NTNU
Norwegian University of
Science and Technology

Privacy services for mobile devices

Solvår Bø
Stian Rene Pedersen

Master of Science in Communication Technology
Submission date: June 2011
Supervisor: Svein Johan Knapskog, ITEM
Co-supervisor: Åsmund Ahlmann Nyre, SINTEF ICT
Karin Bernsmed, SINTEF ICT

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Smartphones with third-party applications are becoming increasingly popular. Recently, they have received attention for quietly monitoring and transferring personal information without users' knowledge. Therefore, it is virtually impossible, even for privacy-conscious users, to properly protect their privacy.

The objective of this thesis is to help users to protect their privacy by increasing their consciousness on how personal information is collected and distributed.

The tasks foreseen in this assignment include:

1. Review existing work concerning privacy on mobile devices.
2. Sketch a design that meets the above objective.
3. Implement selected parts of the design to demonstrate its functionality.

Assignment Given:	24th of January 2011
Supervisor:	Karin Bernsmed and Åsmund A. Nyre
Professor:	Svein J. Knapskog

Abstract

Recent studies have shown that privacy on mobile devices is not properly ensured. Due to a heavy increase of smartphones in the market, in addition to a variety of third-party applications, a demand for improved solutions concerning privacy has arisen. Our objective is to extend users' ability to control applications' access to resources at run-time. We investigate whether such a solution is adequate or not, in order to properly maintain privacy.

We propose a design that provides a higher degree of control by allowing users to set preferences that determines what personal information to share. Previous efforts only give users a binary choice on whether to fake personal information or not. We offer a more flexible solution that allows users to set preferences with a higher degree of granularity. We implement selected parts from our design, in order to evaluate whether this solution serves as a utility or not. Further evaluation is a necessity in order to fully accept or reject the idea. However, our initial results are promising.

Preface

This report serves as a master thesis in Information Security in the 10th semester of the Master's Program in Communication Technology at The Norwegian University of Science and Technology, NTNU. SINTEF ICT gave the assignment.

We would like to give our supervisors, Karin Bernsmed and Åsmund Ahlmann Nyre, a generous reward for their irreproachable input and weekly feedback. We appreciate their continuously ongoing enthusiasm throughout this project. Their contributions and insightful ideas have also been of great value to us. We would also like to give gratitude to Professor Svein J. Knapskog for guidance throughout this thesis.

Trondheim, June 10, 2011
Stian Pedersen and Solvår Bø

Abbreviations

3G 3rd generation mobile telecommunications

ADT Android Development Tool

AP Access Permission

API Application Program Interface

apk Android Package file format

Apps Application programs

CVS Current Versions System

DVM Dalvik Virtual Machine

EDGE Enhanced Data rates for GSM Evolution

FGPS Fine (GPS) Location

FIA Full Internet Access

GPS Global Positioning System

GPRS General Packet Radio Service

GSM Global System for Mobile Communications(Groupe Spécial Mobile)

GSMA GSM Association

GUI Graphical User Interface

IDE Integrated Development Environments

iOS iPhone Operating System

J2ME Java 2 Micro Edition

JDK Java Development Kit

JVM Java Virtual Machine

MMS Multimedia Messaging Service

OS Operating System

PC Personal Computer

RCA Read Contact Data

SD card Secure Digital Memory Card

SDK Software Development Kit

SMS Short Message Service

SQLite Light version of Structured Query Language

SWAAID Show Widget and Allow After Input and Delay

UI User Interface

URL Uniform Resource Locator

Wi-Fi Trademark of the Wi-Fi Alliance (Wireless Fidelity)

XML Extensible Markup Language

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mobile phones have become a central part of our lives and we bring them wherever we go. Today, the smartphone is a fully-fledged computer and it carries a lot of personal data like call logs, emails, SMS and address book. The phone is also used to store and carry documents, and to access corporate networks. This makes the smartphone even more vulnerable to privacy invasions than traditional computers[Bon10].

Third-party applications for smartphones have become very popular in the last couple of years. These applications, often referred to as 'apps', are software packages made by a independent third-party developers. The user may purchase and download apps from online markets. Apps make use of the phone's resources to provide different functionalities and services to the user. By resources we mean the GPS, Internet access, contact list, calendar, phone- status and information storage, email accounts and camera, among others.

Lately, third party applications have received some attention in media with regards to their lack of privacy[23]. An extensive survey done by the Wall Street Journal[24] revealed that approximately 50% of the top 100 applications for Iphone and Android collect information about the users and their habits, without the users' consent.

1.1 Privacy on mobile devices

Vendors of smartphone Operating System(OS)[1] are aware of the potential privacy risks related to third party applications. There are different techniques to reduce this risk. Nokia and Apple use a technique called *Application Vetting*. The developers submit the application to a trusted party for testing, and the application has to be approved in order to be published. The goal of this process is to filter out applications with malicious behavior, but this technique has suffered from problems with both false accept and false reject.

[1]Examples of vendors of smartphone OS are Google, Apple and Microsoft

Android and the J2ME platform use an alternative approach to vetting, which is *mandatory access control*. With mandatory access control the application developers have to present a list of requested access permissions to the resources the application intend to use. These permissions will then be presented to the user upon installation. With Android, the user must either accept or reject the entire set of permissions. This means that the user cannot reject some of the permissions and accept others. If the list of requested permissions is rejected, the application will not be installed [BRSS11].

1.2 Motivation

In an effort to meet the discovered privacy risk in relation to third party applications, GSMA has published Mobile Privacy Principles[21]. The principles describe how the user's privacy should be respected and protected by applications that have access to personal information. Amongst others, three of the principles state that:

- Users shall be given opportunities to exercise meaningful choice, and control over their personal information.
- Users should be provided with information about, and an easy means to exercise, their rights over the use of their personal information.
- Users should be provided with information about privacy and security issues and ways to manage and protect their privacy.

These principles are concerned with providing users with sufficient information regarding their privacy. In addition, developers should provide users with proper tools in order to manage their personal information. In a press release in relation to the privacy principles[25], GSMA stated that 'the key challenge is to find new and mobile-friendly methods to help consumers make informed decisions about their privacy'. This challenge, together with the three principles stated above, forms the basis for our motivation.

The main problem with Android's mandatory access control is that the user has to give permissions upon installation. The user has no control of when and why the application needs access, or how the application intends to use the collected data. Also, some of the listed access permissions may be difficult to understand. In web services, the intended use of the personal collected data has to be listed in a privacy policy defined by the service-provider. In many cases, before users can install software or register with a service, the users have to read and accept these policies. Even though this practice also has some limitations, it still explains the intended use of collected data. This practice is not mandatory for developers of smartphone applications. Some of the application developers for mobile devices have a privacy policy, but the users have to search the web on their own initiative in order to find it.

The mandatory access control in Android is the only implemented privacy feature with regards to third-party applications, as far as we are concerned. Once installed, the application can access the requested resources at any time, without the users knowledge [BRSS11].

Our intention is to provide a solution that give users the ability to control and modify these access permissions after the application is installed.

1.3 Objective

1.3.1 Main objective

The main objective in this thesis is *to help users protect their privacy by increasing their consciousness on how personal information is collected and distributed.*

This will be done by a software-tool with the ability to control and modify access permissions after an application has been installed.

Privacy concerns arise whenever personal information is collected and stored. In order to meet these concerns, we will focus on *how to control* the collection of personal information. Other aspect of privacy, such as anonymity, would not be covered to the same extent.

1.3.2 Research Questions

Based on the main objective we identified two research questions:

- How can we give users an increased consciousness on how personal information is collected and distributed from their mobile devices?
- Can we propose a software-design that would help users protect their privacy?

1.4 Research method

In order to answer the research questions, we will undertake the tasks given in the problem description:

- Review existing work concerning privacy on mobile devices.
- Sketch a design that meets the main objective in this thesis.
- Implement selected parts of the design to demonstrate its functionality.

This master thesis is based on the research method *design science*. In design science the fundamental question to answer is 'What utility does the artifact provide, and what demonstrates the utility?'[HMPR04]. In our context, Hevner defines an artifact as 'constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations (implemented and prototype systems)'[HMPR04].

Design science research requires the creation of an innovative and purposeful artifact for a specific problem domain. In our approach, the artifact would be a design model. We would implement and evaluate selected parts of the design to demonstrate its utility, and whether it is a contribution to the area of privacy on mobile devices or not.

We have chosen Android as our preferred development platform. The reason for choosing Android instead of other competitive platforms, such as iOS (iPhone), Symbian (Nokia) or Windows Mobile, was due to the increasing popularity and extensive documentation available for Android developers.

To evaluate the functionality of the implemented parts, we would perform a descriptive test based on a set of scenarios. Further, to evaluate the usability and usefulness of the solution, an experimental test on a group of test participants would be carried out.

1.5 Outline

Chapter 2 gives an overview of the Android platform, along with a brief description of how privacy and security is maintained in the Android Operating System. If the reader in advance feels confident on how Android is compounded, the reader is urged to continue to Chapter 3.

An overview of relevant research is given in Chapter 3.

Chapter 4 continues with a description of our proposed design.

In Chapter 5, we describe the implementation of selected parts of our design.

Chapter 6 covers our evaluation of the prototype and design, done by a functionality test and a usability test.

In Chapter 7, we first discuss various limitations and assumptions attached to our prototype and design in general. Next, we discuss some challenges and advantages, and compare our solution to related work.

Future work is covered in Chapter 8. In this chapter, we focus the attention towards potential improvements to the scope of our project that can be made in the future.

Chapter 9 wraps up our project in a conclusion.

Chapter 2

Background

This chapter first give a brief historical introduction to Android. Further, we touch upon the main components that combined make up the Android architecture. Next, the security model incorporated is also described. This model is inherited from Linux, which is the operating system Android is solely based on. In addition, a core and critical component of every Android application is introduced, namely the *manifest* file.

2.1 Introduction to Android

The mobile operating system *Android* was established in the spring of 2005. The creator, Andy Rubin, had a vision of a free, open-source platform that allowed for any coder to contribute[13]. He invented the Android software stack, consisting of an operating system and key mobile applications. After the coalition with Google in 2005, Android has gained a solid foothold in the market. A study conducted by *Canalys*[1] reported that Android was the best-selling smartphone operating system in the fourth quarter of 2010[18].

Android is licensed under the Apache License[17], which makes it open-source and therefore able to incorporate new technologies as they emerge. By allowing the community to contribute to a common pool of apps, serves as an enrichment and extends the functionality of the mobile devices. Apps are created using the Java programming language, and they can either be distributed through *Android Market*[20], or be downloaded directly from third-party sites. The operating system is based on a customized Linux kernel, and is built to be open and accessible. The operating system does not differentiate between applications created by third-party developers, and the devices' core applications. In turn, this allows developers to fully utilize the hardware and allow for users to tailor the phone to suit their interests and needs[16].

[1] *Canalys* is an independent analyst house focusing on technology. http://www.canalys.com

Android offers a variety of connectivity options such as WiFi, Bluetooth and wireless data over cellular networks (GPRS, EDGE and 3G). In addition of being highly connected to its surroundings, Android also offer the utilization of location-based services (such as GPS), accelerometers and camera. A simplified view of the software stack in Android is shown in Figure 2.1.

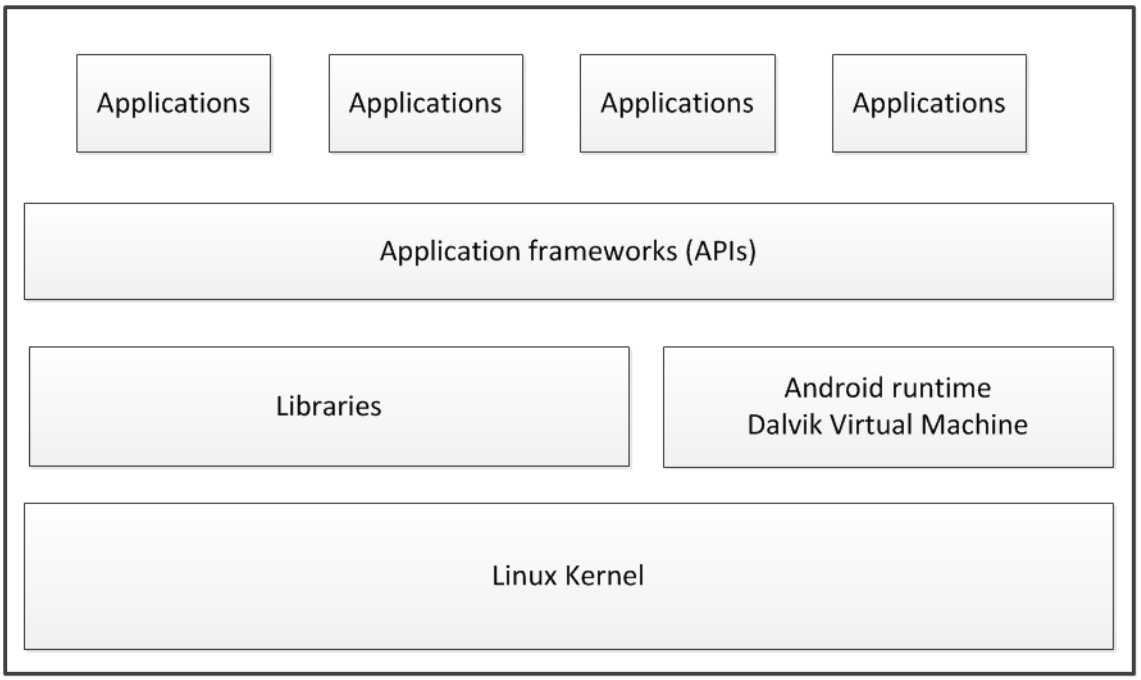


Figure 2.1: The Android software stack. [15]

A good reason for adopting Android as the development platform of choice is because of the ease-of-access to the underlying hardware of an Android device. Historically, the lack of access to the underlying hardware have been frustrating for mobile developers. However, since the Android platform is open source, anyone can access them. The number of available resources is obviously dependent on the device. They can easily be combined to perform different tasks, such as monitoring the environment and even sending email about it.

2.2 Android Architecture

As previously mentioned, the Android software stack is based on a customized Linux kernel. Linux processes are running within the kernel, and applications run in a virtual environment within the process. Software developed with Java is usually ran in a *Java Virtual Machine* (JVM). Even though software developed for Android is coded in Java, it is executed in the open-source *Dalvik Virtual Machine* (DVM), which is fully in accordance with the open-source nature of Android. Applications developed for Android is referred to as *packages*. Applications interact with hardware resources through the *Android API*[12]. The application architecture is illustrated in Figure 2.2.

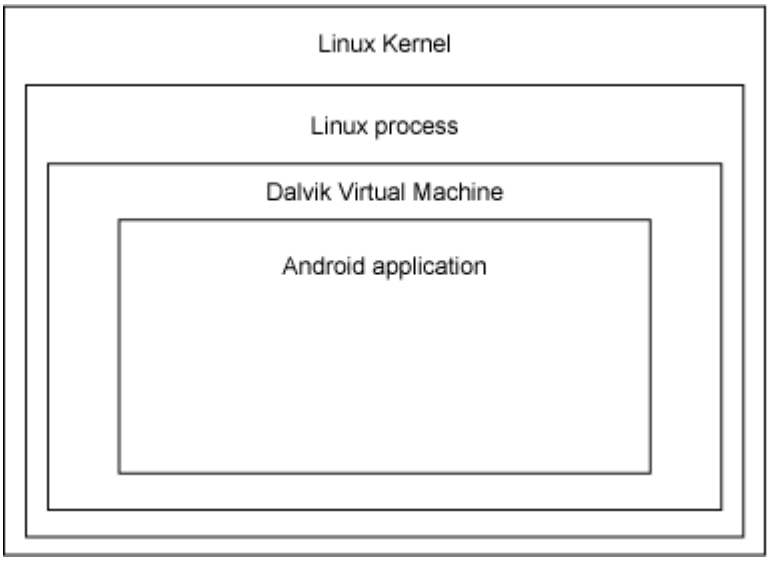


Figure 2.2: The Android architecture.[15]

Applications are built up from multiple *components*, where each component provides different functionality. As a consequence, these components have the need to communicate with each other. Components are able to interact by sending messages, called *intents*, to each other. The different component classifications are listed in Table 2.1.

