Examensarbete
Kandidatexamen i datateknik, Akademin industri och samhälle
Högskolan Dalarna

Client-side security for anonymous data collection from handsets

Författare: Daniel Dalberg
Handledare: Mark Dougherty
Examinator: Siril Yella
Ämne/huvudområde: Datateknik
Kurskod: DT2008
Poäng: 15hp
Examinationsdatum: 2014-09-22

Högskolan Dalarna
791 88 Falun
Sweden
Tel 023-77 80 00
Abstract
The widespread use of smartphones today, and the hardware available in these smartphones should make it possible to use these devices as a foundation for digital guides at, for example, tourist attractions with a historical connection. This report examines whether it is possible to create an Android application for smartphones that functions as a digital guide. Furthermore, an attempt to delete sensitive data, in this case location data from the volatile memory on a smartphone running Android, is done. It turns out that a smartphone can be used as a foundation for a digital guide as long as the area that is to be covered by the guide is located in an area that is covered by the mobile networks and that it is possible to communicate with the GPS system. Deleting all the sensitive data from the volatile memory however, is more or less impossible.
# Contents

Table Of Figures .................................................................................................................. 1  
Table Of Tables ...................................................................................................................... 1  
Wordlist .................................................................................................................................. 1  
1 Introduction .......................................................................................................................... 2  
   1.1 Background .................................................................................................................... 2  
   1.2 Purpose ......................................................................................................................... 3  
   1.3 Project Motivation ......................................................................................................... 3  
   1.4 Questions ...................................................................................................................... 4  
   1.5 Delimitation .................................................................................................................. 4  
2 Method .................................................................................................................................. 5  
3 Technical Background ......................................................................................................... 6  
   3.1 Android Operating System .......................................................................................... 6  
   3.2 Eclipse .......................................................................................................................... 8  
   3.3 Global Positioning System ......................................................................................... 8  
   3.4 Geofence ...................................................................................................................... 9  
4 Theoretical background ........................................................................................................ 9  
   4.1 Erasing Traces of Program Activity From Memory ..................................................... 9  
   4.2 Dumping Volatile Memory from an Android Device ................................................ 10  
5 Definitions ............................................................................................................................ 11  
   5.1 Buffer zone ................................................................................................................... 11  
   5.2 Area of interest .......................................................................................................... 12  
   5.3 Point of interest .......................................................................................................... 13  
6 Application Development ..................................................................................................... 14  
   6.1 Application Testing ..................................................................................................... 14  
   6.2 User Transparency ...................................................................................................... 16  
   6.3 User Data .................................................................................................................... 17  
   6.4 Server Communication And Data Anonymization ...................................................... 18  
   6.5 Location Tracking ....................................................................................................... 20  
7 Discussion ............................................................................................................................. 22  
   7.1 System Architecture ................................................................................................. 22  
   7.2 Scalability ................................................................................................................... 23  
   7.3 Destroying Location Data ......................................................................................... 24
Table Of Figures
Figure 1 .................................................................................................................. 7
Figure 2 .................................................................................................................. 9
Figure 3 .................................................................................................................. 13
Figure 4 .................................................................................................................. 15
Figure 5 .................................................................................................................. 16
Figure 6 .................................................................................................................. 17
Figure 7 .................................................................................................................. 19
Figure 8 .................................................................................................................. 20
Figure 9 .................................................................................................................. 21

Table Of Tables
Table 1 .................................................................................................................... 32

Wordlist
BCrypt: A hashing algorithm that can be adjusted to require more CPU power to generate the hashes. This provides some protection against Moore’s law.

GCM: Google Cloud Messaging for Android is a service that makes it possible to send data from your own servers to your users’ Android-powered device, and also to receive messages from devices on the same connection.
1 Introduction

In this chapter a brief background to digital guides will be given followed by a description of the purpose for this thesis. After that a brief motivation to why there is a need of the type of application developed in this project will be given, followed by the projects delimitations.

1.1 Background

In the 1990s, new technology made it possible to start the development of products that could replace guided tours in Museums. The technology used at the time was Bluetooth. In 2001 the global positioning system (GPS) became available, leading to the development of digital guides that used this technology.

