested permissions.

Studies have shown that this permission granting process is confusing and that most users do not fully appreciate the implications of their decisions. For instance, Kelley et al. conducted semi-structured interviews of Android users and found that they paid limited attention to permission screens and had poor understanding of what the permissions implied [19]. Permission screens were shown to lack adequate explanations. Felt et al. [14] reached similar conclusions based on results from Internet surveys and lab studies..

In Android, in the absence of a permission manager such as App Ops, once permissions are granted to an app at installation time, users have no opportunity to change their minds, short of uninstalling the app. When dealing with apps accessing sensitive resources, users face a dilemma, as they can only choose between two extreme options: foregoing to use an app altogether, or allowing the app to unconditionally access sensitive data or functionality. With the introduction of App Ops in Android 4.3, this situation changes [31, 3]. While by default App Ops is a hidden permission manager, users can make it visible by downloading a corresponding app. When they do, they are given the ability to selectively toggle individual app permissions on and off. Users of the latest versions of iOS (iOS 6 and above) are by default given similar settings, which enable them to selectively toggle access to sensitive data and functionality such as location, contacts, calendar, photos and etc.

Besides the default permission interfaces offered by iOS and Android, several security and privacy extensions have been proposed, including extensions offering users finer-grained controls. For example, MockDroid [6] and TISSA [33], both designed for Android, and ProtectMyPrivacy [2], which runs on jail-broken iPhones give users the option to obfuscate responses to API calls made by mobile apps. AppFence [17], a successor to TaintDroid[10] allows users to specify resources that can only be used locally. In Apex, Nauman et al. [26] provide fine-grained control over resource usage based on context and runtime constraints such as the current location of the device or the number of times a resource has been used.

The challenge with all these solutions is that they continue to impose an unrealistic burden on users. The number of settings a user would have to configure remains unrealistically high. The work presented is intended to address this.

2.2 Privacy Policy Learning and Pattern Discovery

Frank et al. presented results obtained using data mining to analyze Android app permission requests [15]. Using matrix factorization techniques, they identified over 30 common patterns of permission requests. In contrast, our work does not focus on permission patterns as such but rather on the identification of patterns in settings selected by users when it comes to granting permissions to mobile apps. In other words, while Frank et al. focused on identifying common combinations of permissions requested by apps, we focus on (1) identifying clusters of users with similar preferences when it comes to granting permissions to apps, and more generally on (2) evaluating techniques to predict the permissions a user is likely willing to grant to an app. .

Our work is in part motivated by similar uses of machine learning to predict people's location sharing privacy preferences. This includes work by Lin et al., which demonstrated the feasibility of using machine learning techniques to predict the way people modulate the data they disclose in response to requests for their location under different situations [23] as well as earlier work by Sadeh et al. on predicting people's location sharing privacy preferences [29] . While this research showed that people's privacy preferences are complicated and often reflect tradeoffs between utility and privacy, it also showed that these preferences lent themselves to the development of quantitative models that can help predict people's decisions. Cranshaw et al. [9] described the use of multivariate Gaussian mixtures to develop classifiers capable of incrementally learning users' location sharing privacy preferences. Kelley et al [20] and later Mugan et al. also explored the development and performance of user-understandable machine learning techniques to incrementally refine models of people's location sharing privacy preferences [25]. Work by Ravichandran et al. and later Mugan et al. further showed that even user location sharing privacy preferences are diverse, an important part of their complexity could be captured with a limited number of privacy profiles [25, 28]. Wilson et al. [32] studied the impact of privacy profiles on people's location sharing decisions in the context of a 3-week pilot. They observed in particular that location sharing privacy profiles seem to have a long-term impact on the privacy settings people converge towards over time. Our work is inspired by this earlier research in the sense that we are also aiming to learn models that can help predict the permission settings a user is likely to chose for a given app and also aim to develop privacy profiles. However rather than looking at the sharing of a single piece of sensitive information, namely a user's

location, we explore the more complex problem of predicting a total of 12 permission settings for thousands of different mobile apps. In this regard, our work also relates to that of Fang and LeFevre on learning a user's sharing preferences in the context of a social network [11]. The mobile app permission domain we study is however more complex, given the number of mobile apps and permissions we need to consider..

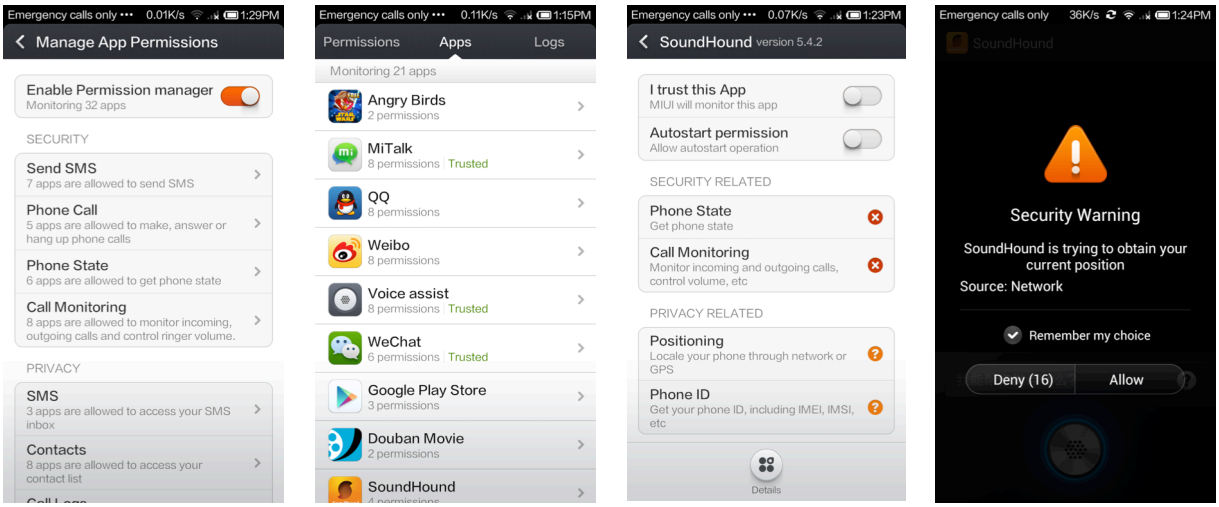

Figure 1. UI of LBE Privacy Guard on a MIUI 2S phone

In short, to the best of our best knowledge, the work reported herein is the first attempt to predict individual users' mobile app permission settings and also the first to study actual permission settings on such a large scale. We believe that our results, while preliminary, are particularly promising and offer the prospect of significantly reducing user burden while empowering users to effectively control a large number of mobile app permission settings.