Classification	Description
Activity	When the application is started from either the home screen or through the application manager, an *activity* is launched.
Service	A *service* component typically performs background processing such as WiFi monitoring, calculating etc. Descriptive for applications that need to persist for a long time.
Content Providers	*Content providers* are helping with data storage and retrieval facilities, such as providing access to persisted data.
Broadcast Receivers	Responsible of receiving messages from other applications.

Table 2.1: Android application classifications.[6]

Linux is known to assign a user ID to a specific user of the system, serving as the identification. The analogy is the same to Android, except that user IDs are used to identify applications and not individuals. Applications running on Android are isolated from one another and ran in individual sandboxes, as illustrated in Figure 2.3.

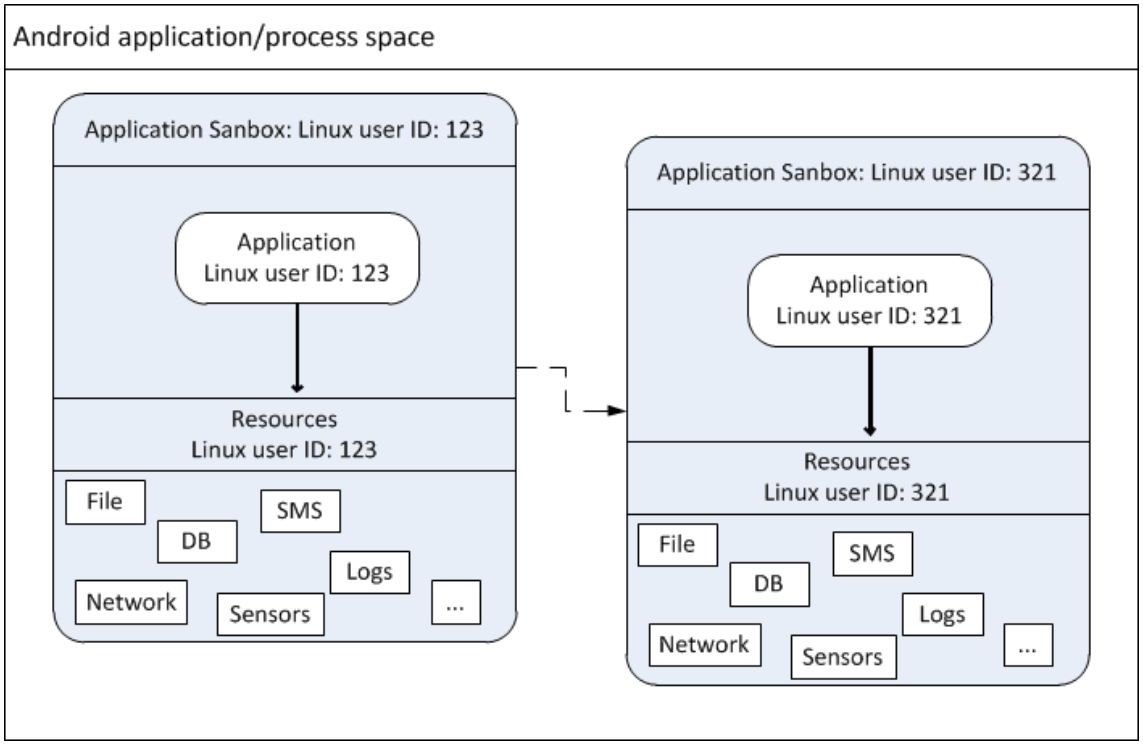


Figure 2.3: Two Android applications which resides within its own sandbox.[14]

2.3 Privacy and Security

Running each application in a sandbox enforces inter-application separation. In addition, applications have individual permissions to either allow or deny access to the device's resources. By default, each application does not have any permissions granted. This prevents applications from accessing resources. There are two ways of requesting access to the system or resources. One way is to assign the same user ID to two different applications[11], as shown in Figure 2.4.

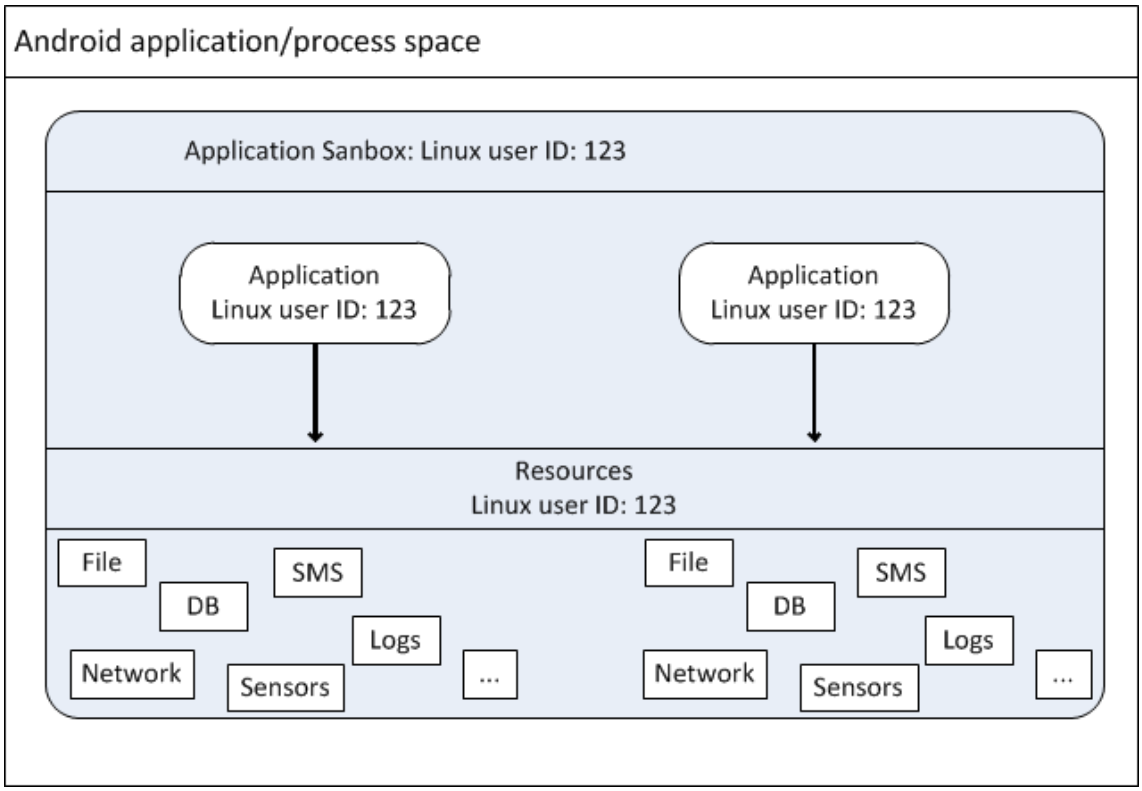


Figure 2.4: Two Android applications assigned with the same user ID. [14]

A different, and more common, way of doing it is to request permission to resources through the application's *manifest* file[10]. The manifest file is a XML file that contains essential information about the application. The Android OS have to know this information before it can execute the application. As mentioned in Chapter 1, this set of requested access permissions, listed in the *manifest* file, is presented to the user upon application installation. The user then has to accept all of the listed permissions to be able to install the application. It is worth mentioning that all applications must have a manifest file present in order to work.

Chapter 3

Related work

In this Chapter, we describe some of the previous research efforts, which are most relevant to the scope of our project. Some of the projects, such as MockDroid and TaintDroid, are similar to our approach, because they focus on the topic related to privacy for mobile devices. Others, like TightLip, do not mainly focus on mobile devices, but some of the ideas and findings in these approaches are still interesting. The most relevant work is described first.

3.1 The MockDroid Project

MockDroid is defined as 'a modified version of the Android operating system which allows a user to 'mock' an application's access to a resource.' [BRSS11] The user is still able to use the application, though with the lack of some functionality that is dependent on mocked data. One of the most important modifications is that the set of access permissions is duplicated so that each permission has both one real and one mocked version. Even if MockDroid is a development project, it still provide support to mock the coarse and fine-grained location, Internet connection, SMS, MMS, calendar, contacts, device ID and broadcast intents.

According to the creators of MockDroid, some of the benefits of using fake or mocked data are:

- Control over optional features.
- No unwanted sharing of personal data.
- Rather than providing the precise location of the phone to an application that shares such data with friends, the user might provide the location of the nearest city.
- Controlling expensive operations: E.g. preventing applications that constantly connect to the Internet (giving the user high 3G data connection costs) by denying the application Internet access.

By default, when an application is installed, there are no mocked permissions and all the permissions are granted as presented in the manifest file. To deny some of the access permissions an application has, the user have to use the MockDroid software called Mocker[BRSS11].

By using MockDroid, the user would be able to control the collection and distribution of personal data, which is a great contribution to privacy on mobile devices. However, as the title implies, by providing mocked data the applications may loose important functionality. For most users, it might be a problematic decision to sacrifice functionality to obtain privacy.

3.2 The TaintDroid Project

TaintDroid is an extension to the Android platform that assumes that downloaded third-party applications are not to be trusted. A tracking system monitors how these applications access and manipulate users' personal data in real-time. By labeling data from privacy-sensitive sources, TaintDroid detects when sensitive data leaves the system. If this labeled data is transmitted over the network, TaintDroid registers its label, which application it came from and the destination.

By utilizing TaintDroid in a test of randomly chosen Android applications, the findings revealed that two-thirds of the applications used sensitive data suspiciously. For instance, the test revealed that half of the studied applications exposed location data to advertisement servers without users awareness.[EGC+10]

TaintDroid is a great solution to understand how, and when, personal information is collected by third-party applications. However, it does not offer a solution on what the user ought to do about it. With respect to applications with malicious behavior, TaintDroid may have an impact on whether or not a user should un-install application. A disadvantage with TaintDroid is that even though applications are not acting suspiciously, they may have 'too much' access, and may continue to collect and distribute information.

3.3 Automating Privacy Testing of Smartphone Applications.

The authors aim to give the users increased control and understanding of their privacy on mobile devices. The AppInspector is a tool for testing apps to figure out if they are malicious or not. The tool gives users a warning if the application is suspicious prior to downloading it. In contrast to the other approaches mentioned in this chapter, the function of AppInspector is not to handle the privacy for installed applications on a mobile device. Instead, it should help users to better understand how these applications handle their privacy-sensitive information in advance. The intention is to allow for the user to make informed decisions about which applications to install or not. The challenge is then to get a good diagnostic of the malicious applications to be able to present correct and informative feedback to the user [GgCCJ11].

This is a good solution when it comes to understanding how malicious applications collect and distribute personal data. As mentioned in Chapter 1, the survey done by Wall Street Journal revealed that about 50% of the most popular applications collects and distributes information in ways that might be suspicious. The question is whether AppInspector would give a warning for every other application a user want to download. If that is the case, the warnings would soon enough turn out to be annoying, and not very striking. On the other hand, if AppInspector only give a warning for applications with malicious intentions, it would not cope with privacy concerns related to 'well intended' applications that have access to more resources than necessary.

3.4 What You See is What They Get

This paper introduces a Graphical User Interface (GUI) called *the sensor-access widget*. This widget shows an animation of how personal data is being collected from different sensors. The widget also has a feature for granting or denying applications' access to different sensors. It works in real-time, and will appear at the users desktop if an application try to gain access to either the Camera, location, microphone, accelerometer or thermometer.

The widget provides the policy *Show Widget and Allow After Input and Delay(SWAAID)*. This means that the user will notice a countdown timer, and will be given the choice to deny the application access during this countdown. If not, the application is granted access [HS10].

In the evaluation of this proposal the authors discusses different aspects of the widget. One important disadvantage is the impact on usability. This widget will, especially on mobile devices, consume a considerable portion of the screen. It might also be quite distracting for the user if the widget appears every time they try to use an application.

3.5 The TightLip Project

In [YMC07], *TightLip* is introduced. This is defined as 'a privacy management system that helps users define what data is sensitive and who is trusted to see it rather than forcing them to understand or predict how the interactions of their software packages can leak data.' This approach is concerned with data sharing in public spaces.

The intention of TightLip is to allow users to better manage their shared files and spaces. This is done by helping them define what data is important, and who to trust to access this data. The research consisted of three main challenges. First to define sensitive data and trusted hosts. Second, how to track the data through the system to identify potential leak of data from the system. Finally, developing policies to deal with the potential leaks.

In TightLip they used a mechanism called *doppelgangers*. Doppelgangers are sandboxed copy processes that inherit most, but not all, of the state of an original process. In TightLip the doppelgangers are created when a process accesses sensitive data. These doppelgangers follow the original processes through the system. If the process tries to leave the system, TightLip enables a policy module that handles the potential breaches. One solution is to swap the original process with the doppelganger. The process that leaves the system will then not include the sensitive information because the doppelgangers consist of *scrubbed data*. By scrubbed data, the authors claim that an amount of information has been removed (scrubbed away) from the original data, returning data free of sensitive information.

The use of doppelgangers is obviously dependent on a secure solution for scrubbing the data. If a part of the sensitive data still resides within the doppelganger, the intention for using TightLip fades away.

3.6 Results from the literature study

The literature studied conducted in this project have given us both ideas and inspiration, as well as a wider perspective regarding shortcomings in this area of research. The MockDroid project turned out to be most relevant for the development of our design and prototype. The findings in the MockDroid project is therefore used as the basis of our design, with the intention to go beyond the state of the art, and provide the user with a higher degree of flexibility compared to MockDroid's 'all or nothing' solution.

Chapter 4

Design

We have proposed a design based on both the motivation found in Chapter 1.2, and an analysis of the related work, as described in Chapter 3. In order to find a set of reasonable requirements for the design, we first thought of a couple of user-scenarios, as described in Chapter 4.1. The functional requirements for the design are coped with in Chapter 4.2. Finally, Chapter 4.3 includes a description of our design.

4.1 User scenarios

4.1.1 Introduction

To demonstrate the intention of our design, as well as identifying requirements, we have created a couple of scenarios with our main character Bob. Bob uses an Android smartphone in both business-related matters, as well as private. He uses his phone a lot, and he frequently downloads new content from Android Market[20]. Lately, Bob has become increasingly concerned with privacy on his phone. The newspapers have turned their attention to issues related to third-party applications. Bob felt he lost somewhat control over his phone, and found help by downloading the Android middleware named Privacy Service. This service includes a functional application named the Privacy Application.