Attempts have been made to construct digital guides using the GPS and or smartphones. One attempt that uses the GPS system is described in (Bohlin & Brandt, 2013). The system consisted of a box containing a personal digital assistant (PDA), two speakers and a battery unit. A GPS-receiver was mounted om top of the box. The PDA and the GPS-receiver where connected through a wire. Information about the points of interests were presented to the user via video clips shown and stored on the PDA.

An attempt to construct a digital guide facilitating a smartphone is described in (Björk & Huhtanen, 2011). Their system consisted of an application featuring a web interface and a static map with points of interests indicated on the map. The information about the points of interest consisted of pictures and short texts.

Today’s smartphones harness hardware that makes it possible to use them for location tracking with good accuracy and to display different types of multimedia to their users. This opens up the opportunity to use these devices as a foundation for digital guiding as they already contain all the hardware needed to track the device and to let the user assimilate different types of narratives, text/picture based or audio/video based, so no other device than the smartphone itself is needed. The wide use of smartphones today also gives the use of these devices a potential to reach many users. These users are also familiar with handling the devices, reducing any problems originating from user unfamiliarity. Furthermore today’s development of wireless networking technology makes it possible for information to be stored on a server and transmitted to the users based on their location. This allows for dynamic content in applications used by smartphones.
1.2 Purpose

The purpose of this thesis is to produce an Android application prototype of a digital guide to be used on smartphones. The digital guide will use the smartphones’ GPS to track the location of the device and a remote server will be used to hold the narratives and other information. Sensitive data like location data is to be handled in such a way that it will be difficult to intercept and read or change the data while in transit, or to recreate the data once it is erased by the application. Furthermore the application should be transparent to the user informing the user about, for instance, when and why the GPS is used or when data is sent to the server.

Requirements concerning the application from the taskmasters Daniel Brandt, senior lecturer in human geography Dalarna University, and Magnus Bohlin, professor in human geography Dalarna University are listed below:

- Read device position
- Show multimedia presentations
- Sense hotspots – Events triggered based on particular GPS values.
- Log device position.
- Send device movement history to server.
- Collect and send user data to server
- Show and submit form containing questions relevant to the area visited.

1.3 Project Motivation

According to the taskmasters Daniel Brandt and Magnus Bohlin the GPS logging feature is what makes this application different from existing digital guides, since they lack this feature.

The GPS logging feature records movement patterns and makes it possible to measure the time users spend at a certain point of interest. Furthermore the taskmasters state that the advantage with this application is that it measures the revealed preferences instead of the traditional stated preferences.

The logged GPS information can be used in different ways:

- It makes it possible to see how much time a person spends at a certain location. This enables measurements of how the performed interpretation catches the user’s interest. If the visitors set aside less time than expected the information can be used to further investigate if it is necessary to improve or redo the interpretation.

- It is important to identify visitor movement patterns in order to be able to analyze how the visitors move through an area. This information can be used to manage crowding issues and be a part of the strategic management of a visitor attraction. It may involve avoiding crowding or to add additional visitor attractions to improve the spatial distribution of the visitors to achieve a better sustainability of the attractions.
- By exposing the visitors to different offers, it is possible to get a perception of how these offers are received by the users. This can be of great value to commercial interests, but can also be used to handle management issues like directing visitors to handle crowding.

The application which is intended for visitor attractions and their interpretation can also be given a more general implementation, for instance in a commercial context or at events.

1.4 Questions
Aside from the requirements set by the taskmasters there are other questions that are interesting from a forensic point of view. In this thesis an attempt will be made to answer some of those questions:

Can the collected movement history be deleted from the device in a forensically sound way once it has been sent to the server?

How do the actions taken to handle the security issues stated above impact the applications performance and usability?

1.5 Delimitation
The development of the application was done towards the Android operating system using the Eclipse development environment.

The Android platform has been chosen since it is an open source platform with well documented APIs. Development towards this platform is also taught as a part of the program Digital Brott och eSäkerhet, hence making it convenient to use.

The choice to use Android as platform limits the number of smartphones that are able to run the application. However statistics show that 78% of smartphones sold in 2013 were Android devices (Ahonen, 2014). Further limitations come from the fact that the Android release version Honeycomb, API 11, has been set as the lower boundary for Android versions that can run the application. The API 11 has been chosen due to that APIs whit lover version numbers lacks support for some features used by the application. For instance as seen in (Seralo, 2014) the ability to stream Http lives streams were not included in the Android platform before the release of API 11.