3. THE LBE PRIVACY GUARD DATASET

3.1 LBE Privacy Guard

LBE Privacy Guard is a privacy and security app that requires a rooted Android phone and allows users to selectively control the permissions they are willing to grant to apps running on their phones. LBE Privacy Guard relies on API interception technology to give its users the ability to review up to 12 different permissions that can possibly be requested by an app. For each app on his or her phone and each permission requested by the app, the user can select between (always) "granting" it, (always) "denying" it, or "requesting to be dynamically prompted" each time the app attempts to access the resource associated with the permission – the resource being sensitive data or functionality. A user can at anytime revisit these permissions and elect to modify his or her selection for a given app. LBE Privacy Guard is available in the Google Play app store [21] as well as several third party app markets for rooted devices. It is also pre-shipped with a customized Android ROM called MIUI[1], which is fairly popular in mainland China. In the present study we analyze a dataset that captures the permissions of a total of 4.8 million LBE Privacy Guard users mainly based in mainland China.

The LBE app organizes all API calls by "permissions'. Our dataset covers a period of 10 days and includes user settings for the following 12 API permissions: "Send SMS", "Phone Call", "Phone State", "Call Monitoring", "SMS DB", "Contact", "Call Logs", "Positioning", "Phone ID", "3G Network", "Wi-Fi Network" and "ROOT". As the reader will notice, the nature of these permissions is very similar to that found in canonical versions of Android. For this reason, in the remainder of this article, we will simply refer to them as "permissions".

For each app-permission pair, the LBE app has 4 different possible settings:

(1) **"Allow":** The user grants the app access to the permission.

[1] http://en.miui.com/features.php

(2) **"Deny"**: The user denies the app access to the permission.

(3) **"Ask"**: Each time the app actually calls the corresponding API the system pops up a window prompting the user for a one-off decision. The window follows a 20-second countdown. In the absence of a decision within 20 seconds, the system assumes a "Deny" response. Users can also check a "Remember my choice" box to indicate that they would like their decision to become permanent (until they possibly change their mind). In this case, the settings remembered by the system change from **"Ask"** to either **"Allow"** or **"Deny"** depending on the user's election. (See Figure 1)

(4) **"Default"**: This indicates that the user has never manually modified the settings. Default settings are interpreted according to the following logic:

a. "Allow" when the permission is for access to "Wi-Fi Network", "3G Network" or "Phone ID".

b. "Allow" for the app is in a list of "trusted" apps, whatever the requested permission. Trusted apps are a collection of system apps or apps from LBE "trusted partners".

c. "Ask" in all other cases

3.2 Data Collection

Our dataset comprises the permission settings of 4.8 million LBE users in the form of permission logs collected over a 10-day time period - from May 1, 2013 to May 10, 2013. Each log record contains permission settings for all the apps (identified by package name) installed on a given device. For each app, the log records the list of permissions the app requests and the most recent settings for these permissions (namely "Allow", "Deny", "Ask" or "Default. Each user is represented by the hash of a unique user id. The term "user" here refers to a unique Android device running the LBE app. For the purpose of our analysis, we simply assume that each Android device corresponds to a distinct user. Apps are packages and are also represented by unique IDs.. Our dataset does not include app information such as installation files, versions, or app store from which an app was downloaded. The LBE app is always running on the phone either in the front view or in the background. It periodically detects if a Wi-Fi network is available. If so, the app tries to upload its log. At most one log is uploaded each day. The logs are sent regardless of the operational status of the app. If the app is not running in ROOT mode or not functioning properly, the log will simply include "Default" for all the app permissions. Below we discuss how we sanitized our dataset to deal with these types of issues.

Over the 10-day period, the dataset collected information about 4,807,884 unique users and 501,387 unique apps. The dataset comprises a total of 159,726,054 records, with a total of 118,321,621 unique triples of the form [user, app, permission]. It is worthwhile noting that, among the 4.8 million users in the dataset, 159,011 (or 3.4%) modified their settings for at least one app-permission pair over the 10-day interval. Among them 2,978 (0.06% of the users) went back and forth for at least one setting. In our analysis we focus on the final settings collected for each user over the 10-day interval. In other words, we do not limit ourselves to those users who modified their settings during the course of the ten days. This is further discussed below.

3.3 Preprocessing

Because our objective is to study people's privacy preferences as they pertain to the 12 permissions captured in the dataset, we proceeded to remove entries that might bias our analysis. In particular, we decided to focus on users who had actively engaged with the permission settings. This is in contrast to users that passively accepted them, or downloaded the app on a phone that was not rooted (in which case the user cannot control the settings), or perhaps did not even realize they had the ability to manipulate the settings. In addition, we also decided to focus on mainstream apps and removed entries that may correspond to more esoteric ones such as apps found only on secondary app markets. This is further detailed below.

(1) Our analysis focuses on what we refer to as "**representative users",** namely users who (i) have installed at least 20 apps requesting at least one permission, and (ii) have manually selected at least one "Deny" or "Ask" setting for a permission request. These restrictions are intended to eliminate users who have a particularly low number of apps on their phones – US smartphone users have been reported to have an average of 41 apps on their phones [24], and users who for one reason or another did not engage with the permission settings.

(2) Our analysis also focuses on what we refer to as "**representative apps",** namely apps that have at least one permission request, have at least 10 users in our dataset and were available on the Google Play store over the 10 day interval of this study.

This latter requirement is intended to limit our analysis to mainstream apps, in contrast to apps from less reputable stores, which might prompt users to adopt more cautious settings and possibly distort our analysis.

(3) Finally, as part of our sanitization process, we also removed app-permissions that were only recorded for 5 or fewer users. These app-permissions are assumed to correspond to exotic versions of some apps, possibly malware.

Following this screening process, our resulting dataset still had a total of 239,402 "representative users" (5.0% of the initial population) and 12,119 "representative apps" (2.4% of the initial count). The number of decision records for these users and apps totaled 28,630,179 (or 24.2% of all records we started with). On average each user had 22.66 apps on his or her smartphone. This sanitized dataset was deemed sufficiently large and diverse to warrant meaningful analysis, without being subject to the possible biases discussed above. Below we simply use the term "users" and "apps" to refer to the "representative users" and "representative apps" resulting from our screening process.