4.1.2 Scenario 1

After Bob has installed and started the Privacy Application, he is met with a screen listing all of the installed applications on the device. He navigates through the list and chooses one of his favorite games, namely *MyTetris*. A new window is brought to the foreground, and a list of all the access permissions granted to MyTetris appears. These permissions were granted upon installation of MyTetris. A relatively simple game like MyTetris should be able to work just fine without any access permissions. However, MyTetris has been granted access to the fine (GPS) location, access to modify and delete content of his SD-card and full Internet access. Bob does not want the application to access his location nor the Internet, and clicks on a checkbox found next to the listing of these permissions.

Because Bob is engaged with his highscores, he leaves the checkbox next to 'SD Card' unchecked. Next time MyTetris requests access to resources, it would receive a fake location and no Internet connection.

Bob is playing the game for a while without noticing anything different. But when he reaches a higher score than ever before, he suddenly realizes he is not able to share his high score with his friends. Bob opens the Privacy Application to grant MyTetris access to the Internet. After he has shared the high score with his friends, he turns off the Internet access again.

The advantage of using the middleware, in this scenario, is that Bob is able to choose if and when an application may be granted access to a resource.

4.1.3 Scenario 2

Once more, Bob opens the Privacy Application, but this time he chooses an application named *MyDailyNews*. MyDailyNews feeds him with fresh news every day. From the list of its requested access permissions, Bob sees that MyDailyNews has access to his fine (GPS) location, in addition to unrestricted Internet access. He does not want to grant any access to this application, so he clicks on both of the checkboxes within the Privacy Application.

When Bob opens MyDailyNews again, he is met with an error message stating that the application would not work without access to the Internet. He goes back into the Privacy Application, and unchecks the checkbox next to 'full Internet access'. When he opens MyDailyNews the next time, the expected list of news is empty, and a message tells him to check whether his GPS is turned on or not.

Bob realize that he has to share his location to be able to use MyDailyNews, but he does not want to share his exact location. A compromise has to be made. Within Privacy Application, he decides to click on 'fine (GPS) location' to get further options. A new window reveals itself, and present Bob with detailed information about the 'fine (GPS) location' permission. He finally understands whatÕs meant with 'fine (GPS) location'. He suddenly understands the risks arisen from sharing his location. He clicks a 'settings' button located at the bottom of the screen to get more options to handle his situation.

A new window appears, and he sees a list of options from where he can chose the granularity of his GPS location. He figures that he hardly ever read the local news. His interests are in the regional and national news. He decides that MyDailyNews only needs to know which state, and country, he is resident in. He clicks the checkbox next to the 'state' option. When he now opens the MyDailyNews application, he does not receive local news.

There are two main advantages of using the middleware. First, the information window gives Bob a better understanding of the requested access permissions. Secondly, by navigating to the settings menu, he gets the flexibility to share 'some' information, compared to choosing 'all or nothing'.

4.2 Functional requirements

The main functional requirements to our *middleware* are to:

1. Allow users to control the sharing of personal information by having the ability to override access permissions granted to the application upon installation.
2. Extend the granularity of the permissions to include 'some' information, in addition to 'all or nothing'.
3. Give users an explanation of the consequences from granting an access permission.

4.3 Our design

Our design is based on a middleware solution, as illustrated in Figure 4.1. The middleware includes both an application and a service, named *Privacy Application* and *Privacy Service*, respectively. The Privacy Application is interacting with input from a user, while the Privacy Service performs tasks silently in the background. The relation between the Privacy Application and the Privacy Service is through a shared database. The middleware is place between the applications and the APIs in the software stack. Calls either from or to the API will be handled by the middleware by introducing this scheme.

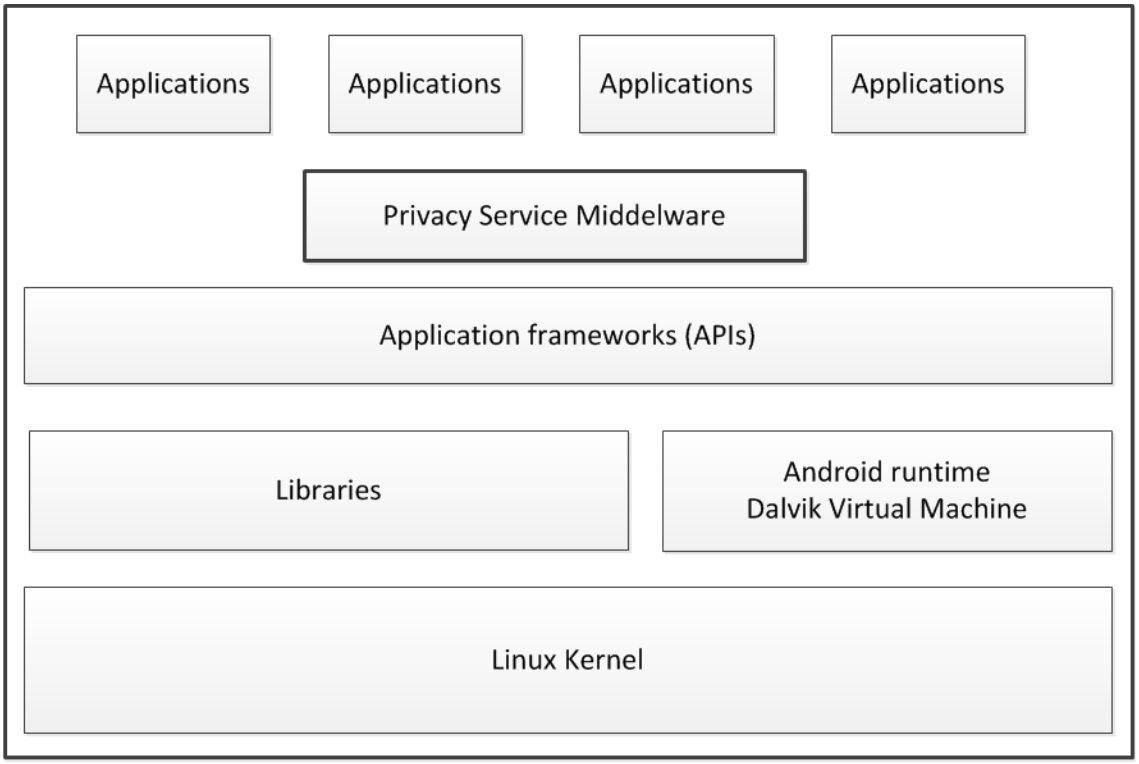


Figure 4.1: Android Software Stack including the middleware.

When Privacy Application is started, a list of all the applications installed at the phone is presented to choose from. When an application is selected from the list, a new list of all the access permissions granted to that specific application is given to the user. In addition, the user can observe a checkbox corresponding to each access permission. The listed access permissions serves as hyper-links the user may click on in order to proceed to the settings menu.

The idea is that users easily can click on the checkboxes next to those permissions irrelevant to the selected application. If one of the access permissions is checked, the response from the related API call would be faked. As a consequence, an 'all or nothingÕ choice has been made by the user. The benefit of using this approach is that users easily can navigate through the list of access permissions, and chose to either fully accept or reject them. However, the 'all or nothing' approach can introduce a ultimatum to the user; either at the expense of functionality or privacy.

Resources	**Description of the access permission**	**Data Separation options**
Location	Allows an application to access coarse (e.g., Cell-ID, WiFi) location and fine (e.g. GPS) location.	Give the user the opportunity to choose the accuracy of the location. The user can choose between; *exact, city, state, country* and *no location at all*.
Internet	Allows applications to open network sockets.	Open a limited Internet access through a filter. In addition, the user should also be able to prevent the application from connecting to the Internet when the application is not being used.
Calendar	Allows an application to read the user's calendar data.	The user can choose what events to share (e.g. Share only public events).
Contacts	Allows an application to read the user's contact data.	The user can mark which contacts to share, and which to hide.
Accounts	Allows access to the list of accounts in the Accounts Service	The ability to choose which accounts that should be visible for the application.
Storage	Allows an application to write to external storage.	Define a new folder inside the external storage, and then give the application read, write and delete options inside this folder.

Table 4.1: The proposed data separation options in our design.

The main difference between our design and the design of MockDroid[22], is the usage of data separation. For some applications, as an alternative to the 'all or nothing' approach, it might be useful to be able to give access to a subset of data, or to control the quality and accuracy of the provided data. The ability to provide 'some' data means that the user does not have to trade functionality for a good privacy. An example is location-based services. Instead of giving an application permission to access the phone's GPS, which may introduce a privacy risk, the user can share the location of e.g. the nearest city. The user would be able to use services such as 'show me the nearest hospital' without giving away the device's exact location. To start, we have proposed solutions to accomplish data separation for the six resources listed in Table 4.1. Table 4.1 briefly describe which access permissions are related to each resource, along with the proposed data separation options. The designed user interfaces based on the different data separation options described in Table 4.1, are presented in Appendix A.

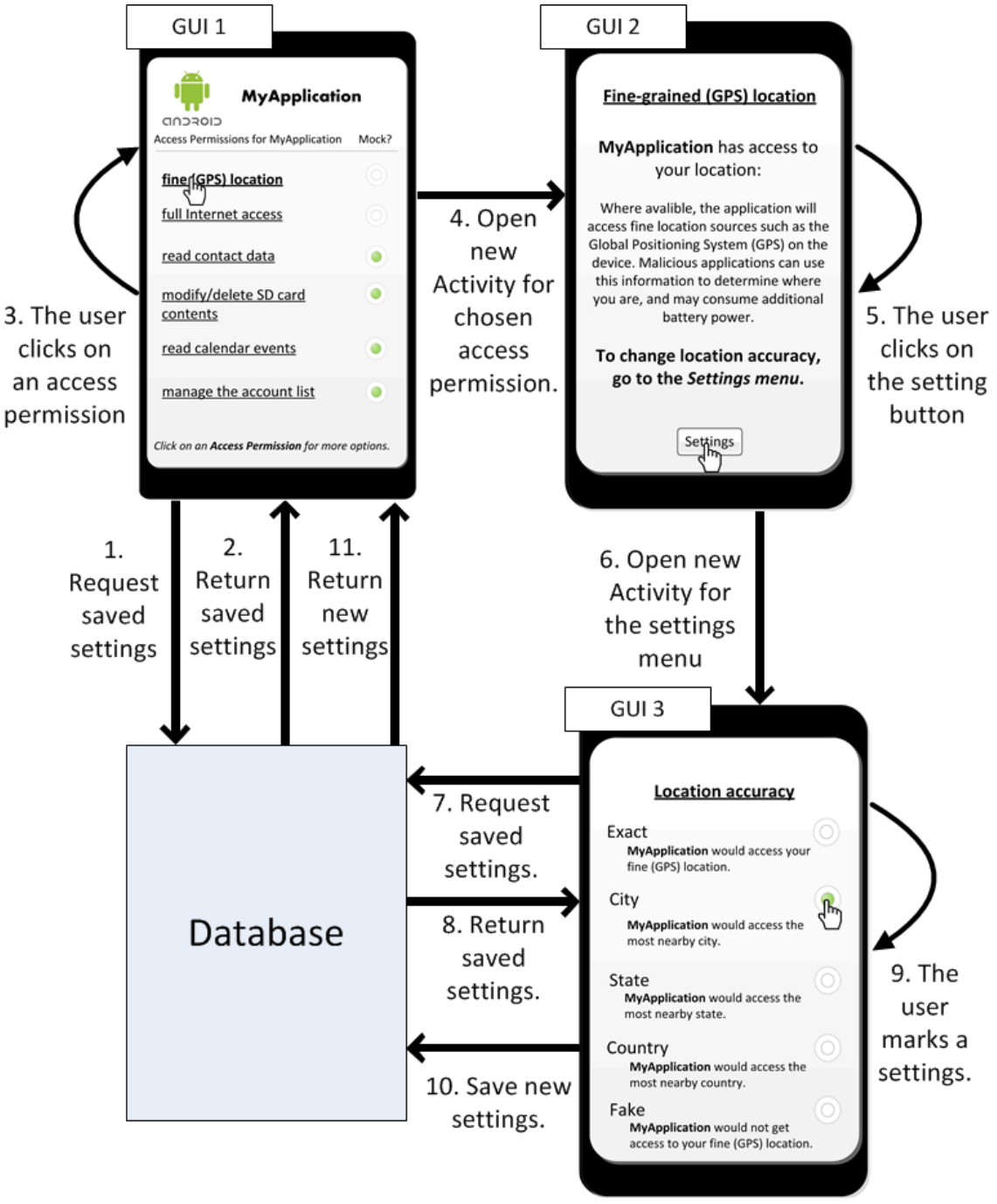


Figure 4.2: Communication between the Android activities and the database. See Appendix A for full size version of the Graphical User Interfaces (GUI).

Figure 4.2 shows the information flow between three different user interfaces and the database, after the user has clicked on the access permission *fine (GPS) location*. The hyper-link would open a new window that holds a short description of the chosen access permission (GUI 2). The reason for having this interface is to give the user a better understanding of a particular access permission, and the privacy risk related to it. The 'settings' button would lead the user to a setting menu. The settings menu (GUI 3) presents a list the possible granularities of the location. If some of the options are to be checked, they will be saved in the database.

By querying the database, the Privacy Service can keep track of what data to return upon a request from an application. Figure 4.3 illustrates the information flow between the Privacy Service and an application requesting the GPS. When the application sends a request to the middleware, the Privacy Service checks with its corresponding records in the database, and return coordinates in accordance to the specified granularity.

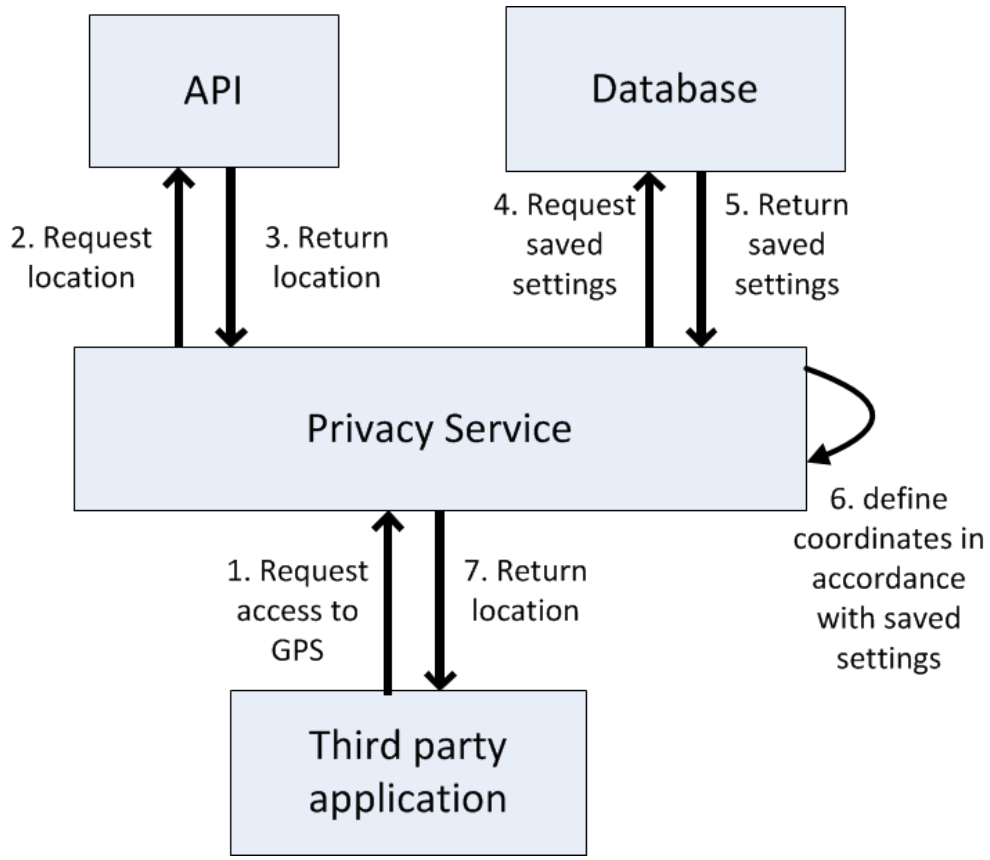


Figure 4.3: Communication between Privacy Service and a third party application.