Eclipse was chosen as development environment for the simple reason that it has been used throughout the author’s education and he has hence become accustomed to it.

Traffic analysis has not been considered as a relevant threat, however the communication between the application and the server will be encrypted. The information in transit between the nodes will therefore be unreadable for anyone that might be eavesdropping.

Only the basic features of the application were developed due to the tight timeframe.
2 Method

The work process started with an informal interview of taskmasters Daniel Brandt, senior lecturer in human geography at Dalarna University, and Magnus Bohlin, professor in human geography at Dalarna University. The goal of the interview was to obtain an understanding of why the application needed to be developed, what they wanted the application to do and what demands they had on the application. The informal approach of the interview was chosen because that it would have been difficult to gather all the information required if for instance a questionnaire had been used. It would also been hard to discuss different aspects of the application if another approach had been used.

When the demands of the application had been determined a discussion about the main system architecture was held and the main features were confirmed by all parties.

In early stages of the system development critical details of the system architecture such as client server communication, GPS and geofence utilization was tested with help of prototyping.

The development model used was built on an iterative approach. In this case this means that key features of the system were developed and tested gradually. An example of this is the communication between the application and the server that was considered a key feature and had to be developed and tested in an early stage of the project.

During the development of the application different developer forums have been consulted for information on how to implement different functionalities in the application. The forums that have been used most frequently are: stackoverflow.com and developer.android.com but other relevant forums have also been used. A miner literature review has also been done of literature that was deemed relevant to the solution.

When the application development was done the application was compared with an application proposed and developed in a previous thesis (Björk & Huhtanen, 2011).

- To test that the application`s usability, the system performance and that the forensic countermeasures taken are working satisfactorily the best thing to do would be to run a series of live tests where a number of people would use the application during one or more guided tours and then examine the results.
- The application`s usability could be examined by questioning the users about how they experienced the application.
- To examine the system performance a number of programs could be installed on the server to measure the response times and other parameters that are relevant to determinate the server load and performance.
- It would be suitable to examine the forensic countermeasures by dumping the memory from the devices used during the tours and to examine the memory dumps for traces of the location history. The local and external storage of the devices would also need to be examined for such traces.
However it has not been possible to perform these kinds of test due to a number of reasons. Due to the tight timeframe no test group could be prepared. Even if a test group had been prepared the timeframe for the project would not had been big enough to perform memory dumps from used devices and to examine these dumps for traces of sensitive information.

Furthermore no direct access to the computer working as server for this project was given to us by Dalarna University, effectively preventing any kind of performance measuring of the server and the system in its whole.

The testing of the system that’s been done has mainly been done through the virtual phone interface supplied by the development environment. These tests were mainly conducted to confirm the code implemented in the system worked as expected, and therefore only one client were used. Some live tests have been conducted, but only at a very small scale, only using one client, to confirm that the system was working as expected in a live environment. These tests are obviously not comprehensive enough to generate the kind of load on the system that is needed to be able make any kind of measurements of the systems scalability or performance.

The method used to make it more difficult to retrieve sensitive data from the application has not been tested at all due to the tight timeframe.

This means that all claims about the systems performance, scalability or the ability to remove sensitive data from the device memory are purely theoretical.

3 Technical Background
In this chapter of the report, systems and software that are of importance to the application and the project will be reviewed and explained to the reader.

3.1 Android Operating System
In this section a brief introduction to the Android operating system will be given. This is done in order to familiarize the reader with the environment that the application runs in.

The Android platform was created by Android Inc. Google bought the platform in 2007 and released it as the Android Open Source Project. The Android Platform is developed and distributed by the Open Handset Alliance (OHA). OHA consists of many different companies. The version history of the Android operating system is shown by (Seralo, 2014). Error!

Reference source not found.

The Android system architecture can be divided in to five layers:

- Kernel
- Native Libraries
- Android Runtime
- Application Framework
- Applications
The Android operating system uses a Linux kernel. At the time of writing the current Android version “KitKat” uses the Linux 3.4 kernel or newer, depending on the device that Android is compiled to run on. Changes have been made in the Kernel to, among other things, meet the demands for power and memory management in handheld devices.