4. DATA ANALYSIS

4.1 Predicting App Permission Settings for Individual Users

4.1.1 Diversity of Users' Preferences

As already indicated, each user in our dataset had an average of 22.66 apps. On average a random pair of users had 3.19 apps in common, and each app requests an average of 3.03 permissions. A high-level analysis of user settings for different app-permission pairs shows that while there are some app-permission pairs on which the majority of users agree, there are also many such pairs for which users have diverging preferences. For instance, if one considers permissions for the top 100 apps, users agree on settings for only 63.9% of the app-permission pairs associated with these apps, if agreement is defined as 80% or more of the users selecting the same settings for a given app-permission pair (e.g. granting Angry Bird access to one's location). If one considers all the app-permission pairs for which we have at least 5 users, 80% agreement drops to 51.4%.

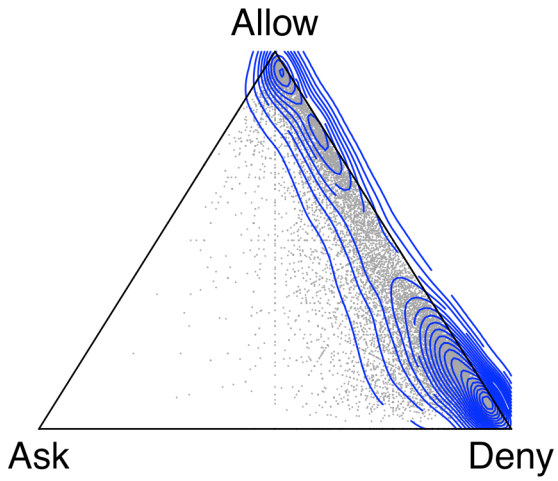

Figure 2. Distribution of users' decisions ("Allow", "Deny" and "Ask") for each app-permission pair.

Figure 2 plots the density of app-permission pairs with at least 10 user decisions based on the mix of decisions recorded for each of these pairs. Specifically, the top corner corresponds to a mix where 100% of users "allow" an app-permission, the bottom left corresponds to the case where 100% of users "ask" to be prompted for an app-permission, and the bottom right a mix where 100% of users select "deny". While many dots, each representing an app-permission pair, are concentrated around the top and bottom right corners, many are not (e.g. dots concentrated along the right side of the triangle). The plot also shows an overall bias towards either granting permissions or denying, with few users requesting to be prompted.

4.1.2 Modeling and Predicting Users' Decisions

With users having an average of over 20 apps each and each app requesting nearly 3 permissions, users are theoretically responsible for manually making around 60 privacy decisions. An obvious question is whether this number of privacy

decisions could possibly be reduced by automatically predicting the settings a user would want to select – recognizing that not all users feel the same way and that therefore a one-size-fits-all model is unlikely to work. Given that our main motivation is to alleviate user burden, we limit ourselves to a model where the set of decisions is restricted to "Allow" or "Deny," i.e. we exclude the "ask" option.

Specifically, we look at whether it might be possible to build a classifier that could be used to predict a user's app-permission setting in the form of a function

$$f: (user, app, permission) \rightarrow decision$$

The prediction model is trained using a collection of decision records in the form of {user, app, permission, decision} quadruples. As we further trim our dataset to limit ourselves to decisions that are either "Allow" or "Deny", we are left with a corpus of 14.5 million records corresponding to a total of about 239,000 users and 12,000 apps.

Through experimentation, we have found that good results can be obtained by simply using a linear kernel SVM as our model. This model also has the advantage of being quite efficient computationally [16]. The results reported below were obtained using a state-of-the-art toolbox called LibLinear [1] with both L2-loss dual support vector classification with linear kernel and L2-loss dual logistic regression to train the classifier with highest prediction power under linear kernel complexity.

Below, we report results obtained using ten-fold cross validation, where:

- We randomly split all users into ten groups of equal size.
- For each fold, one of the 10 groups is used for testing and the other 9 groups for training. For each user in the training set, all the decision records (Allow and Deny) for this user are used to train the classifier.
- For each user in the test group, we randomly choose 20% of the apps installed by the user and the corresponding permission decisions made by the user (Allow or Deny) for training as well. This data could be obtained by looking at apps already installed by the user or by simply asking the user to make some decisions for a small group of randomly selected apps – equivalent to asking the user a few questions.
- The remaining 80% of the apps downloaded by users in the test group are used to evaluate the accuracy of the classifier.

4.1.3 High Dimensionality and Sparsity Challenge

One challenge with using our dataset has to do with its high dimensionality coupled with the sparsity of data: a typical user has a little over 20 apps, but the dataset contains over 12,000 apps. A standard technique for overcoming this challenge involves the use of Singular Value Decomposition (SVD) to produce a more compact, yet essentially similar dataset by effectively projecting the data along a limited number of eigenvectors that collectively capture most of the information contained in the original dataset.

To this end, we define a preference matrix $\#User \times \#app\_permissions$ matrix of preferences P, where each entry in the matrix corresponds to a user's decision for a given app-permission. Specifically:

$$P[u][m] = \begin{cases} +1, & \text{if user } u \text{ chose "Accept" for app\_permission } m \\ -1, & \text{if user } u \text{ chose "Reject" for app\_permission } m \\ 0, & \text{if no selection has been recorded} \end{cases}$$

To the extent that many users share similar preferences, one can expect the rank of this matrix P to be much smaller than either the number of users or the number of app-permissions. In our analysis we used the "irlba" toolbox [4] in R and its implementation of the SVD algorithm [27] to produce a more compact dataset. The SVD method transforms the matrix P as:

$$P = U \cdot \Sigma \cdot t(V),$$

where $U \cdot t(U) = V \cdot t(V) = I$. Σ is a $u \times m$ diagonal matrix of eigenvalues, which are sorted in descending order. The "irlba" directly calculates an N-dimensional approximation of matrix P as:

$$P \xrightarrow{sim} U' \cdot \Sigma' \cdot t(V'),$$

where Σ' is the top left N×N sub-matrix of Σ. We generate the feature vectors of users and items as follows:

$$F_U = U' \cdot sqrt(\Sigma') \qquad F_M = V' \cdot sqrt(\Sigma')' \qquad P \xrightarrow{sim} F_U \cdot t(F_M)$$

sqrt(Σ') is a diagonal matrix whose values are the square roots of the corresponding diagonal values in Σ'. For each user u and entry m, we then have:

$$P[u][m] \xrightarrow{sim} F_U[u] \cdot t(F_M[m])$$

Below we report results obtained by limiting dimensionality to the 100 most significant eignvectors (N=100), which provides for a compact, yet expressive summary of the original dataset.