Chapter 5

Implementation

This Chapter holds a detailed description of the development of our prototype. First, we give an overview of the lab setup and tools used for our implementation. The use of proper tools is essential to ease the work, and to get an thorough overview of the development. Second, we describe how we developed the Privacy Application, the Privacy Service and the test application. For the reader to get a good visual understanding, we have included screen shots from the most important user interfaces in our prototype.

5.1 Environment

The laboratory used during this project consisted of two identical workstations to implement parts of the design described in Chapter 4.3, and a mobile handset in order for the prototype to be tested properly. Both workstations ran unmodified versions of the Windows 7 Professional 32-bit operating system, with Intel Core2 2.12 GHz processors and 4 Gigabytes of memory. The mobile was a LG-P500 [3] touch-screen smartphone, better known as the *LG Optimus One*. The LG handset was running its original Android 2.2 Froyo software stack without modifications.

5.2 Implementation Tools

The prototype in this project was developed using a clean install of Eclipse 3.6.2. Eclipse is an excellent development platform that includes a variety of built-in functionality for both debugging purposes and opportunities to run and test applications. Eclipse is highly extensible and customizable because it is built on a plug-in architecture.

The classic Eclipse 3.6.2, better known as *Eclipse Helios*, include a Java Integrated Development Environments (IDE), a CVS client for version control and a XML editor. A Java editor alone is not sufficient to develop Java applications. In addition, the Java Development Kit (JDK) was installed.

JDK is essentially a Java Platform, consisting of the Application Program Interface (API), a Java compiler and the Java Virtual Machine (JVM) interpreter. Please note again that Android applications do not run in the JVM, even though they are created in Java, but in a custom Dalvik Virtual Machine (DVM) that is optimized for the Android platform (see Chapter 2).

After installing the core Java development components, the Software Development Kit (SDK) package was installed. The SDK is a kit used for developing applications for the Android platform. SDK include an Android emulator and essential libraries to build Android applications. To integrate SDK with Eclipse, the Android Development Tools (ADT) were installed. The ADT is a custom plug-in for the Eclipse IDE, and extends the capabilities of Eclipse to allow creation of Android applications[1].

5.3 Privacy Application

In Chapter 2.2 we briefly discussed the main components that, either separate or combined, make up an application. *Activities* are User Interface (UI) windows that contain one or more *views*, and each view contains some information that is presented to the user. A view can consist of simple text, a gallery of pictures or a list of items, to name a few. 'Privacy Application' is one of the applications that make up our prototype. This is the application where users can set their *preferences* with regards to each individual application installed on the device. By *preferences* we mean the ability users have to override the requested permissions to access resources on the device, which is specified in the *manifest* file of the application.

5.3.1 The Manifest File

In Chapter 2.3 we briefly mentioned the manifest file. Android requires this file to be present in the root directory of the applications' Android Package. The application will not run if the manifest file is not included. In addition to the required existence of the file, it has to be precisely named 'AndroidManifest.xml'. The 'AndroidManifest.xml' file is automatically generated by the Eclipse IDE, so we did not have to create it manually. Among other things, the manifest file does the following[10];

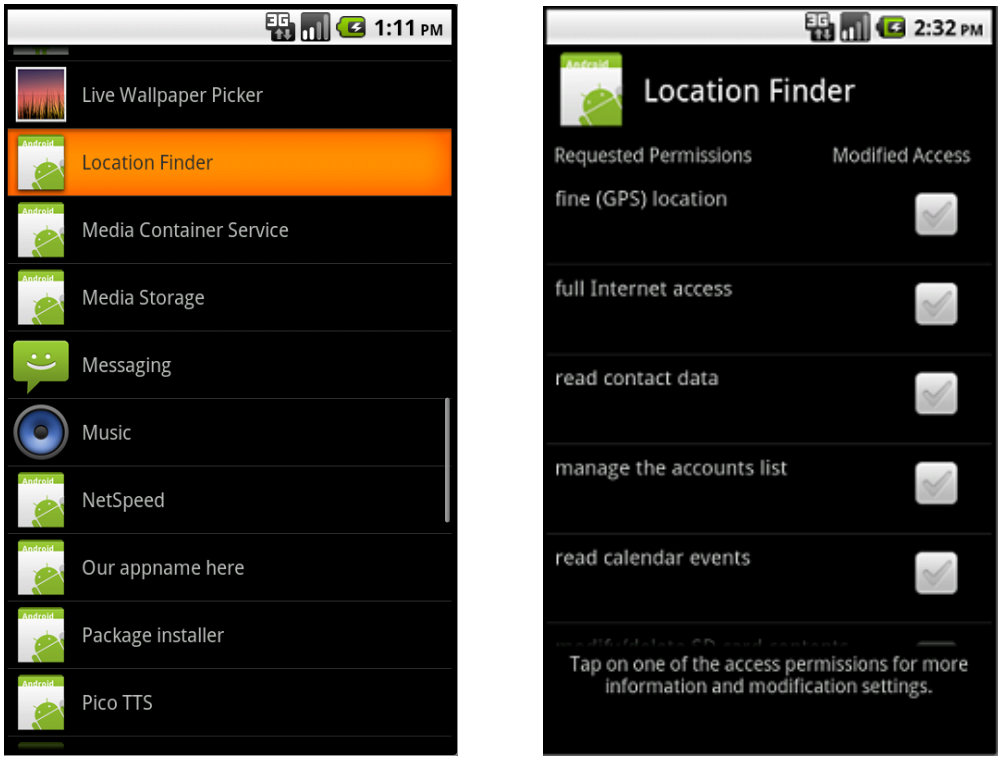
- Register components of the application - activities, services, broadcast receivers and content providers the application is composed of.
- Declares which permissions the application must have to access certain parts of the API (such as sensors) and how to interact with other applications.
- Declares permissions other applications must have granted in order to access the application's components.

[1]Complete installing instructions for JDK, SDK and ADT can be found at http://developer.android.com/sdk/installing.html

As seen in Appendix B, our manifest file include an intent-filter for the activity, that basically launches and brings the Privacy Application to the foreground of the screen whenever itÕs installed. Recall at this point that intents are the messages sent between applications within the operating system. It also specifies a reference to all of the activities and services, in addition to a 'provider' that is broadcasted system-wide on the device. The provider is a *Content provider*, and is further explained in Chapter 5.3.4.

5.3.2 Privacy Application's Activities

In Chapter 2.2 we introduced different classifications that make up an Android application. The user interface of an application is displayed on the screen of the device through an *Activity*. Each activity represents a unique screen in the application, and internally there exists a stack of activities. When moving from one screen to another, the new activity is pushed to the top of the stack and becomes visible to the user. If the user pushes the back button on the device, the current activity is popped from the stack and the previous activity is resumed.



(a) Listing of all installed applications.

(b) Listing permissions requested in Location Finder's manifest.

Figure 5.1: Two activities from the Privacy Application.

One part of our prototype, which is an application named 'Privacy Application', consists of four activities. The first activity of our 'Privacy Application', is simply a list of all installed applications on the device, as seen in Figure 5.1a. The Android API offers ways of fetching META data from packages (.apk's) installed on a device. By utilizing the API *PackageManager*, we simply inflated our activity with rows of each package name and its associated icon. An 'onListItemClick' listener is activated when one of the package names in the list is clicked. An intent is created, and the name of the chosen package name from the list is added to it. A *startActivity(intent)* function within the listener is responsible of pushing the activity shown in Figure 5.1b to the top of the activity-stack.

Figure 5.1b shows the requested permissions for the chosen application, in this case 'Location finder'. Please recall at this point that requests to access resources on the device must be stated in the manifest file of the application. Because permissions are defined as meta data, they can easily be pulled from the PackageManager. Each row consists of the textual meta data description of the requested permissions for the application. In addition, we implemented a checkbox associated with each permission. To check the current state of the checkbox, we make a call to our Content Provider to check whether the modified access-tag have been set previously or not. Exactly how the state of the activity is saved, and organized, is explained further in Chapter 5.3.3.

There are two listeners associated with each row in the activity shown in Figure 5.1b. One for the requested permission, and one for the checkbox. The listener connected to the textual description is starting a new activity that simply gives the user a more thorough description of that permission. In addition to a thorough explanation, the activity also have a 'settings' button who starts yet another activity. The 'settings' activity allows the user to define the coarseness of the location, as shown in Figure 5.2. The state of each checkbox in this activity is also fetched from our Content Provider, explained in Chapter 5.3.3.

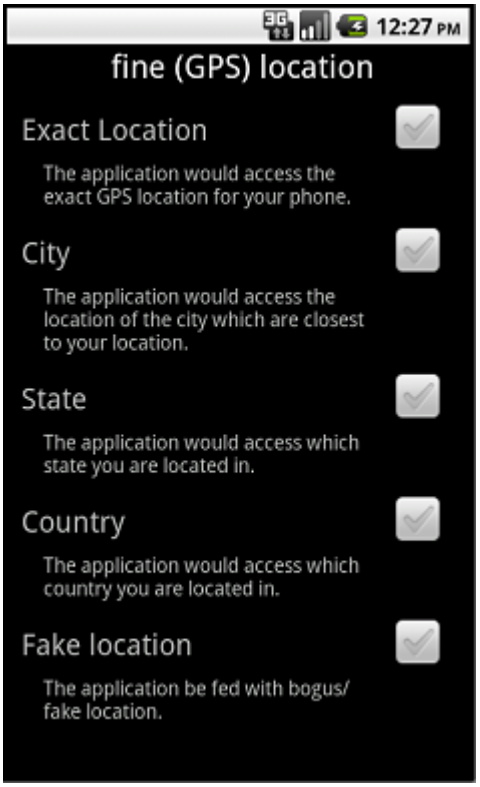


Figure 5.2: The 'settings' Interface showing coarseness of GPS location.

5.3.3 Saving Activity State

An important issue we had to overcome was how to share the preferences saved in 'Privacy Application' to the middleware service. Remember that by preferences we mean the ability users have to override the requested permissions to access different resources on the device. In addition, we would like to keep the state of activities in our 'Privacy Application'. The obvious reason for this is that choices, or preferences, made earlier should be reflected in the UI. Be it whenever the application is paused and restored, or when the application/device is restarted. Android provide several options for saving persistent application data. Table 5.1 gives an overview of the different options.

Option	**Description**
Shared Preferences	Store private primitive data in key-value pairs
Internal Storage	Store private data on the device memory
External Storage	Store public data on the shared external storage
SQLite Databases	Store structured data in a private database
Network Connection	Store data on the web with your own network server

Table 5.1: Different approaches to store data in Android.[9]

The solution of choice is obviously dependent on the specific needs. Whether the data should be private to your application only or not, or whether it should be accessible to other applications or not. The space requirements of the data are also an important factor.

We decided to use a SQLite database in order to save activity state in our prototype. In fact, we utilized a *Content Provider*, whom is described in Chapter 5.3.4. By using this approach, the data becomes accessible to both the 'Privacy Application' and to the 'Privacy Service', as described in Chapter 5.4.

5.3.4 Sharing of private data

In Chapter 2.3 we briefly described the security architecture in Android. By using a *Content Provider* we can overcome any issues related to sharing of private data. A *Content Provider* exposes read and/or write access to any private data of an application, dependent on whatever restrictions one want to impose for it.

A Content Provider is basically a lightweight database that is broadcasted to the system by the application that created it. Common data types, such as audio, video and images etc., are stored by Content Providers already shipped with Android. This is how developers are able to utilize commonly shared data in their applications. While the Content Providers are responsible for saving data, querying is done by *Content Resolver* objects. These objects are described in greater detail in Chapter 5.4.

In our prototype, none of the Content Providers already shipped with Android were suitable. That is why we decided to design and implement our own. Our Content Provider is created by the middleware, and thereby broadcasted by it. How data is actually stored is up to its designer. All content providers implement a common interface for querying and returning results - as well as for adding, changing and deleting data[8].

_ID	**AP**	**FAKE**	**FAKE_SETTING**
1	F_GPS	1	4
2	RCA	0	0
3	FIA	0	0
...			
...			

Table 5.2: Structure of our Content Provider.

The simple structure of our Content Provider database can be seen in Table 5.2.The database in the Content Provider consists of four identifiers. _ID is a standard automatically incrementing identifier of each row. AP is an acronym for Access Permission. Some of these can be seen in Figure 5.1b, where F_GPS identifies the 'fine (GPS) location', RCA corresponds to 'read contact data' and FIA to 'full Internet access', and so on. The FAKE column simply identifies whether the Modified Access checkbox have been checked or not. An Integer value of either 1 or 0 is stored in the database, where 0 indicates that the checkbox should be unchecked, and the other way around for 1.

The last column, FAKE_SETTING, indicates 'coarseness' of the returned value given from our middleware. Figure 5.2 illustrates this for an applications requesting the GPS location. The FAKE_SETTING column may hold any Integer value from 0 to 5 (for the location accuracy), where 0 indicates that none of the options are chosen. A value of 1 indicates the Exact Location, 2 indicates City, and so on.

The Content Resolver we implemented also serves as a great tool to save the state of the activity. Since the current settings for each Access Permission is saved in our Content Provider database, it is an easy task to query this database to maintain the previous state of the activities in our 'Privacy Application'.

5.4 Privacy Service

In our prototype we introduce a *Service* for the interaction between the Privacy Application and other applications. A Service is an Android component that can run long-time processes silently in the background [Mei08]. An additional advantage is that a Service does not need direct user interaction. Because we wanted our middleware to constantly listen for requests, we implemented a service instead of using an activity.

Our Service, referred to as the *Privacy Service*, is triggered by an intent message. An intent-filter in the manifest-file defines which intents the Privacy Service should listen for. The manifest-file of our Privacy Service can be found in Appendix B. The Service should not only listen for intents sent internally in the middleware, but also catch intents sent from other applications. In this prototype, Privacy Service is only triggered by two types of actions. The action 'START_SERVICE' would obviously start the Privacy Service. The action 'LOCFINDER_GPS_REQ', sent from the *Location Finder*[2], will trigger the Privacy Service to obtain, and return, a location to the requesting application. In order for the communication between applications to work properly based on intents, it is important that both the sending and receiving application use the same unique actions.

When Privacy Service is started from an intent, the middleware should request the devices' current location from the GPS. In our prototype, the coordinates are set statically to [63.419444,10.4025]. These coordinates are the exact location of our lab.

To obtain the user preferences defined in Privacy Application, Privacy Service use a Content Resolver to fetch this data. A Content Resolver is an Android class that provides access to the Content Provider implemented in Chapter 5.3.4[7]. First we query the Content Resolver to obtain the ID of the access permission *fine (GPS) location*. This ID is further used to get the FAKE_SETTING for fine (GPS) location. If FAKE_SETTING is zero or one, the coordinates will be set to the exact location. If the setting is 5, the coordinates will be set to [0.00,0.00]. If the setting is two, three or four, the Privacy Service would call a method named *getFromLocation(double lat, double lon, int maxResults)*. This method makes a call to Google Maps[1] to find the address, City, State and Country corresponding to the provided coordinates. Dependent on the accuracy defined in the Privacy Application, the Privacy Service would run a new method named *getFromLocationName(String name, int maxResults)*. This method makes yet a new call to Google Maps and returns the center-coordinates for the provided City, State or Country.