The Android runtime consist of the Java core libraries and the Dalvik virtual machine. The Dalvik virtual machine interprets the Java byte code to Dalvik byte code which then is compiled to native code.

The Frameworks in the Application Framework layer provides abstractions of the underlaying native libraries for the Dalvik virtual machine and defines its abilities towards the applications run in the virtual machine.

All applications that run on Android -each executes in their own instance of the Dalvik virtual machine.

The Android operating system does not use swap space when managing its memory. Instead Android uses paging and memory-mapping. This means that the only way to decrease the amount of memory an application uses, is to terminate all references to objects that the application no longer uses or needs. The Android garbage collector will then be able to reclaim the memory once it is run by the Dalvik virtual machine.
When the user switches between apps, Android keeps the processes that are not running a foreground application, an application visible to the user, in a least-recently used (LRU) cache. This allows for quicker load times when the user switches between applications. When the system runs low on memory it will start killing processes in the LRU cache, starting with the application that was least recently used.

3.2 Eclipse

In this section a brief overview of the development environment used to develop application is given. This is done to familiarize the reader with this tool.

Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. The eclipse IDE is written mostly in Java and can be used to develop applications. By means of various plugins, the Eclipse IDE can be used to develop applications in many other programming languages. One of these plugins is the Android Development Tools (ADT). It can be used to write programs for the Android platform. The ADT contains the Android software development kit (SDK) that can be used to debug applications and to export finished .apk files to distribute. An .apk file is generated when an Android project is compiled; it is a container for the application binary. It contains all of the information necessary to run the application on a Android device.

3.3 Global Positioning System

In this section a brief overview of the Global Positioning System is given. This system is a key feature of the application and the author therefore feels that it is important to explain its functionality to the reader.

The Global Positioning System (GPS) is owned by the USA and maintained by the US Air-force. The GPS system consists of at least 24 satellites that circles the earth in medium orbit. Each satellite orbits the earth twice a day. The GPS satellites are organized into six equally-spaced orbital planes surrounding the earth. This ensures that users can locate at least four satellites from almost any point on the planet. The GPS system also includes a control segment. The control segment consist of: one master control station, an alternate master control station, 12 command and control antennas and 16 monitoring sites. The purpose of the control segment is, to among other things, to maintain the satellites in their proper orbits and to adjust the clocks of the satellites.

Each of the satellites broadcasts information containing the satellites location, status and the precise time, using radio signals. When a GPS device receives the signal it notes the exact time the signal was received. The device then calculates the distance to the satellite using, the travel time of the signal (the difference in send and receive time) and the speed of the signal. In order to calculate the position of the device in two dimensions, the GPS device needs to know the distance to at least three satellites. The device can then calculate its position using geometry. To get a position in three dimensions four satellites are needed.
In this section a brief overview of the Geofence API is given. This system is a key feature of the application and the author therefore feels that it is important to explain its functionality to the reader.

The geofence API is a part of the Google play services API. To be able to use its features the phone needs to have the Google play application installed. When a geofence is defined a center position, a radius of the geofence and a period of validity is supplied as in-parameters. The center position consists of latitude and longitude, the radius is a distance in meters and the period of validity is defined in milliseconds. When the geofence has been defined it has to be registered with the Android location service to become activated. The geofence registers when the device enters and/or exits a predefined area. When a transition is detected the geofence API notifies the intent-service or broadcast-receiver that was assigned when the geofence was registered. The core information that are sent are: what geofence has been triggered and what type of transition was it.

4 Theoretical background

4.1 Erasing Traces of Program Activity From Memory
Data that needs to be available to a application are stored in the volatile memory. These data are stored in plain text since it has to be accessible to the application. Locard's exchange principle can be applied here, in the sense that it is very hard to run a program on a computer without leaving any traces in the memory.
Even if the data were to be encrypted before it was stored in the memory, the process handling the encryption would have to store the plaintext information in the memory during the encryption process and therefore the data would still be stored in the memory in plaintext. The encrypted data would then have to be decrypted each time it was to be used. This would not only be impractical, but the security gained would be minimal, all while putting a heavy workload on the system, achieving a huge reduction in system performance while gaining almost nothing in security.