An alternative to using SVD involves simply aggregating all user information along the 12 permissions available in the data set. This can be done using $\#User \times \#Permissions$ matrix of preferences P, where each entry in the matrix aggregates decisions made by a given user for the corresponding permission, as:

$$P[u][m] = \begin{cases} +1, & \text{if user } u \text{ always chose "Accept" for permission } m \\ -1, & \text{if user } u \text{ always chose "Reject" for permision } m \\ \dfrac{a-r}{a+r}, & \text{if among all decisions for permission } m, \text{ user } u \text{ chose "Accept" } a \text{ times and "Reject" } r \text{ times} \\ 0, & \text{if no record available of user } u \text{ for item } m \end{cases}$$

Below we report results obtained by enriching the dataset with either of these two models, namely a model where preferences are aggregated around each the 12 permissions (Model 1) and one obtained using SVD (Model 2).

4.1.4 Performance of the Default Settings Prediction

Preliminary analysis discussed in subsection 4.1.1 suggests that people's privacy preferences when it comes to granting permissions are diverse. In this subsection, we take a closer look at the importance of different features in building classifiers that can be used to predict a user's permission decisions. As discussed in Subsection 4.1.2, we use 10-fold cross validation. We also include in the training set permission decisions for 20% of the apps installed by users in the testing group. This is intended to capture scenarios where we use privacy preferences for apps a user has already installed to predict permission decisions for new apps he or she downloads on his/her phone.

Table 1: Feature Compositions

Feature Set	Features	Description
FS-1	Permission IDs	Preference statistics of all users on apps & permissions
FS-2	App IDs	
FS-3	App ids & Permission IDs	
FS-4	User IDs	+ Users' overall preferences
FS-5	User ids & Permission IDs	
FS-6	User ids & App IDs	
FS-7	User ids, App IDs & Permission IDs	
FS-8	FS-7 appended with aggregated P[u][m] for corresponding user and permission	+ Numerical estimation of users' preferences from aggregation of permission or SVD on user and app-permission pairs.
FS-9	FS-7 appended with estimated P[u][m] for corresponding user and app-permission pairs from top-200 apps	
FS-10	FS-7 appended with estimated P[u][m] for corresponding user and app-permission pairs from top-1000 apps	

Table 1 summarizes the 10 feature sets considered in this particular part of our study. They include a feature set where we aggregate decisions across all users and all apps (FS-1), a feature set where we aggregate decisions across all users and all permissions (FS-2), one where we aggregate decisions across all users for each app-permission (FS3), one where we aggregate decisions for each user across all apps and all permissions (FS-4), one where data is organized by user ID and permission ID (i.e. aggregated across all apps for each user-permission pair) (FS-5), one where data is aggregated across all permissions for each app-user pair (FS-6), and one where data is broken down for each user by app-permission pair (i.e. user-app-permission triples) (FS-7). We also consider three feature sets where FS-7 is enriched with:

- The 12-permission user profiles introduced in 4.1.3 as "Model 1: - referred to as Feature Set 8 (or FS-8) in Table 1

- An SVD model (Model 2 introduced in 4.1.3) of user-permissions obtained by focusing on the 200 most popular apps in the dataset
- An SVD model (Model 2 introduced in 4.1.3) of user-permissions obtained by focusing on the 1,000 most popular apps in the dataset

As can be seen in Figure 3, looking at the prediction accuracy obtained with each of these feature sets, users, apps and permissions all contribute to enriching the model and increasing its predictive power, with FS-7 (accuracy of 85.03% and Std Err = 0.08%) outperforming the other six feature sets FS-1 through FS-6. Supplementing these features with SVD models based on the top 200 or 1000 most popular apps does not help and in fact results in lower predictive accuracy. On the hand adding user profiles based on the 12 permissions ("model 2"/FS-8) does enhance accuracy, bringing it from 85.03% to 87.8% (Std Err = 0.06%). The lack of improvement with the SVD model could be due to the fact that we took too many apps into account (200 and 1000 most popular apps). A model based on a smaller number of apps (which would increase the likelihood that a bigger fraction of the apps are shared by many users) could possibly yield better results. The improvement based on the 12-permission model suggests that simple profiles based on aggregating user decisions along each of the 12 permissions provide additional discriminative power. Intuitively, this amounts to differentiating between different groups of users who may be more or less comfortable granting different combinations of permissions across many apps. (e.g. people who have a problem disclosing their location versus people who do not mind).

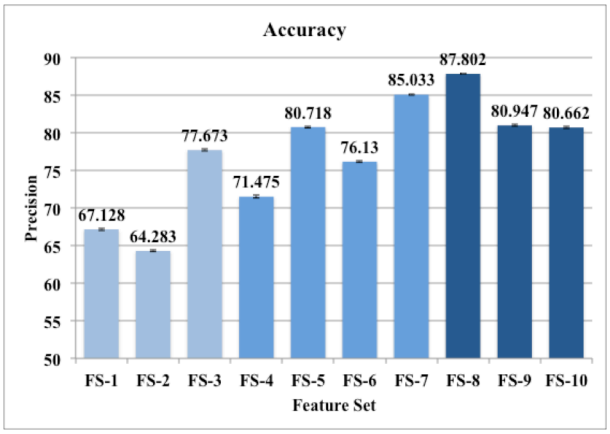

Figure 3. Accuracy of Predictions

4.1.5 Evaluating Interactive Scenarios

While 87.8% accuracy is promising, it is easy to imagine that even higher accuracy could possibly be achieved if one could single out predictions that have a relatively low level of confidence and just ask users to manually make those decisions. This observation opens the door to the evaluation of more interactive scenarios and the exploration of tradeoffs between accuracy and the number of decisions where we might want to query the user – in other words tradeoffs between accuracy and user burden. While it is unrealistic to expect users to want to manually specify decisions on over 60 permissions (average of over 20 apps per user and over 3 permissions per app), it is not unreasonable to think that users might be willing to enter 5 to 10 decisions. In theory, if users were ready to manually enter all 60 decisions, one could theoretically reach 100% accuracy. The question is how much accuracy do we lose by requesting users to only provide a fraction of these decisions.

Results presented in this subsection were obtained using the LibLinear tool for large-scale classification already mentioned in subsection 4.1.3. We use L-2 loss logistic regression from LibLinear and compute labeling confidence measures for each test data point. The classifier provides the same accuracy as that reported for FS-8 in Figure 4 (87.8%) while also estimating the probability of each class label.