[2]Location Finder is an application implemented for testing and demonstration purposes only. More details about the Location Finder can be found in Chapter 5.5.

Finally, the Privacy Service has obtained the proper coordinates based on the coarseness specified in the Privacy Application. The results are broadcasted system-wide by an intent with the action 'LOCFINDER_GPS_LOC', and will be received by the Location Finder. We would explain the role of the Broadcast Receiver in chapter 5.5.

5.5 Test Application

To test our prototype we needed an application that could interact with our middleware. For this reason, we implemented a simple application named 'Location Finder'. Location Finder returns a list of URLs to some newspapers based on the device's GPS location.

Instead of sending a request to the API directly, the Location Finder requests the GPS coordinates from the Privacy Service. When Location Finder is started, an intent with the action 'LOCFINDER_GPS_**REQ**' is sent to the Privacy Service. In order for the Location Finder to obtain the returned coordinates from the Privacy Service, a *Broadcast Receiver* is needed. Broadcast Receiver is an *Android Abstract class* created to receive broadcast messages [6]. In addition to implementing a Broadcast Receiver in Location Finder, a *reciever* tag would have to be included in its Manifest file. When the Privacy Service broadcasts the intent including the action 'LOCFINDER_GPS_**LOC**', the Broadcast Receiver in Location Finder is able to obtain the coordinates. The Manifest file for Location Finder is included in Appendix C.

The User Interface in Location Finder is illustrated in Figure 5.3. It consists of a button, one text field including a list of newspapers, and a second text field showing the current location. When the 'Newspaper' button is pressed, a method named *updateWithNewLocation()* would execute. This method finds the address, City, State and Country, related to the coordinates from Google Maps. This information is further used to print the list of newspapers. Since the Location Finder application is made for testing and demonstration purposes only, the list of newspapers is generated from a some predefined newspapers. Based on the location, the corresponding newspaper would be presenter, whether it is a local, regional or national newspaper.

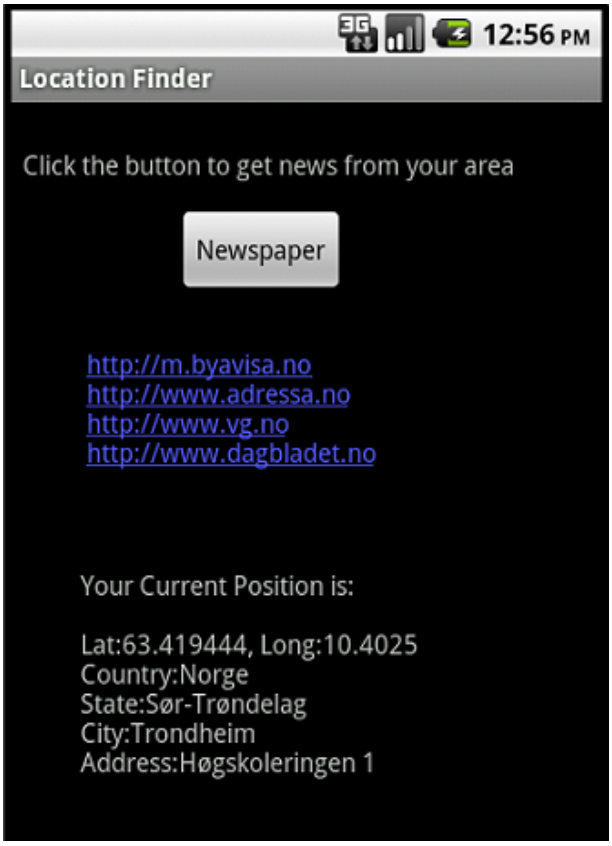


Figure 5.3: The User Interface for Location Finder when the location is exact.

Chapter 6

Evaluation

In this Chapter we evaluate the design and prototype. First, we perform a descriptive test based on three scenarios to evaluate the functionality of our prototype. Second, we perform a controlled experimental test, where we study the proposed solution in a controlled environment to evaluate the usability and usefulness.

6.1 Functionality test

6.1.1 Preparations

We have tested our prototype with the test application described in Chapter 5.5. The test is based on three different scenarios. The purpose of the functionality test was to find out whether the prototype worked as intended or not.

To be able to test the prototype on a real device, we exported the middleware (Privacy Application and Privacy Service) and Location Finder, and stored them at the devices' SD-card. To install the applications, we used an application named AppInstaller[20].

6.1.2 Scenarios

We used a scenario for each feature we wanted to test the functionality for:

1. There is not specified any user preferences for Location Finder, and the application should work as if Privacy Application does not exist.
2. The box next to *fine (GPS) location* is checked.
3. The Accuracy is set to: Exact Location, City, State, Country and Fake Location.

6.1.3 Result

We tested each scenario several times to make sure we got the expected result every time. During the test we discovered a performance delay in the prototype. We had to push the 'Newspaper' button in Location Finder a couple of time before the changes were updated in the activity.

6.1.3.1 Scenario 1

The first time Privacy Application was opened, we expected the *Fine (GPS) location* box to be unchecked, and the location accuracy to be exact. To test this, we ran Location Finder before opening Privacy Application the first time. The result was not as expected. The result from Location Finder was with the accuracy of the 'State'. When we opened Privacy Application, we discovered that even though the checkbox next to *fine (GPS) location* was unmarked, the option *State* was marked in the settings menu. We unchecked the checkbox next to *State* and ran Location Finder once more. This time Location Finder showed the exact location, and newspapers from *Trondheim, Tr¿ndelag* and *Norway* were listed (As shown in Figure 5.3 in Chapter 5.5).

This test revealed that there is a bug in the prototype, which lead to the unexpected result in Scenario 1. To fix this, we would have to debug our code and make sure this problem would not occur in later versions of the prototype.

6.1.3.2 Scenario 2

When the checkbox next to *Fine (GPS) Location* was checked, the expected accuracy of the location is a fake location with the coordinates [0.0,0.0]. When running tests based on this scenario, Location Finder stated that no news were available because the application could not obtain any coordinates. Since the result of the test based on this scenario was as expected, we concluded that the test was successful.

6.1.3.3 Scenario 3

In this scenario we ran several tests, with different preferences, to see if the functionality of the accuracy settings worked as intended (See Figure 5.2 in Chapter 5.3.2, for graphical illustration of the accuracy settings).

- Exact location:
  When the accuracy was set to *Exact Location*, we got the same results as in scenario 1.

- City:
  When the accuracy was set to *City*, we got the same newspapers listed as when the location was exact. However, when the location was exact, the address in Location Finder was *H¿yskoleringen 1*, which is correct. When the accuracy of the location was set to *City*, the address was *Kongens gate 16-18*. This is, according to Google Maps, the address of the center coordinates of Trondheim City.

- State:
  When the accuracy was set to *State*, we got a list of newspapers from *Tr¿ndelag* and *Norway*. When the location was the center of *S¿r-Tr¿ndelag*, Location Finder did not find the address and the name of the city. This is because the information in relation to the coordinates obtained from Google Maps is limited. Some coordinates have more information stored in relation to them than others. But since the accuracy was *State*, and the results from Location Finder returned both the state and the country, this was enough for the application to list the expected newspapers.

- Country:
  When the accuracy was set to *Country*, we got a list of two newspapers from *Norway* and a newspaper from *Buskerud*. For the same reason as with *State*, Location Finder did not find the address and the name of the city, when the accuracy was set to *Country*. In this case, the state was *Buskerud*, even though we were located in *S¿r-Tr¿ndelag*. The reason for this is because the center coordinates of Norway, according to Google Maps, is in the state of *Buskerud*. This did not affect the result, because the accuracy was *Country*, and Location Finder returned the expected Norwegian newspapers.

- Fake:
  When the accuracy was set to *Fake Location*, we got the same results as in scenario 2.

All these results was exactly what we expected and we concluded that the test based on Scenario 3 was successful.

6.2 Usability test

The purpose of usability testing is, according to Bevan and Macleod, 'to ensure that the delivered product reaches a minimum required level of usability, to provide feedback during the design on the extent to which the objectives are being met, and to identify potential usability defects in the product'[Bm94].

The goal of this user test was to get some advice and ideas on how to improve our design. The focus was on both usability and usefulness. The reason for testing the usability, is to discover, at an early stage, if there are some parts of the design the users do not understand. By getting a second opinion, we hope to figure out if there are parts of the solution that would be better using a different design.

6.2.1 Preparation

Before we could run the test we had to find out who the test participants should be, and what parts of the prototype we wanted to test. We developed a set of tasks to be performed during the test, and some questions to answer afterwards. When the test procedure was ready, we performed a pilot-test to see if the test procedure and tasks worked as intended. Based on this pilot, and the functional test, we hoped that potential bugs and misunderstanding were discovered. Luckily, we were aware of the performance delay, and discovered a bug that is described in Chapter 6.1. Therefore, we were able to make sure that these discovered factors would not affect the tasks performed in the test.

We used a 10 step procedure for usability testing as a guide for carrying out the test [Tog91]. See Appendix D for detailed information about the test procedure.

6.2.1.1 Test group

Due to the complexity of the test and the limited time available, we decided to use a small group of five participants with a good general knowledge of Android, privacy and computer science. This group consisted of four males and one female, all in their twenties. All the participants were students at the university. The participants tested the prototype individually, with no information of each otherÕs results. One of the main reasons for using a group of experts, besides the time schedule, was because we wanted to test the usefulness. Since this prototype is at an early stage, without much functionality implemented, we believed it would be easier for experienced users to give valuable feedback. In addition, if our group, assuming that they have an interest in privacy and security above average, does not think the design is useful, probably no one else would either. We understand that we have to perform a test with a larger group of regular users, before we can conclude that our design is generally useful and usable.

6.2.1.2 Test exercises

These are the tasks the participants had to complete during the test:

- **Task 1**
  Open Location Finder to get a basic understanding on how it works. Set the access permissions *Fine (GPS) Location*, for Location Finder, to 'fake'. Run Location Finder again to test if the location is faked.

- **Task 2.**
  Set the accuracy of the location for Location Finder to your city. Run Location Finder to test if the location is as you intended.

- **Task 3.**
  Set the accuracy of the location for Location Finder to your state. Run Location Finder to test if the location is as you intended.

- **Task 4.**
  Set the accuracy of the location for Location Finder to your country. Run Location Finder to test if the location is as you intended.

- **Task 5.**
  Once more, set the permissions for Location Finder to 'fake'. Run Location Finder to test if the location if faked. If not, try once more.

  The reason for running the last task twice (1 and 5 are similar), was to see if the participants understood that the location could be set to 'fake' in both the activity listing the access permissions (Figure 5.1b), and the activity listing the accuracy of the location (Figure 5.2). Hopefully the user would either discover this when performing task 1, or discover it throughout the test. This would also reveal which one of those two choices the participants would prefer.

- **Task 6.**

  Please answer these three questions:

  1. On a scale from 1 to 5, how difficult was it to understand how the application worked?
  2. Would you prefer this solution, or a simpler application were you only had to choose between exact and fake location?
  3. On a scale from 1 to 5, how useful do you think this application is?
  4. Would you have downloaded an application like this if it was available at the market?

6.2.2 Result

6.2.2.1 Task 1

The test group had no problem understanding how the Location Finder worked. However, two of the participants found it difficult to navigate from Location Finder to the 'settings menu' in Privacy Application. When Privacy Application is opened, the first activity is the list of applications. In relation to this list, there is no title- or information bar, telling the user 'choose an application from the list to continue'. This made some of the participants confused, because it was not intuitive how to continue to the next activity.

All the test participants had some problems understanding the activity where the access permissions were listed (Figure 5.1b). Three of the participants immediately continued to the settings menu and sat the location to fake. They did not understand that the checkbox 'modified access' next to 'fine (GPS) location' would set the location to fake in a more efficient way than navigating to the settings menu.

The other two had problems understanding the meaning of 'fake', and how to set the location to fake. As opposed to the other participants, it was not intuitive for these two that the access permissions also worked as a link to get to the settings menu. One of them asked if fake and modified access meant the same. Both ended up with checking the box next to 'fine (GPS) location', but they were not sure whether that was correct or not.

Based on the results from Task 1, we discovered that our prototype would need some GUI improvements to be more intuitive and user friendly. The list of applications the users are faced with when the Privacy Application is opened, need a title or information bar explicitly telling the user where they are and what to do. Also, another name rather than 'modified access' should be used in relation to the checkboxes next to the listed access permissions.

6.2.2.2 Task 2, 3 and 4

Two of the participants, the same two that used the 'modified access' checkbox to set the location to fake in Task 1, had to use a couple of seconds to understand how to set the location to city. Both did not discover the textual guidance at the bottom of the activity right away. When they both had oriented themselves with the screen, both understood how to navigate further to the settings menu. Both commented that they never read information located at the bottom of a screen.

Some of the participants were a little confused after they sat the location to city. They guessed that the setting was saved when it was checked, but they were not entirely sure. One of the participants commented that he missed a *setting is saved* confirmation. Another was confused regarding the use of checkboxes. Normally checkboxes indicates that more than one option can be marked at the same time, and the participant were unsure on what would happen if both city and state were checked.

Based on these tasks, we have discovered three parts that need improvements. First, the textual information located at the bottom of activities should be located elsewhere. Second, to make sure the user is confident that the settings is being saved, some kind of confirmation should be given in the settings menu. Third, in the settings menu, it would be more reasonable and intuitive to use radio buttons instead of checkboxes, to indicate that only one setting can be checked at time.

6.2.2.3 Task 5

Only one of the participants used 'modified access' in Task 1 and 'fake location' in Task 5. The others solved this task the same way as they did in Task 1. After the test, we asked whether they understood what the checkboxes 'modified access' meant or not. Only one had almost figured it out. He thought that the checkbox indicated that either the location was 'exact' or 'modified', and 'modified' was dependent on the accuracy setting in the settings menu. This is partly correct, but he did not understand that if he had marked the checkbox in task 1, it would be sat to 'fake'. The other participants did not understand the meaning of the 'modified access' checkboxes, and could not figure out the link between 'modified access' and the settings menu.

As discovered in Task 1, a more intuitive solution instead of the 'modified access' checkboxes is needed.

6.2.2.4 Task 6

The answers we were given regarding the questions in Task 6 were:

- **On a scale from 1 to 5, how difficult was it to understand how the application worked?**

  The average score was 3. This is not a sufficient result, and improvements would have to be made, to make the application easier to understand.

- **Would you prefer this solution, or an simpler application were you only had to choose between exact and fake location?**

  All the participants preferred a solution like our prototype instead of an 'all or nothing' approach. In most cases they would only vary between 'exact' and 'fake', so it is important to implement shortcuts for these settings. However, in some cases, they would like to have the ability to customize their preferences.

- **On a scale from 1 to 5, how useful do you think this application is?**

  The average score is a solid 3. They all thought the solution was useful, but they did not think the privacy risk was sufficient at the moment.

- **Would you have downloaded an application like this if it was available at the market?**

  Three of the participants would not download it at the moment, because they are not concerned about the privacy risk related to their smart-phones. However, both would have considered it, if it suddenly got more attention in the media, or they heard of a 'disaster' related to privacy on mobile phones.