Even if the memory used by a variable were to be reclaimed by the system when the variable is no longer used, the information stored in that particular part of the memory would not be gone. The data will remain in the memory, although unreferenced, until it is overwritten with new data or the power to the memory is turned off. In other words; it is possible to recreate data that is no longer referenced, if the memory where to be dumped and thereafter carved. For this to work the memory dump would have to be performed and completed before the part of the memory that holds the data in question is overwritten, and all of the memory would have to be dumped.

There are basically three methods that can be used to minimize the traces of information left behind in the memory by the program.

Either the variables can be cleared when the data that the variable holds no longer is needed by the program, or the variables can be cleared before the program is terminated. Clearing the variables would here mean that the variable content in some way is obfuscated before the memory is handed back to the system.

A third option would be to use homomorphic encryption (Gentry, 2009), a type of encryption that makes it possible to perform calculations on encrypted data and still get correct results when the answer is decrypted. However at the time of writing, fully homomorphic cryptosystems are not very efficient. For example: if it was possible to perform a search on Google with encrypted key words, it would increase the computation time of the search with a trillion (Schneier, 2009). This type of encryption is in other words not practical to use at its current state.

4.2 Dumping Volatile Memory from an Android Device

In this section a quick review of two different methods used to dump volatile memory from android devices will be made. This is to illustrate to the reader how important it is to protect and manage sensitive data with care in smartphone applications.

There are several ways to dump the volatile memory from an Android device. In this project two methods have been considered. The first is the use of a Kernel module called Linux Memory Extractor (LIME) (Macht, 2013). The other is the use of the method described in (Stüttgen & Cohen, 2014)

The use of the LIME module can only come in question in certain cases. The reason for this is that in order to dump the memory using LIME, the kernel version used by the device that the
memory should be dumped from, needs to be identified. Then a compatible kernel needs to be found. The compatible kernel then needs to be cross-compiled with the LIME module. When this is done, the original kernel on the device has to be replaced with the new cross-compiled kernel. Far from all phone manufacturers have released information about what kernel they are using for their different Android versions, so it won't always be possible to find a compatible kernel. This limits the range of situations when use of the LIME module can come in question.

The other method also has its limitations. It can only be used on devices running kernel versions in the range from 2.6.38 to 3.10. The method works in the following manner: In order to dump the memory of the device, the running kernel used by the device is scanned in order to find out what modules it uses. When this information is known, the modules are investigated in order to find modules that are using APIs needed to dump the memory of the device. If a suitable module is found it is infected with a parasite module capable of dumping the memory of the device. The parasite module modifies its host to pass control to the parasite when the module is called. The parasitized module then can be called from the running kernel to dump the memory of the device.

Both of these methods need the device to be rooted before they are applicable.

As shown in (Stirparo, et al., 2013) and (Ntantogian, et al., 2014) it is possible to find sensitive data in memory dumps, even from applications designed for banking. This shows that it is hard to remove traces of application activity from the volatile memory, and that developers of smartphone applications need to consider carefully the way in which they treat sensitive data in their applications.

5 Definitions

5.1 Buffer zone
The buffer zone is a geographical area surrounding the area of interest. It’s used as a flag in the communication between the application and the server. The device can be outside or inside the buffer zone. Entering or exiting the buffer zone will trigger events in the application. During the initial stage of development the borders for all the different zones used by the application was defined with the help of “boxes”, consisting of four coordinates. From these four coordinates the maximum value of longitude and latitude was defined as one point and the minimum longitude and latitude as another point. These two points were then used as reference points. The server would compare incoming coordinates to the reference points, stored in a database, and if the incoming coordinates where between the maximum and minimum values the coordinate was assumed to be inside the “box”.

The “box” method requires the application to consult the server each time the device receives a location update, to monitor the device location in relation to the area and points of interest.
The geofence API, however, does not need to communicate with the server at all; its performance has also been optimized to use less battery power and the API also provides other, additional, features. In other words, using the geofence API when possible would be beneficial.

Instead of using the “box model” to define the area, and points, of interest, the application was modified to use Google’s geofence API. In this way, the performance of the device was increased and the network overlay reduced.

The “box model” is now only used to define the outer buffer zone.