Accordingly, we can compute the confidence of a given labeling decision as

$$Confidence = |Prob(Label = +1) - Prob(Label = -1)|$$

where the predicted label is the one that has the higher probability, either +1 ("Allow") or -1 ("Deny") A threshold can then be selected, where if the confidence of a labeling decision falls below that threshold, the user is queried. A lower threshold simply results in lower user burden but also lower accuracy, whereas a higher threshold results in more user queries but also a higher level of accuracy.

Results obtained by varying the threshold level and adjusting the percentage of decisions (or "data points") where the user is queried (horizontal axis) are presented in Figure 4. Again, these results are obtained using 10-fold cross validation. Figure 5 plots precision on "unlabeled data", namely on those decisions where we do not query the user, as well as overall precision,

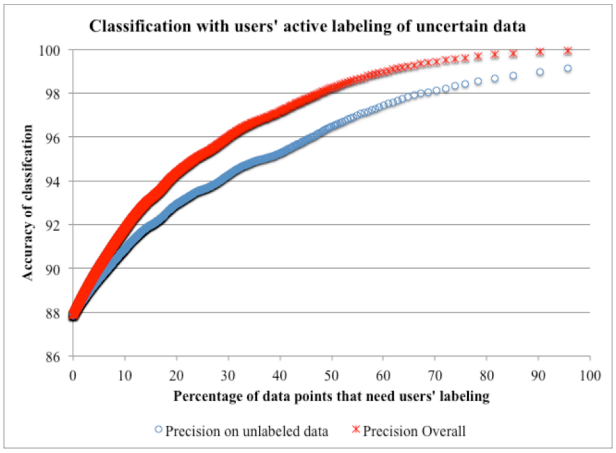

Figure 4. Classification with uncertainty

namely combining both predictions made by the classifier when confidence is above the threshold and predictions made by the user when confidence is below the threshold. We assume that, by definition, querying the user has 100% accuracy.

As can be seen, when asking users to make just 10% of the permission decisions, overall accuracy climbs from 87.8% to 91.8%. Given that users have already installed 4 applications out of an average of about 20 and that an app requires an average of 3 permissions, this simply amounts to asking users to provide 5 permission decisions (10% of 48 app-permission pairs). If users were willing to answer 10 permission decisions, overall accuracy would jump over 94%.

4.2 Simplifying Privacy Decisions Using Privacy Profiles

In our prior work in the context of location privacy we found that, while people's privacy preferences are often complex and diverse [5], a relatively small number of privacy profiles can be identified, which collectively do a good job at capturing these preferences [19, 18b, 22]. Each profile effectively corresponds to a different group or cluster of like-minded users and captures their privacy preferences. By asking users a few questions or presenting them with easy-to-understand descriptions of available profiles, it is possible to match individual users with profiles, In turn, these profiles can help predict with a high level of accuracy many of the users' location privacy preferences. A major motivation for our study of the LBE dataset is to determine to what extent mobile app privacy preferences, as captured in this dataset, exhibit similar patterns, namely to what extent a relatively small number of privacy profiles could be identified to simplify app permission decisions.

4.2.1 Generating Privacy Profiles by Clustering Like-Minded Users

Each user can be modeled as vector of app-permission decisions. As already discussed in subsection 4.1.4 such vectors are very sparse and did not yield the best predictive performance in our tests (see Fig. 3). Instead, aggregation of user preferences along each of the 12 permissions in the LBE dataset was shown to yield greater performance (FS-8). Accordingly, we represent each user as a 12-dimensional vector similar to the one used for Model 1 in subsection 4.1.3.

Using a K-means algorithm with Euclidean distance, we proceed to identify clusters of users. This is done using the standard "cluster" toolbox in R for our implementation.

4.2.2 Interpreting the Resulting Privacy Profiles

Before discussing the results of our analysis, we need to briefly introduce a few metrics. We start with a "discriminative" metric intended to help capture those most salient permissions or pairs of permissions characterizing a given cluster. We then proceed to also introduce three metrics intended to help us evaluate the benefits of using different numbers of privacy profiles (or user clusters).

We represent a user's decision on whether or not to grant a permission to a given app as a variable d $\in \{-1,+1\}$, where "+1 denotes "Allow" and -1 "Deny". For each permission p, user u and decision d, we define $S(u,p,d)$ as the number of instances that the user u has assigned decision d to permission p. We also define $S(u,p) = S(u,p,+1) + S(u,p,-1)$, namely

the total number of decisions on permission p made by user u. For each permission p, decision d and privacy profile C, we define $A(C,p,d)$ (in range [0,1]) as the average users' agreement in privacy profile C on assigning decision d to permission p:

$$A(C,p,d) = \frac{\sum_{u \in C} S(u,p,d)/S(u,p)}{\sum_{u \in C} 1}$$

We can now introduce a **discriminative** score for permission p in privacy profile C as:

$$Disc(p,C) = max_d\left(\frac{\sum_{C' \neq C}\left(A(C,p,d) - A(C',p,d)\right)}{K-1}\right)$$

For example, if we have 3 privacy profiles, and 99% of users in one of the profiles agree to deny access to the phone's location (across all apps), while 5% and 3% of the users in the other two profiles respectively agree to deny it, then we claim that the "Denying access to location" permission has a discriminative score of 95%[2].

Similar discriminative metrics can be computed for permission pairs. Below, when characterizing privacy profiles, we rely on single permissions and permission pairs with the highest discriminative scores, showing those five permissions and/or permission pairs with the highest score. Sample descriptions of K privacy profiles (for K=3 and K= 6) are shown in Figure 6 and Figure 7.

4.2.3 How Many Privacy Profiles Do We Need?

We now turn our attention to determining a good value of K, namely the number of clusters or privacy profiles to rely on. In comparing different values of K, we consider three distinct metrics.

(1) Precision of predicting default settings for users

As stated earlier, an important objective of our work is to determine to what extent a small collection of profiles can collectively help achieve a high level of accuracy.

To this end, we re-run the classification task while replacing the identities of users with their cluster membership. The resulting loss in accuracy will tell us to what extent the profiles are collectively capturing the complexity and diversity of privacy preferences of our user population. We use the same 10-fold cross validation procedure discussed in section 4.1.2. We denote the average precision as $\boldsymbol{Prec(K)}$**.**

(2) Interpretability & understandability

This is a more subjective metric. Here as we vary the number of clusters *(K)*, we want to know to what extent we can still identify a small number of features that can be used to characterize each cluster. The idea is that these compact descriptions could possibly be presented to users who would then identify which profile best matches their preferences – based on a relatively small (and hence understandable) number of features. An alternative approach might be to simply ask each user discriminative questions to determine which cluster best captures their privacy preferences.