  Some of the participants would be skeptical to installing a middleware like ours, simply because of the risks imposed by granting access to all of the resources. But if the middleware solution for instance were to be a part of the OS, they would most certainly use it.

Chapter 7

Discussion

In this Chapter we discuss the advantages, problems and challenges with our design and prototype. First, we summarize some of the limitations and assumptions in our solution. Finally, we discuss our solution based on the findings in Chapter 6, and compare the advantages and disadvantages from our design with related work.

7.1 Limitations and assumptions

The prototype created in this study has a series of limitations and assumptions attached to it. We wanted to determine whether the utility value of the design was sufficient to be further implemented. For this particular reason, we decided to implemented a prototype supporting only a location-based service. By using this approach, it would be easier to abandon the project at an earlier stage, and possibly focus on other solutions.

As mentioned in Chapter 5.4, we chose to set positions statically instead of using our mobile device's embedded GPS. There are several guides on how to obtain the location from the device's GPS, and it should not be faced with any major problems with implementing this functionality. Even though emulators can simulate a real GPS module, positions still have to be statically sat in the emulator. We faced numerous problems when we tried to set the locations statically in the emulator, and as a consequence we chose to statically save the locations in the application. To the scope of our project, it is irrelevant how the Privacy Service obtains the GPS locations. The importance of having positions is far more important than how they are obtained.

Another problem we had to face with the emulator was to resolve the address of the location in correspondence to the GPS coordinates. The Android API includes a feature named *Geocoder*[4], which combines maps with locations. By utilizing the Geocoder, it is possible to convert back and forth between the coordinates and the address [Mei08]. Unfortunately, we were not able to get the Geocoder to work properly. It turned out that the emulator in Android 2.2 throws an unexpected exception [2]. As a consequence, we decided to use Google's map service instead.

Google's Map service returns a set of 'center' coordinates when the accuracy is sat to either 'City', 'State' or 'Country', as the test results from Chapter 6.1 shows. It does not really matter whether these coordinates are the 'exact' center, or just some coordinates defined as 'center' by Google.

Besides the fact that the prototype only handles different granularities of the GPS location, it only handles applications that settings are set in the Privacy Application. That is, if application X is installed on the device, and never accessed through the Privacy Application, there will not exist any database entry for it in the Content Provider. As a consequence, the Privacy Service is not aware of its presence. If application X dispatches a request for the GPS, the Privacy Service will not find any record for that application.

We discovered a significant performance delay during the functionality-test, as mentioned in Chapter 6.1. We believe the reason is because the source code is not fully optimized to handle the rapid change between different accuracy settings. Consequently, users may be confused if this delay was not to be minimized. An example would be a user who change the preferences for an application, and are presented with previous settings if the application were to be started right after. In reality the user would, most likely, not change the preferences as often and rapidly as we did during our tests. The problem may therefore have been amplified during our tests compared to a real-world scenario.

Finally, itÕs important to note the fact that this prototype only has sufficient functionality implemented to support one application. By this, we mean that the database resident within the Content Provider described in Chapter 5.3.3 only is optimized to handle our test application described in Chapter 5.4. The Content Provider is easily extensible in order to handle multiple application, but this was omitted from our implementation, and is regarded as future work.

7.2 Middleware

With regards to our problem description, one of the objectives was to implement a prototype of the design proposed in Chapter 4. Our goal was to apply certain modifications to the Android operating system in order to gain control of application's access to resources. The idea of revoking access to particular resources at run-time was first published in [BRSS11]. In addition to a scientific paper, the authors of MockDroid also published the source code. The MockDroid project consists of both a series of files that is applied to the Android operating system, known as *patches*, and the Java code of the Mocker application itself.

We wanted to apply MockDroid to a mobile device provided to us. Unfortunately, the MockDroid project was developed and optimized for the HTC Nexus One handset[26]. Since MockDroid is a development system, there are no guarantees that it will work correctly even if we had a Nexus One device at hand.

Mobile device manufacturers utilize different hardware in their devices, and thereby require different drivers for the operating system to interact properly with the hardware. Because of this, we were not able to apply MockDroid to our LG-P500 handset. Unfortunately, we were not able to successfully apply the necessary drivers without any errors. Taken the risks of installing customized versions of an operating system into consideration, we decided not to jeopardize the warranty of the LG-P500 handset. In addition, we did not want to risk that our handset would suffer from a major failure and not being able to recover or re-boot. This could have had significant impact on our project since we would loose the opportunity to try different software implementations on our handset.

Applications on the other hand, except from some very rare occasions, do not cause devastating deflections on the hardware. If they crash, the OS is able to recover and continue its operation. This makes applications less harmful than changing the behavior of its surroundings. For these reasons, we decided not to continue with our approach to customize a version of the OS. It's less likely that a user would have second thoughts on installing a third-party application compared to installing patches that modifies the core components of the OS. Based on the fact that users in general only are interested in a seamless 'plug-and-play' experience, the use of a middleware application seems to be a more valid approach. From a technical point of view, forcing changes to applications by modifying the environment where they reside may seem to be the better solution. Both approaches have their pro's and con's.

If the middleware solution turns out to be the preferred solution, the question of how to enforce application to interact with a middleware arises. Our prototype solution is based on the assumption that the application that request access sends its request to the middleware, and not to the resource directly. Our paper does not attempt to solve this issue.

7.3 Simplicity, control and flexibility

In relation to MockDroid, our solution gives the users a higher degree of flexibility in cases where it may hard to choose between real and mocked data. The user-test described in Chapter 6.2, we asked the participants if they would prefer a simple solution with the ability to choose between 'mocked' or 'real' data, or a more flexible solution with the opportunity for data separation. The test revealed the importance of keeping the application as simple as possible. Users urged to have the ability to choose to share 'some' data, instead of just 'mocked' or 'real' data. In addition, it is important to allow for the user to set their preferences in a efficient manner, by extending the solution with more well-defined shortcuts. In our opinion we have accomplished to extend the flexibility, but we think there is a potential to further extending the solution, and will be regarded as future work.

7.4 Security

In general, when developing an application like the middleware solution, it is very important to maintain security. The middleware is responsible for major amounts of sensitive data. The users should be confident that the personal information accessible to the middleware is secured. If the middleware can not uphold the protection of the sensitive information, the user would be exposed to higher risks by using the middleware compared to not using it.

Since our prototype only may be regarded as a proof of concept, we have not taken specific measures to ensure the security of the middleware. It is important to note that our proposed design does not introduce obvious breach to security. With that being said, we see some potential security problems with our implementation. In Chapter 5.4, we described the concept of broadcasting intents to exchange messages between the middleware and third-party applications. Since the intents may include personal data, it is important that the broadcasted message is only received by the intended recipient. An possible way to make broadcasting intents more secure, is to require recipients to have a Receiver Permission in their manifest file[19]. If the Privacy Service broadcasts an intent meant for the Location Finder, it can make sure that only the Location Finder has the required permission to receive it.

Even though the use of receiver permissions would make broadcasting more secure, it may still not be sufficient. As far as we are concerned, there might be other, and better, solutions in order to make communication more secure. Security issues are utterly important to address in future development.

7.5 Usability

Based on the low average score in the usability-test conducted in Chapter 6.2, we suggest some adjustments to the design and implementation. The results from the tasks performed in the test, revealed some parts of the user interface that were not intuitively adequate. Task 1 and 5 from Chapter 6.2.2 showed that a more intuitive solution with regards to the 'modified access' checkboxes is needed. A possible solution proposed by one of the test participants, is to implement three columns of checkboxes. One column to indicate *no access*, one for granting *full access* and the last one to *modified access*. The 'modified access' checkbox should link the user to the settings menu directly. This is an interesting discovery that will contribute to a more intuitive user-interface.

Another discovery we made from the user-tests was related to the positioning of textual guidance on how to operate the current activity window. To place information/guidance at the bottom of a window turned out to be a bad idea, because information in general is processed from top-to-bottom. Another possible improvement would be to include an information header at the top of every activity, clearly stating which activity the user is resident in.

In addition, it could be useful to have some guidance on how to proceed in some of the activities. However, it is important to keep a fine line of how much, or little, guidance and information to present to the user. We hope that by improving the discovered problems revealed in the user-tests, the scores on usability would be higher in future tests.

7.6 Usefulness

One of the things we wanted to investigate from the user-test conducted in Chapter 6.2, was whether the participants found the solution useful or not. Based on the level of knowledge within the test group, the results surprised us. We expected the participants to be more concerned with privacy risks with relation to third-party applications. Even though the participants found the middleware solution useful, several of them did not really care on what kind of data the applications would gather. The general opinion was that a major breach in privacy had to occur before they would care to take actions. This behavior is most likely shared among other users as well. However, we still think the middleware is useful. If a 'catastrophic' incident would occur, the issues with regards to privacy would all of a sudden receive a lot of attention. In such a scenario, a precautionary solution like ours would be required.

Chapter 8

Future work

This chapter gathers some of the loose ends left out throughout the report, and points out some improvements that need to be done in future development.

8.1 Functionality

When we implement a prototype of the design proposed in Chapter 4.3, we decided to omit some of the functionality described. First of all, we decided to only implement sufficient functionality in order to demonstrate different granularities based on the location. Functionality that copes with the variety of possible resources has to be included in the future. In addition, further implementations should be able to handle the diversity of possible access permissions, as we only decided to handle the GPS.

Finally, but not least, since our prototype only support settings for one application, the ability to support multiple applications have to be implemented in the future.

8.2 Performance

Some adjustments and modifications have to be made with regard to performance. Users are not supposed to feel a significant delay when using the Privacy Application. The middleware solution should not introduce any delayed responses from applications in general. From the user-test conducted in Chapter 6.2, we found that some participants were somewhat annoyed by the delay. Optimization of the source code should be conducted in the future. Extensive performance improvements have to be conducted in order to uphold a seamless end-user experience.

8.3 Privacy Application

A possible extension to our prototype is to offer a set of predefined application preferences. The idea is based on the fact that many applications within different genres may, but are not restricted to, request the same resources. However, it is hard to define what kind of accesses a genre of applications, such as games, would request. But one can assume, for instance, that games in general should be granted access to the Internet, for whatever reason. Social networking applications may be granted access to your GPS regardless, and so on. This is one possible approach to ease the workload on users in the future.

In our prototype, the Content Provider was unaware of the presence of a specific application until it was chosen from the list within our Privacy Application. This problem should be coped with by the middleware in the future. Awareness, and a corresponding database record, should be made as soon as a new application is installed on the device.

8.4 Privacy Service

To be able to test the middleware for different locations, the Privacy Service should use real GPS coordinates. For future implementation, the Privacy Service should obtain the coordinates from the Location Manager[5], instead of the static coordinates used in this prototype. Since GeoCoder[4] is an Android class, it is probably a more reliable solution for coordinate-to-address mapping. It should be investigated whether Google Maps or Android's GeoCoder should be used in future implementations of the prototype.

As mentioned in Chapter 7.4, future implementations of Privacy Service should be more well-equipped to handle potential security threats regarding to the communication between the Privacy Service and applications in general. To do so, reliable and thoroughly tested solutions for communication should be used.

8.5 User interface

There are a lot of improvements that can be made in general to the user interfaces. Figure 8.1 shows the activity that presents a detailed textual description of the currently chosen access permission. This is a part of our solution that has not received too much attention. The textual description is Android Market's description of the permission[20]. Later solutions should preferably have more customized descriptions, and the focus is on the risks related to granting access to that particular resource.

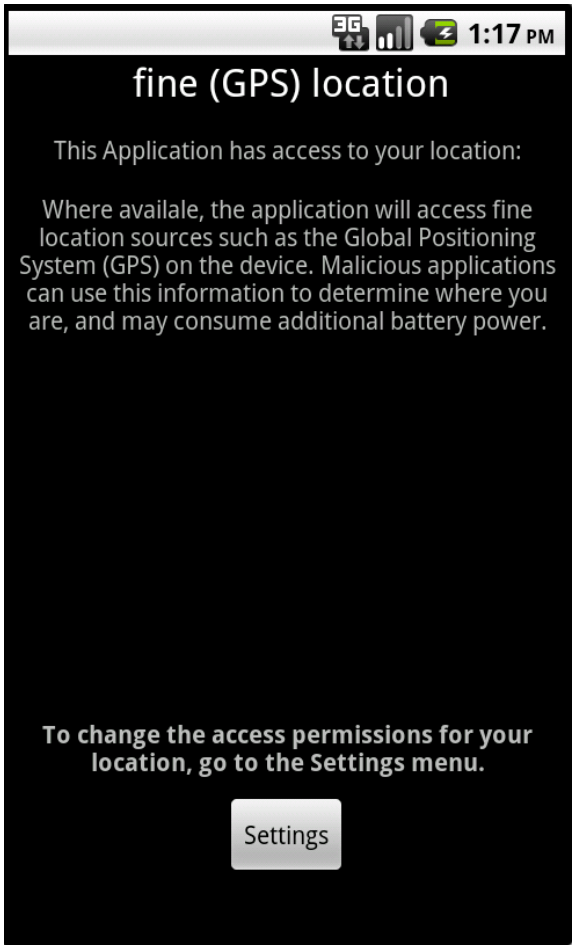


Figure 8.1: Textual description of a access permission.

One of the results we found from the user-test in Chapter 6.2, revealed that some users were confused by what activity they were faced with. Future implementations should take this into consideration, and make the interfaces easier to understand and navigate through.

Some of our test participants were also somewhat confused by whether their selected settings were saved or not. This issue can easily be solved by providing the user with a pop-up window confirming their selection.

8.6 Testing

A series of tests should be carried out on both this and future versions of the prototype. Time and cost analysis should be done to evaluate the performance of the solution. To figure out if the solution is too resource demanding with a larger amount of application requests, the middleware should be tried out with more than one test application. In addition, if more than one test application is implemented, a descriptive test based on security scenarios should be carried out to determine if the security is maintained. To evaluate how well our solution scales, scalability testing should also be carried out. Functional black-box testing should be done to test for special cases, inconsistencies etc. And obviously, to complement the usability test performed in this thesis, an extensive usability test on a larger group of participants should done.

Chapter 9

Conclusion

In order to meet the problem description, we first reviewed existing work concerning privacy on mobile devices. Second, based on the findings from the literary study, we were able to sketch a design. Our third task was to implement selected parts of the proposed design in order to demonstrate its functionality. Finally, based on our findings we were able to answer the research questions in this thesis.

Our first research question was: *How can we give users an increased consciousness on how personal information is collected and distributed from their mobile devices?*

We have proposed a tool that allows users to control what kind of personal information applications should be able to access. By controlling the amount of information to share, users should get an increased consciousness on how personal information is collected and distributed.

Our second research question was: *Can we propose a software-design that would help users protect their privacy?*

After conducting a literature study, we found a few project relevant to the scope of our project. We found one project of particular interest, namely the MockDroid project. Based on the knowledge we gained from that project, we came up with a design based on a middleware solution. The idea is a shared and centralized application that would monitor and control requests to internal resources, sent on behalf of an application. In addition, we managed to successfully implement a prototype with the functionality for location-based services.

Our main objective was to help users protect their privacy by increasing their consciousness on how personal information is collected and distributed. Even though our solution is not complete, we believe that the results obtained from our study serve as a contribution to this main objective.

References

[Bm94] Nigel Bevan and Miles macleod. Usability measurement in context. *Behaviour and Information Technology, Chapter 13, Page 132-145*, 1994.