While this measure is more subjective, for the sake of providing a comprehensive analysis, we define an interpretability score $\boldsymbol{Interp(K)}$ as:

$$Interp(K) = \frac{\sum_C max_p\{Disc(p,C)\}}{\sum_C 1}$$

In short highly discriminative features contribute to the interpretability of clusters. As indicated earlier, the thinking is that clusters that are easy to interpret would also make it easier for users to identify which cluster best matches their preferences.. An alternative scenario might involve asking users discriminative questions to identify clusters that best match their preferences.

(3) Stability of privacy profiles

Stability is yet another desirable attribute of clusters. We do not want our privacy profiles to change in response to small perturbation in the data. We compute a stability metric based on the following algorithm. Given a collection of privacy profiles obtained for a given value of K, we randomly split all the users into 10 folds of equal size. We then use each possible combination of 9 folds (of users) as training data for our K-means algorithm (the same algorithm already introduced in Section 4.2.1). For each combination of 9 folds, we use the resulting cluster centers to re-label all the users.

[2] Max {½ [(1-95)+(1-97)], ½ [(99-5)+(99-3)]}=95

This gives us two sets of cluster labels for the same group of users: the original labels and the ones obtained from the relabeling. We use maximum-weight matching of bipartite graphs to find the mapping between the two sets of clusters. A stability score can then be computed as the percentage of users who remain in the same cluster. The stability score of the original privacy profiles obtained for a given value of K, denoted $\boldsymbol{Stab(K)}$, can in turn be defined as the average stability score taken across all combinations of 9 folds.

Accordingly, we can also define the **adjusted precision** of privacy profiles $APrec(K)$ as:

$$APrec(K) = Prec(K) \cdot Stab(K) + Prec \cdot (1 - Stab(K)) ,$$

where $Prec$ is the average precision of the classifier regardless of cluster membership information of users. In other words, it means the adjusted accuracy would be the combined measurement of accuracy among users who remain in same cluster and average accuracy regardless of privacy profiles among users who fail to have a stable cluster membership.

Finally, in an attempt to summarize all three of these metrics as a single one, we also compute an overall score of for each value of K, as:

$$Overall\_Score(K) = \frac{2 \cdot APrec(K) \cdot Interp(K)}{APrec(K) + Interp(K)}$$

While imperfect, this final metric can help us compare the benefits associated with different numbers of clusters (namely different values of K). Results obtained using the LBE Data set, including all four metrics are shown in Figure 5.

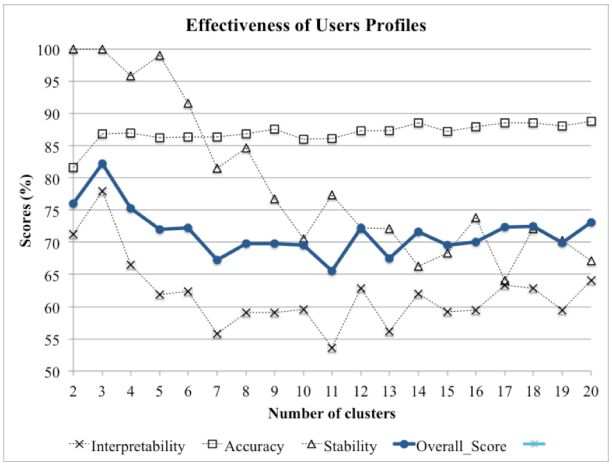

Figure 5. Effectiveness of Privacy Profiles

From the results in Figure 5, we can see that:

- As the number of clusters increases, classification accuracy gradually increases too, especially for $K \geq 3$. The increased complexity of the privacy profiles helps better distinguish between users with different preferences. However, the accuracy of this form of light-weight personalization theoretically converges to that of the fully personalized method captured with feature set FS-10 in section 4.1.4. Results shown in Figure 5 indicate that somewhat similar performance can be achieved with values of K as low as 3.
- As one would expect, the stability of our clusters moves in the opposite direction, decreasing as the number of clusters increases.
- The interpretability scores of privacy profiles fluctuate as K changes, with a rapid drop beyond K=3, as the clusters become finer and more difficult to articulate. At the same time, we believe that this metric should be taken "with a grain of salt". User studies would be needed to better evaluate this issue. In addition, it is likely that a simple wizard could easily be built to sort people among a set of available clusters/profiles by asking them a small number of questions. A user who has already installed a number of applications and configured permissions for these applications could possibly be classified as falling in a given cluster without even having to answer a single question.

If one is to naively follow these scores, one would conclude that a solution with just 3 clusters might be optimal. For reasons just explained above, we are inclined to believe that the interpretability metric used above is somewhat simplistic and that usable solutions could be developed for somewhat higher values of K,, which in turn could yield higher accuracy levels.

Fig 6 and 7 provide discriminative descriptions of profiles/clusters for K=3 and K=6. These discriminative features could provide a basis for asking a few questions to users and determine in which cluster they fall.

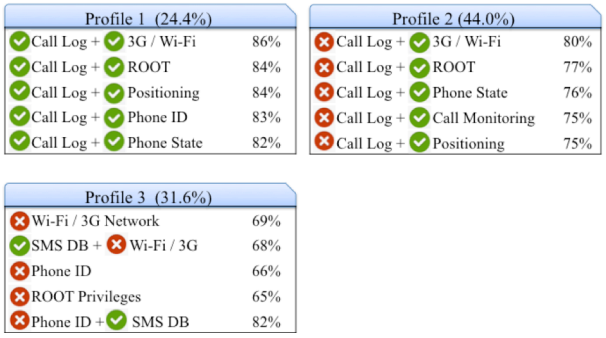

Figure 6. Discriminative Descriptions of Privacy Profiles (K=3)