[Bon10] Marco Bonetti. Mobile privacy: Tor on the iphone and other unusual devices. *DEFCON 18*, May, 2010.

[BRSS11] Alastair R. Beresford, Andrew Rice, Nicholas Skehin, and Ripduman Sohan. Mockdroid: trading privacy for application functionality on smartphones. *HotMobile '11, Phoenix, AZ, USA*, March, 2011.

[EGC+10] William Enck, Peter Gilbert, Byunggon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. What you see is what they get: Protecting users from unwanted use of microphones, cameras, and other sensors. *The USENIX Symposium on Operating Systems Design and Implementation (OSDI),Vancouver*, October, 2010.

[GgCCJ11] Peter Gilbert, Byung gon Chun, Landon P. Cox, and Jaeyeon Jung. Automating privacy testing of smartphone applications. *Duke University, Technical Report CS-2011-02*, February, 2011.

[HMPR04] Alan R Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information system research. *Mis Quarterly vol.28 No.1, pp.75-105*, March 2004.

[HS10] Jon Howell and Stuart Schechter. What you see is what they get: Protecting users from unwanted use of microphones, cameras, and other sensors. *W2SP 2010: Web 2.0 Security and Privacy 2010*, May, 2010.

[Mei08] Reto Meier. *Professional Android Application Development*. Wrox Press Ltd., 2008.

[Tog91] Bruce Tognazzini. *Tog on interface*. Addison-Wesley Professional, 1991.

[YMC07] Aydan Yumerefendi, Benjamin Mickle, and Landon P. Cox. Tightlip: Keeping applications from spilling the beans. *NSDI 2007, 4th USENIX Symposium on Networked System Design and Implementation, Cambridge, MA*, April, 2007.

Web References

[1] Google. *Google Maps*. *maps.google.com*, last accessed June 3, 2011. maps.google.com.

[2] Android discussion group. *Issue 8816: Android - Service not available*. *code.google.com*, last accessed June 1, 2011. http://code.google.com/p/android/issues/detail?id=8816.

[3] LG Electronics Norway. *LG P500 mobile handset*. *www.lg.com/no/*, last accessed May 31, 2011. http://www.lg.com/no/mobiltelefoner/mobiltelefoner/LG-P500.jsp.

[4] Android Developers. *GeoCoder*. *developer.android.com*, last accessed June 6, 2011. http://developer.android.com/reference/android/location/Geocoder.html.

[5] Android Developers. *LocationManager*. *developer.android.com*, last accessed June 1, 2011. http://developer.android.com/reference/android/location/LocationManager.html.

[6] Android Developers. *Application Fundamentals*. *developer.android.com*, last accessed June 1, 2011. http://developer.android.com/guide/topics/fundamentals.html.

[7] Android Developers. *ContentResolver*. *developer.android.com*, last accessed June 1, 2011. http://developer.android.com/reference/android/content/ContentResolver.html.

[8] Android Developers. *Content Providers*. *developer.android.com*, last accessed May 19, 2011. http://developer.android.com/guide/topics/providers/content-providers.html.

[9] Android Developers. *Data Storage in Android*. *developer.android.com*, last accessed May 19, 2011. http://developer.android.com/guide/topics/data/data-storage.html.

[10] Android Developers. *The AndroidManifest.xml File. developer.android.com*, last accessed May 19, 2011.
http://developer.android.com/guide/topics/manifest/manifest-intro.html.

[11] Android Developers. *Android Security and Permissions. developer.android.com*, last accessed May 19, 2011.
http://developer.android.com/guide/topics/security/security.html.

[12] Android Developers. *Android API Package Index. developer.android.com*, last accessed May 19, 2011.
http://developer.android.com/reference/packages.html.

[13] Wired Magazine. *Google's Open Source Android OS Will Free the Wireless Web. www.wired.com*, last accessed March 4, 2011.
http://www.wired.com/techbiz/media/magazine/16-07/ff_android?currentPage=all.

[14] IBM. *Understanding security on Android. www.ibm.com*, last accessed May 19, 2011.
http://www.ibm.com/developerworks/opensource/library/x-androidsecurity/.

[15] IBM. *Introduction to Android development. www.ibm.com*, last accessed May 19, 2011.
http://www.ibm.com/developerworks/opensource/library/os-android-devel/index.html.

[16] Open Handset Alliance. *Android Overview. www.openhandsetalliance.com*, last accessed April 28, 2011.
http://www.openhandsetalliance.com/android_overview.html.

[17] Android. *Android Open Source Project licence. source.android.com*, last accessed May 19, 2011.
http://source.android.com/source/licenses.html.

[18] Canalys. *Android becomes the world's leading smart phone platform. www.canalys.com*, last accessed May 19, 2011.
http://www.canalys.com/pr/2011/r2011013.html.

[19] Android Developers. *Receiver Permission. developer.android.com*, last accessed June 8, 2011.
http://developer.android.com/guide/topics/manifest/receiver-element.html.

[20] Android. *Android Market. market.android.com/*, last accessed May 18, 2011.
https://market.android.com.

[21] GSMA Mobile Privacy. *Mobile Privacy Principles*. *gsmworld.com*, last accessed April 26, 2011.
http://www.gsmworld.com/our-work/public-policy/mobile_privacy.htm.

[22] Digital Technology Group, University of Cambridge. *MockDroid*. *www.cl.cam.ac.uk* , last accessed April 26, 2011.
http://www.cl.cam.ac.uk/research/dtg/android/mock/.

[23] TV2 Nyhetene. *'Angry Birds' tapper telefonen for info og selger den.*. *www.tv2nyhetene.no*, last accessed May 10, 2011.
http://www.tv2nyhetene.no/innenriks/forbruker/angry-birds-tapper-telefonen-for-info-og-selger-den-3383899.html.

[24] Wall Street Journal. *Your Apps are watching you.*. *www.wsj.com*, last accessed April 26, 2011.
http://online.wsj.com/article/SB10001424052748704694004576020083703574602.html.

[25] GSM World. *Press release: GSMA Publishes Mobile Privacy Principles*. *www.gsmworld.com*, last accessed April 26, 2011.
http://www.gsmworld.com/newsroom/press-releases/2011/5992.htm.

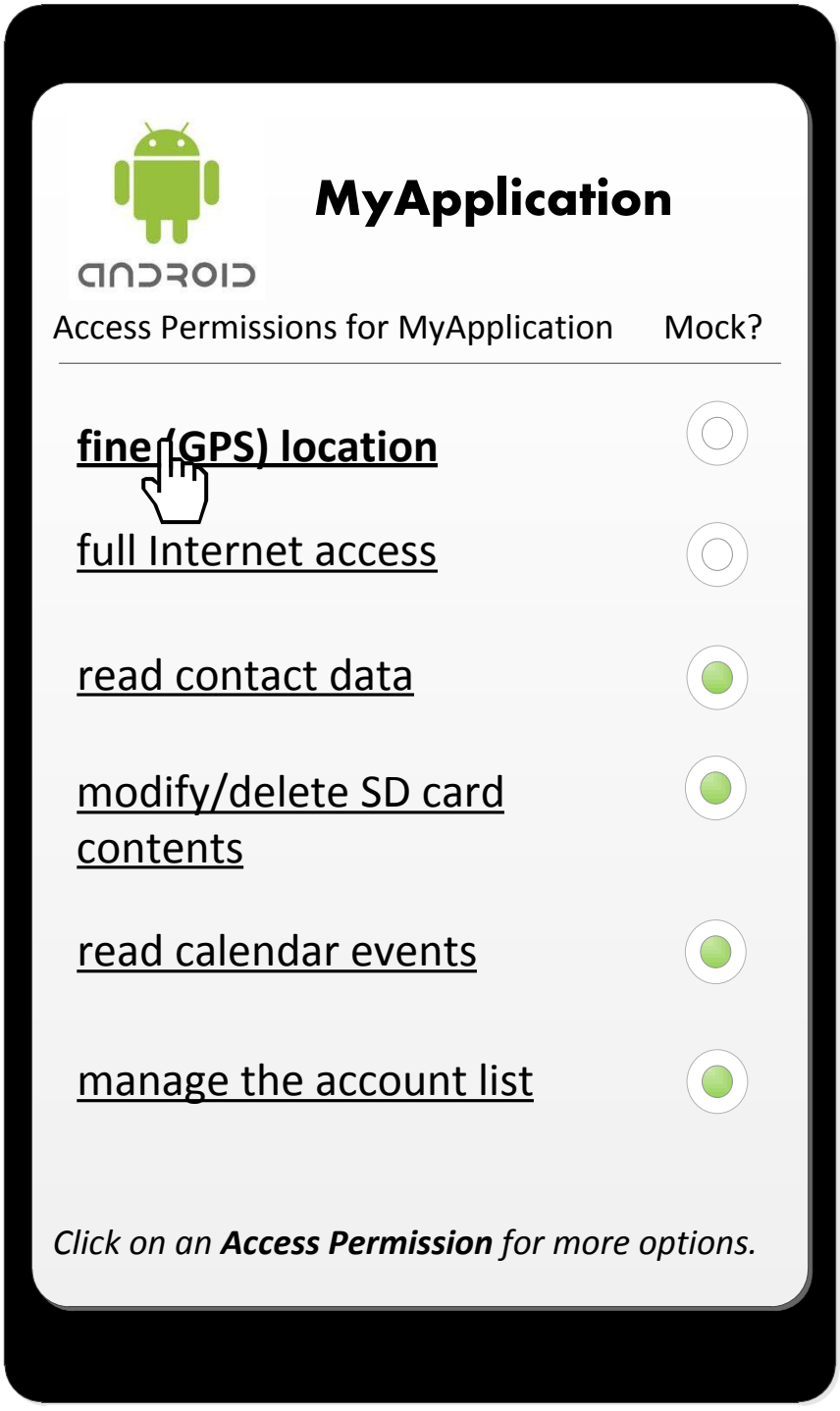
[26] Wikipedia The Free Encyclopedia. *Nexus One*. *en.wikipedia.org*, last accessed March 27, 2011.
http://en.wikipedia.org/wiki/Nexus_One.

Appendix A

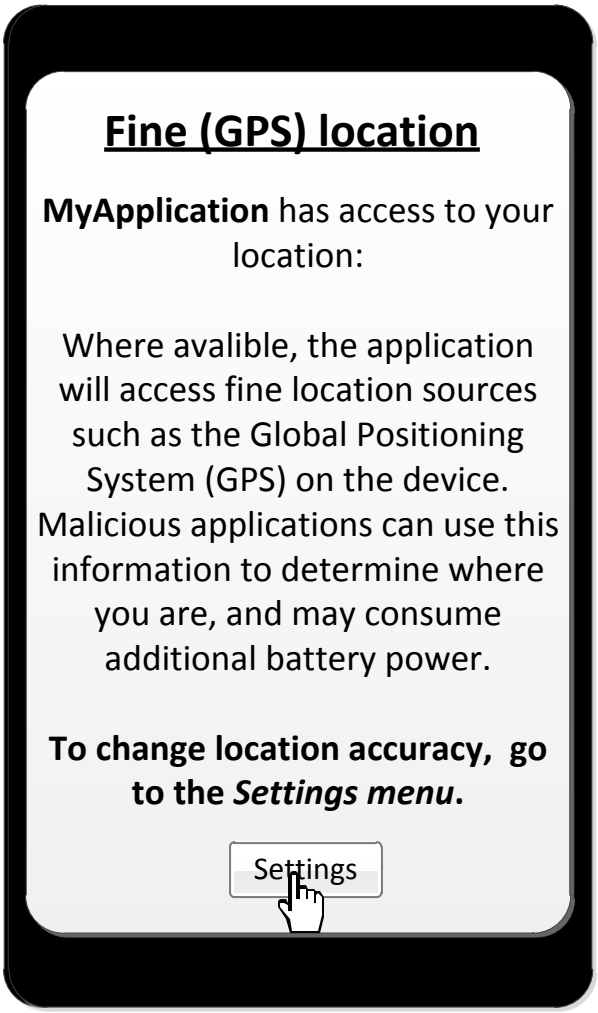
User Interface design

This Appendix contains graphical illustration of the User interface proposals for all the *Access Permissions* that is part of our design.

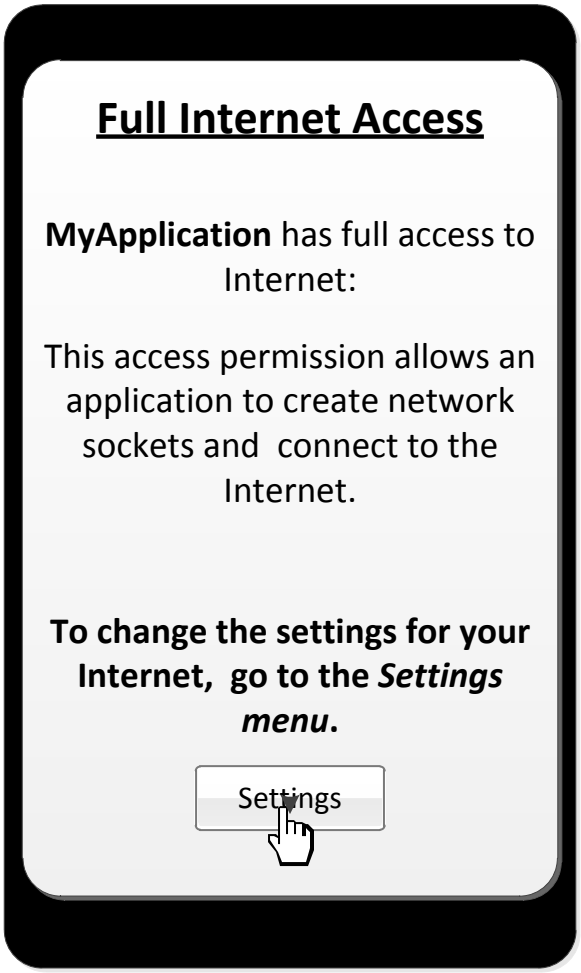
User Interfaces

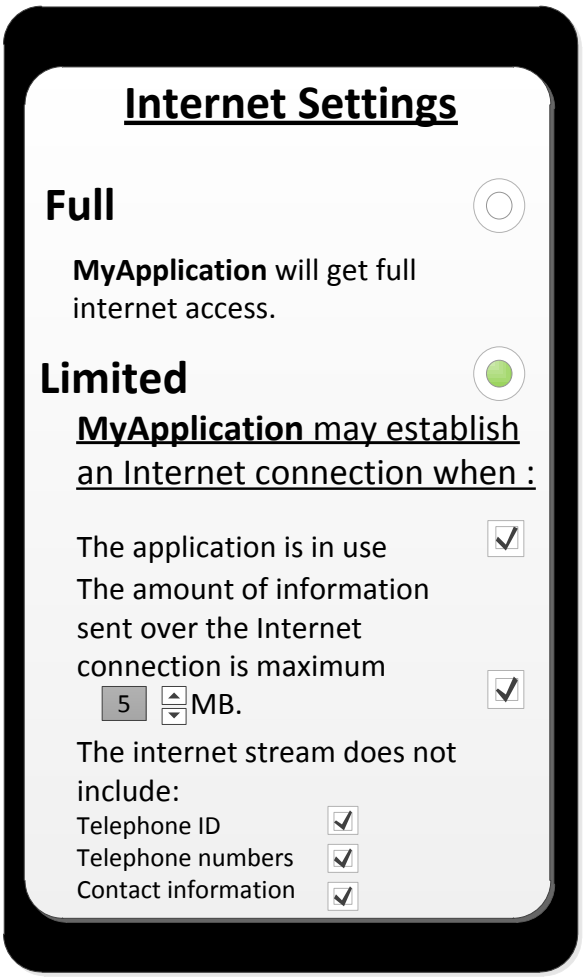


Interface of the data seperation services concerning the location.