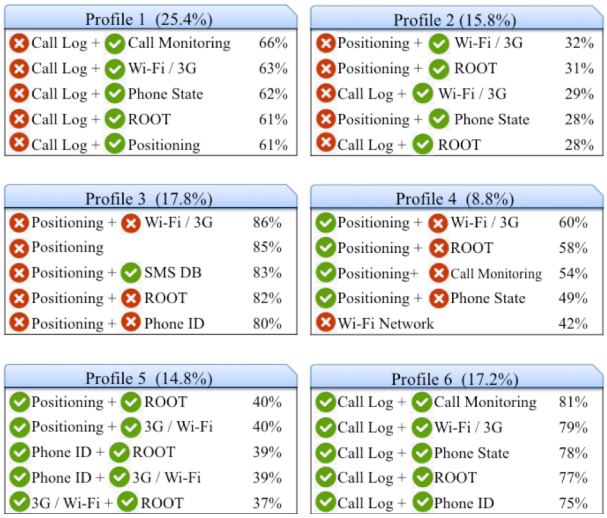

Figure 7. Discriminative Descriptions of Privacy Profiles (K=6)

Beyond the discriminative features depicted in Fig. 6 and 7, it is possible to also visualize and compare different privacy profiles using different color schemes. Figure 8 shows such a representation for scenarios where K=3, K=4, K=5, and K=6. Each cluster is represented by a 12-dimensional vector, with each cell colored according the cluster's propensity to allow or deny the corresponding permission. Dark blue denotes a strong propensity to grant the permission, dark red one to deny, while white denotes a split population – or at least a population whose decisions range about evenly between "allow" and "deny" across all mobile apps. Judging solely from the color schemes, one would conclude that clusters for K=3, 4 and 5 are very distinct, whereas the value of adding a 6th cluster (K=6) is starting to become less obvious. All scenarios seem to have one cluster that is particularly conservative when it comes to granting permissions (C3 for K=3, C4 for K=4, C5 for K=5, and C3 for K=6). Starting with K=6, a second conservative cluster (C4) is starting to emerge, though its population is not quite as reticent as that in C3., In general, we see that some clusters of users appear rather lenient, while others are more conservative. As the number of clusters increases, the nuances become finer. Some permissions also seem to yield more diverging preferences than others. For instance, looking at Figure 7, it can be seen that "Positioning" elicits very different reactions in clusters C3/Profile3 and C4/Profile 4, for K=6.

Figure 9 shows the variances of user privacy preferences for each permission in each profile for different values of K. As can be expected, variance tends to decrease as the number of clusters or profiles increases. For K=1, namely a single one-size-fits-all profile, the average variance of all permissions is 0.511. In contrast, for K=5 (namely 5 profiles), the average variance drops l to 0.216.

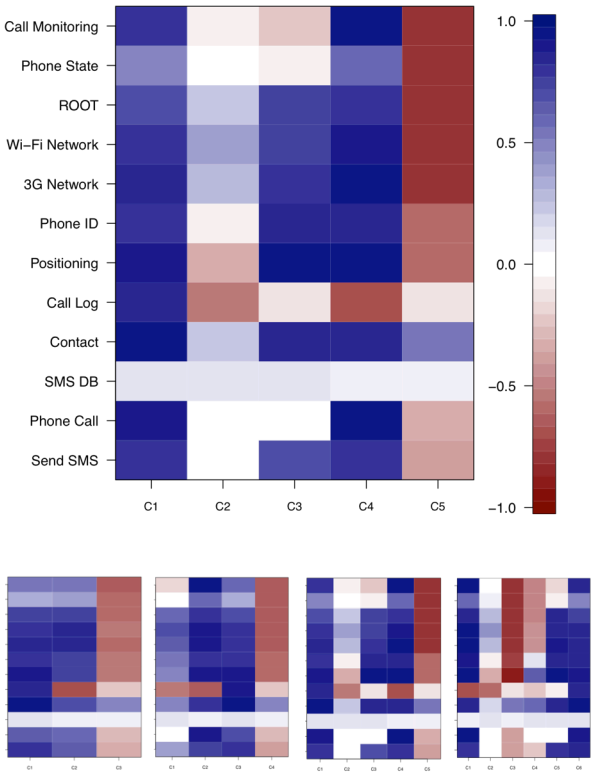

Figure 8. Colored Matrix Map of Average Preferences in Each Privacy Profile

(Top: K=5; Bottom (From left to right): K=3, K=4, K=5 and K=6. The color represents the average preferences of users in the corresponding cluster on the permission. For example, if a cell in the matrix has a value close to -1, then most of the users in the cluster can be expected to deny access to the corresponding permission)

Figure 9. Variances of preferences in each privacy profile in Figure 8

(Top: K=5; Bottom (From left to right): K=1(which indicates the condition that no profiles are available for users), K=3, K=4 and K=6.)

5. CONCLUDING REMARKS

Results from this study suggest that it is possible to significantly reduce user burden while allowing users to better control their mobile app permissions. In particular, we have shown that simple personalized classifiers could be built to predict a user's app permission decisions. In the scenario we considered, we assumed that a user would install and configure a first small number of apps and showed that, using permission decisions made by the user for these apps along with app-permission decisions from a representative population of users, it is possible to predict other permission decisions with a high level of accuracy (over 87%). We proceeded to show that by selectively asking users to manually make decisions on permissions where confidence in our prediction is below a certain threshold, accuracy could climb above 90% - as high as 94% if one is willing to ask for user input on 20% of permission decisions (92% if we limit ourselves to 10% of these decisions). We view these results as particularly encouraging as they offer the prospect of significantly alleviating user burden.

A closely related approach to reducing user burden involves the identification of privacy profiles. Just as our research in location privacy had shown [18, 19b, 22], a relatively small number of privacy profiles can go a long way in organizing people into groups of like-minded users and help predict their permission preferences. While it is too early to claim victory, we believe that this research opens the door to the design of significantly simpler mobile app privacy interfaces, where users do not need to give up control in return for usability. Further research in this area will require refining the techniques and scenarios introduced in this paper as well as exploring ways of combining these techniques. Ultimately human subject experiments will be required to evaluate how users respond to these interfaces in the wild.

There are always potential limitations associated with the type of analysis presented here. In particular, the dataset we obtained is from LBE users with rooted Android devices. While this user population is large, it is likely that it is not identical to the overall smartphone population at large. This could mean that a more representative segment of the population might potentially lead to slightly different clusters and slightly different preferences. We are inclined to believe however that, given the large LBE user base used in this study, we would see fairly similar types of profiles and, in particular, we would continue to see that a relatively small number of privacy profiles could go a long way in capturing the permissions of a user population with rather diverse preferences. As pointed out earlier, we also understand the limitations of the machine learning techniques used in this analysis and suspect that even stronger results could be obtained by refining and combining some of the techniques we introduced.

6. ACKNOWLEDGEMENTS

This research was supported in part by the National Science Foundation under grants CNS-1012763 and CNS-1330596 and by Google in the form of an unrestricted grant to the Mobile Commerce Laboratory. The authors would like to thank LBE Privacy Guard for sharing with them the data analyzed as part of this study.

7. REFERENCES

[1] *LIBLINEAR -- A Library for Large Linear Classification*, 2013.

[2] Y. AGARWAL and M. HALL, *ProtectMyPrivacy: detecting and mitigating privacy leaks on iOS devices using crowdsourcing*, *MobiSys*, ACM, Taipei, Taiwan, 2013, pp. 97-110.

[3] R. AMADEO, *App Ops: Android 4.3's Hidden App Permission Manager, Control Permissions for Individual Apps!* , 2013.

[4] J. BAGLAMA and L. REICHEL, *Augmented Implicitly Restarted Lanczos Bidiagonalization Methods*, SIAM Journal on Scientific Computing, 27 (2005), pp. 19-42.

[5] M. BENISCH, P. KELLEY, N. SADEH and L. CRANOR, *Capturing location-privacy preferences: quantifying accuracy and user-burden tradeoffs*, Personal and Ubiquitous Computing (2010).

[6] A. BERESFORD, A. RICE and N. SOHAN, *MockDroid: trading privacy for application functionality on smartphones*, *HotMobile*, Phoenix, AZ, USA, 2011.

[7] T. BOOK, A. PRIDGEN and D. S. WALLACH, *Longitudinal Analysis of Android Ad Library Permissions*, CoRR, abs/1303.0857 (2013).

[8] E. CHIN, A. P. FELT, V. SEKAR and D. WAGNER, *Measuring User Confidence in Smartphone Security and Privacy*, *Soups*, 2012.

[9] J. CRANSHAW, J. MUGAN and N. SADEH, *User-Controllable Learning of Location Privacy Policies with Gaussian Mixture Models*, *AAAI*, 2011.

[10] W. ENCK, P. GILBERT, B.-G. CHUN, L. COX, J. JUNG, P. MCDANIEL and A. SHETH, *TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones*, *OSDI* Vancouver, BC, USA, 2010.

[11] L. FANG and K. LEFEVRE, *Privacy wizards for social networking sites*, *www*, ACM, Raleigh, North Carolina, USA, 2010, pp. 351-360.

[12] A. P. FELT, E. CHIN, S. HANNA, D. SONG and D. WAGNER, *Android permissions demystified*, *CCS*, ACM, Chicago, Illinois, USA, 2011, pp. 627-638.

[13] A. P. FELT, S. EGELMAN and D. WAGNER, *I've Got 99 Problems, But Vibration Ain't One: A Survey of Smartphone Users' Concerns*, *SPSM*, 2012.

[14] A. P. FELT, E. HA, S. EGELMAN, A. HANEY, E. CHIN and D. WAGNER, *Android Permissions: User Attention, Comprehension, and Behavior*, *Soups*, 2012.

[15] M. FRANK, D. BEN, A. P. FELT and D. SONG, *Mining Permission Request Patterns from Android and Facebook Applications*, *ICDM*, 2012, pp. 870-875.

[16] Y. GUO-XUN, C. H. HO and L. CHIH-JEN, *Recent Advances of Large-Scale Linear Classification*, Proceedings of the IEEE, 100 (2012), pp. 2584-2603.

[17] 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*, *CCS*, ACM, Chicago, Illinois, USA, 2011, pp. 639-652.

[18] J. JUNG, S. HAN and D. WETHERALL, *Short paper: enhancing mobile application permissions with runtime feedback and constraints*, *SPSM*, ACM, Raleigh, North Carolina, USA, 2012, pp. 45-50.

[19] 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*, *USEC*, 2012.
[20] P. G. KELLEY, P. H. DRIELSMA, N. SADEH and L. F. CRANOR, *User-controllable learning of security and privacy policies*, *Proceedings of the 1st ACM workshop on Workshop on AISec*, ACM, Alexandria, Virginia, USA, 2008, pp. 11-18.
[21] LBE, *LBE Privacy Guard*, 2013.
[22] J. LIN, S. AMINI, J. HONG, N. SADEH, J. LINDQVIST and JOY ZHANG, *Expectation and Purpose: Understanding Users' Mental Models of Mobile App Privacy through Crowdsourcing*, *Ubicomp'12*, Pittsburgh, U.S., 2012.
[23] J. LIN, G. XIANG, J. I. HONG and N. SADEH, *Modeling people's place naming preferences in location sharing*, *UbiComp*, ACM, Copenhagen, Denmark, 2010, pp. 75-84.
[24] I. LUNDEN, *U.S. Consumers Avg App Downloads Up 28% To 41; 4 Of 5 Most Popular Belong To Google*, 2012.
[25] J. MUGAN, T. SHARMA and N. SADEH, *Understandable Learning of Privacy Preferences Through Default Personas and Suggestions*, under review (2012).
[26] M. NAUMAN, S. KHAN and X. ZHANG, *Apex: extending Android permission model and enforcement with user-defined runtime constraints*, *ASIACCS*, ACM, Beijing, China, 2010, pp. 328-332.
[27] R, *irlba: Fast partial SVD by implicitly-restarted Lanczos bidiagonalization*, 2012.
[28] R. RAVICHANDRAN, M. BENISCH, P. G. KELLEY and N. M. SADEH, *Capturing Social Networking Privacy Preferences*, *Proceedings of the 9th International Symposium on Privacy Enhancing Technologies*, Springer-Verlag, Seattle, WA, 2009, pp. 1-18.
[29] N. SADEH, J. HONG, L. CRANOR, I. FETTE, P. KELLEY, M. PRABAKER and J. RAO, *Understanding and Capturing People's Privacy Policies in a Mobile Social Networking Application*, The Journal of Personal and Ubiquitous Computing (2009).
[30] S. THURM and Y. I. KANE, *Your Apps are Watching You*, WSJ (2011).
[31] W. VERDUZCO, *App Ops Brings Granular Permissions Control to Android 4.3*, 2013.
[32] S. WILSON, J. CRANSHAW, N. SADEH, A. ACQUISTI, L. F. CRANOR, J. SPRINGFIELD, S. Y. JEONG and A. BALASUBRAMANIAN, *Privacy Manipulation and Acclimation in a Location Sharing Application*, *Ubicomp*, 2013.
[33] Y. ZHOU, X. ZHANG, X. JIANG and V. W. FREECH, *Taming Information-Stealing Smartphone Applications (on Android)*, *TRUST*, 2011.