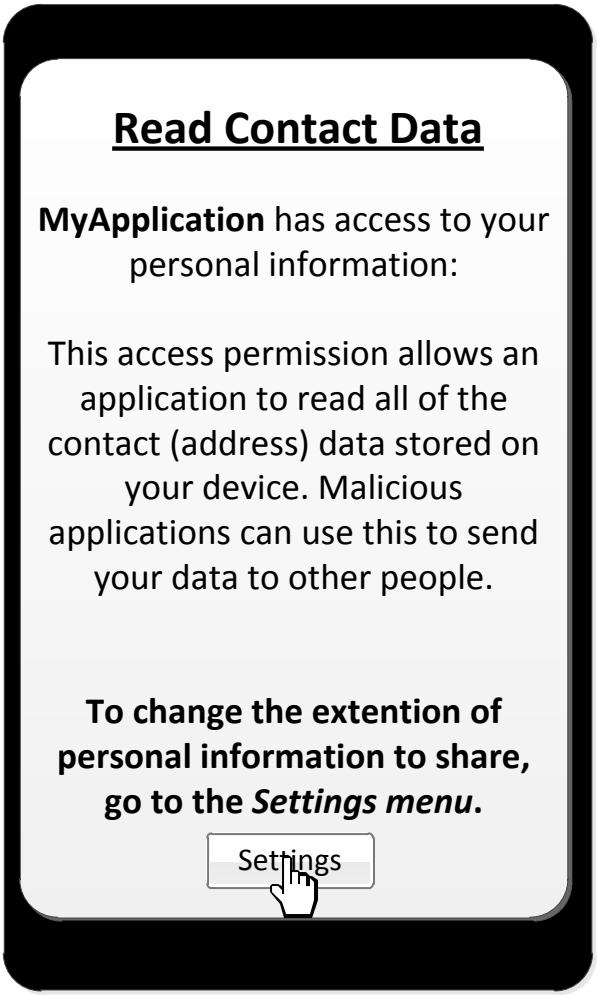


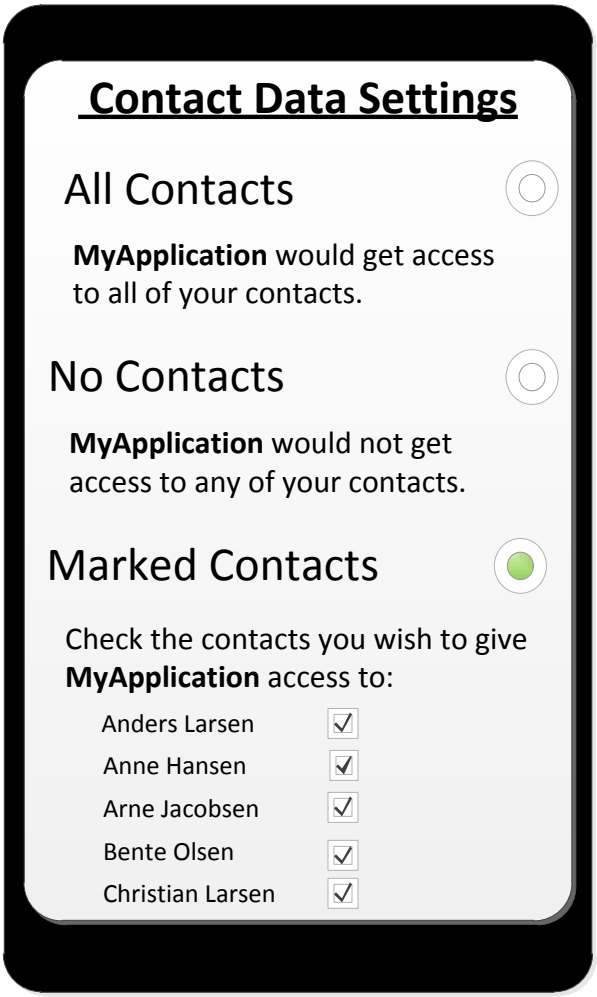
Interface of the data seperation services concerning the Internet.



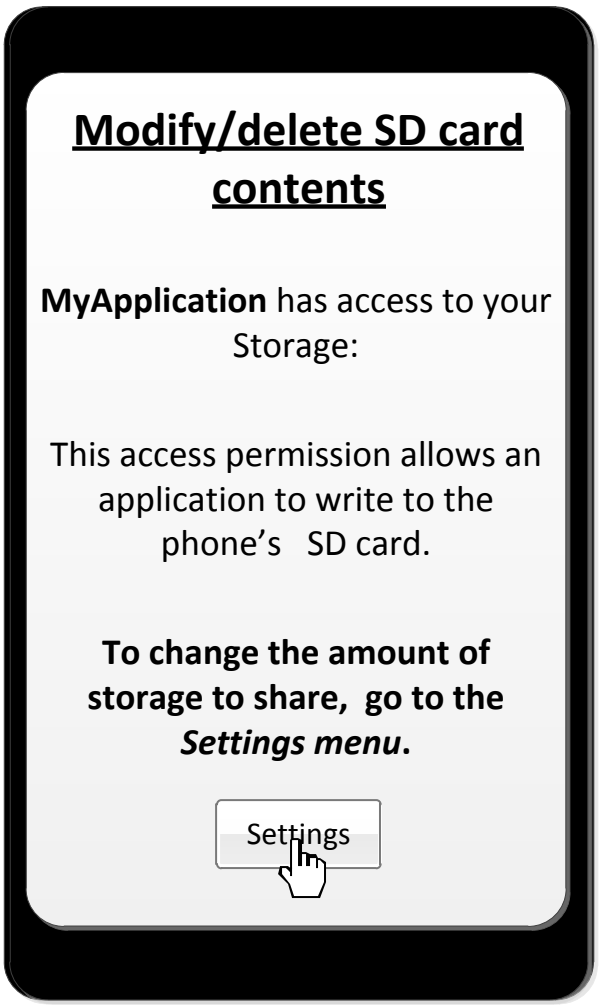


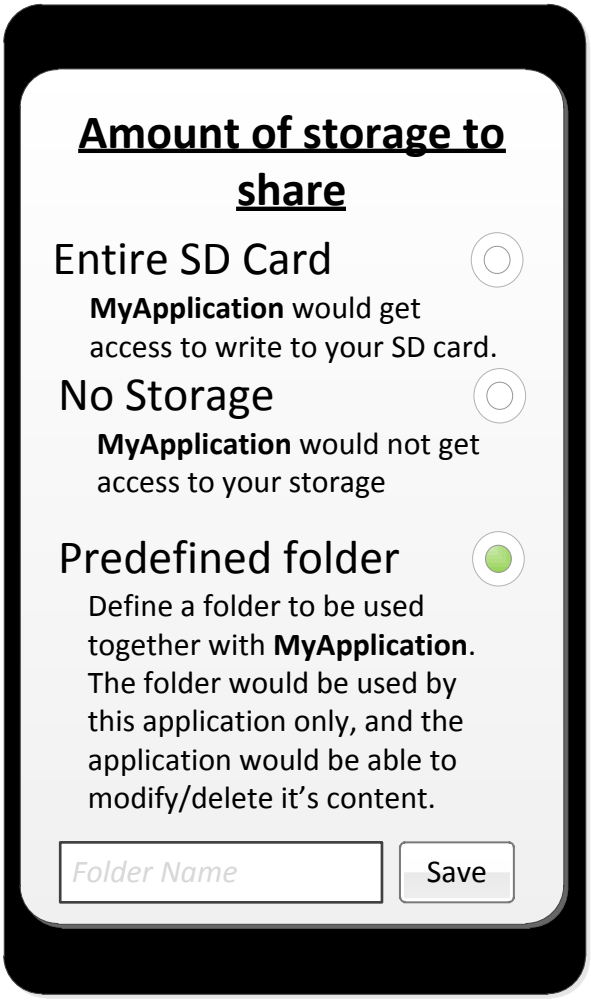
Interface of the data seperation services concerning the contacts.



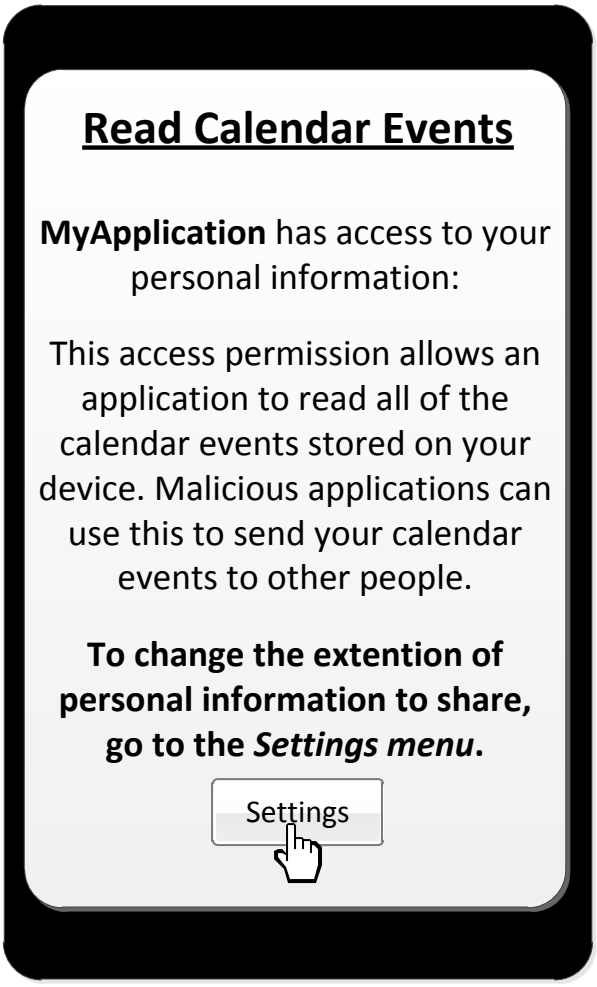


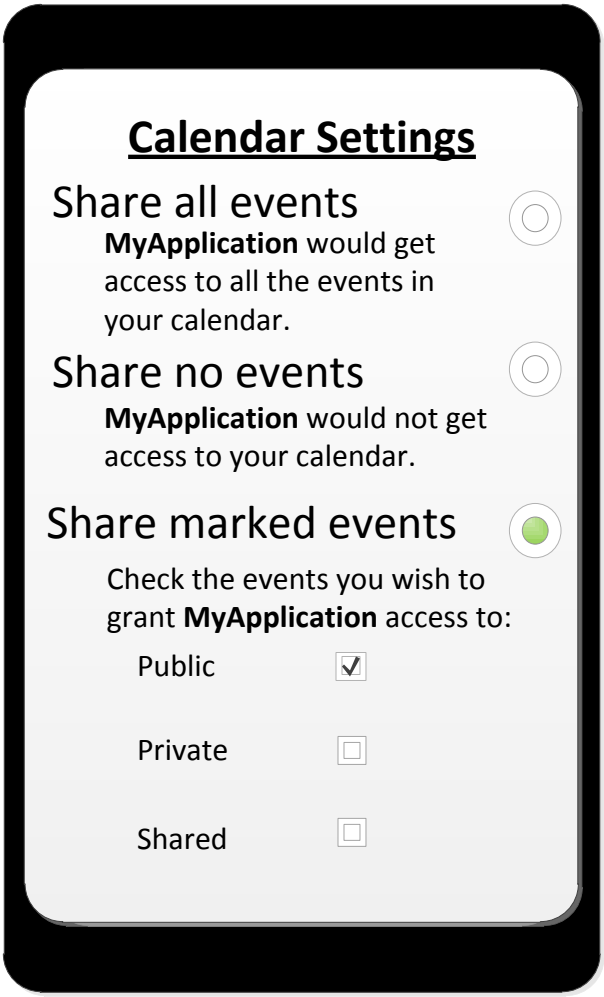
Interface of the data seperation services concerning the storage.





Interface of the data seperation services concerning the calendar.





Appendix B

The Manifest file for Privacy Service Middleware

This Appendix contains the file 'AndroidManifest.xml' for Privacy Application and Privacy Service.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="no.ntnu.stianren.applist"
          android:versionCode="1"
          android:versionName="1.0">

    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".AppList" android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SelectedApp" />
        <activity android:name=".InfoPage" />
        <activity android:name=".ShowSettings" />
        <provider android:name="no.ntnu.stianren.applist.SettingsProvider"
         android:authorities="no.ntnu.provider.settings" />
        <service android:enabled="true"
         android:name="no.ntnu.stianren.applist.PrivacyService">
            <intent-filter>
                  <action
                  android:name="com.location.finder.action.LOCFINDER_GPS_REQ">
                  </action>
                  <action
                  android:name="no.ntnu.stianren.applist.action.START_SERVICE">
                  </action>
            </intent-filter>
        </service>
    </application>

</manifest>
```

Appendix C

The Manifest file for the test application

This Appendix contains the 'AndroidManifest.xml' file for the test application LocationFinder.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.location.finder"
          android:versionCode="1"
          android:versionName="1.0">
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.MANAGE_ACCOUNTS"/>
    <uses-permission android:name="android.permission.READ_CALENDAR"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>


    <application android:icon="@drawable/icon" android:label="@string/app_name"
      android:debuggable="false">
        <activity android:name=".LocationFinder"
         android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".MyBroadcastReciever" android:enabled="true">
            <intent-filter>
                <action
                 android:name=
                "no.ntnu.stianren.applist.action.LOCFINDER_GPS_LOC">
                </action>
            </intent-filter>
        </receiver>
     </application>
</manifest>
```

Appendix D

Test procedure for the usability test

This Appendix contains a 10 step procedure for usability testing.

1. **Introduce yourselves.**

2. **Describe the purpose of the test.**
   In this test we aimed to figure out how usable our solution is. We were interested in feedback from the test participants on the usability of the product, together with a feedback on the usefulness of the design.

3. **Inform the participants that they may break off the test at any time.**

4. **Describe the equipment of the lab environment and explain the limitations of the prototype.**
   We explained to the participants that the test would be carried out on a stationary computer based in our lab, and an emulator would simulate the mobile device. We also explained the participants that the prototype only includes the functionality for location based services, and that the prototype tends to have a small performance lag. They were told that if this lag seemed to affects the test they would be informed.

5. **Learn the participants how to tell everything they think out load throughout the test.**

6. **Explain the user why you cannot help during the test.**
   The reason for this is because the goal were to get hold of the thoughts from the participants, not to explain them our thoughts and intention of the design.

7. **Describe the tasks foreseen for the participants, and introduce the prototype.**
   We started by handing out an introduction to privacy, the objective of our

approach and a short description of why there is a problem with existing solutions.

We explained that our product is an Android Application, and the purpose of the application is to help the user better control the distribution of personal information when using third party applications. In addition, the purpose of the test application Location Finder was explained. The participants were informed that they would not have to evaluate the usability of Location Finder.

We handed out the scenarios to the participants and shortly described the task foreseen.

8. **Answer potential questions, before running the test.**
9. **Finish up by giving the participants the opportunity to speak, before gathering up the loose ends.** We also asked the participants to give the design a score between 1 and 5, both for the usability and the usefulness.
10. **Use the results.**