5.2 Area of interest
An area of interest is a geographical area that contains one or more points of interest. Events are triggered in the application when an area of interest is entered or exited.
5.3 Point of interest
A point of interest is a location that is of interest for the application user. It can either be visible or invisible in the Graphical user interface (GUI). Visible points of interest represent actual points of interest, such as a place where a historically interesting event took place or a historically interesting building. Invisible points of interests are locations where the user might need information or guidance in order to make a choice, such as a road crossing. Entering a geofence representing a point of interest will trigger the event that the particular point of interest represents, the following events are supported:

- Stream playback of audio file
- Stream YouTube video

![Image](image-url)

**Figure 3**
The picture shows the implementation of the buffer zone and geofences representing an area of interest and its points of interest in a fictitious environment. The red square represents the buffer zone. The black circle represents an area of interest. The blue circles represent points of interest.
6 Application Development
The application mainly uses the method proposed in (Prashant, et al., 2014) to communicate with the server, with two exceptions: the application uses https instead of the proposed Google’s GCM to communicate with the server, and the application does not use reverse geocoding technology.

6.1 Application Testing
During the application development, a “homemade” GPS log consisting of coordinates collected from a walk in the surroundings of Rödavägen 3, was created and used. The coordinates were collected from a Sony Ericsson Xperia Arc LT15i using the application “GPS Logger”. With the help of Google Earth a test buffer zone, area of interest and three test points of interest were made, using the “homemade” GPS log as a basis. The area and three points of interests were defined, so that the coordinates in the GPS log would pass through these areas hence simulating a virtual phone passing through them. The buffer zone was constructed so that it would enclose the area of interest mentioned above.

The data used to test the audio streaming feature consisted of a random mp3 file. To test the YouTube streaming feature a random video Id was used.

The live testing has been conducted on two locations. In Borlänge using the same test zone that was used in the virtual tests. The final testing was performed using coordinates and data from an actual tourist attraction in Avesta. The data for this test was supplied by the task masters.

The live test has been conducted using a Sony Ericsson Xperia Arc LT15i running Android 4.0.4, “Ice Cream Sandwich” API 15 and a Samsung Galaxy S3 i9300, running “Cyanogenmod 10.2.0” Android 4.3 API 18.

Test cases used during the testing is shown in Figure 4 below.
Figure 4
The picture shows different user cases used to test the application’s performance.
6.2 User Transparency
To prevent the application from being perceived as invasive and in order to be transparent to the user, the application informs the user about when sensitive data is about to be handled (Center_for_Democracy_&_Technolog & Future_of_Privacy_Forum, 2012). The first time the application is started a privacy policy is shown to the user. The privacy policy describes what the application does, what kind of data that the application collects, and what is done with that data. The privacy policy is also accessible through the application menu system. Before the application uses device GPS or sends data to the server, the user will be promoted with information about what is about to be done and request for consent.

![Figure 5](image)

**Figure 5**
To the left the picture shows the text that informs the user the first time the application is started. To the right the prompt that informs the user about the upload of the location history is shown.
6.3 User Data
The task masters have requested that some information should be collected from the user. This is done the first time the application is started. It is possible to access and change this data through the applications menu system. The data collected is:

- Gender
- Year of birth
- Zip code

According to (Sveriges_Riksdag, 2014) no permission is needed from Datainspektionen to store this information, as long as the users gives their consent. The collected data is sent to the server where it is stored in a database. The information is to be used to get an overview of user groups demographics.

![User Data GUI](image)

*Figure 6*
The picture shows the GUI that is used when the user enters the user data.
6.4 Server Communication And Data Anonymization
In order to preserve the integrity of the data sent to the server the application communicates with the server using https requests. The data that is uploaded to the server is to be used in statistical analyses to track changes in visitor’s behavior when e.g. the content of a point of interest event is changed. In order to identify data uploaded to the server by the same user, a unique identifier for each device is used. The identifier is used whenever the server stores data transmitted from the device.

The unique identifier consist of the device Wi-Fi MAC address, or if the device doesn’t support Wi-Fi the device IMEI-number. This piece of data is hashed 8000 times using the SHA-512 algorithm. The final hash is then used as the unique identifier.
This process is used so that data used as identifier is not saved in plain text. If it was it would be very easy to trace the data back to the device. Which would mean that the source of the data could easily be traced.

To be able to find out what hash belongs to which user, a list of all possible MAC-addresses would be needed. One would also need to know how many times the MAC-addresses have been hashed. It is not impossible that someone could retrieve this information, and there are other ways to do the anonymization so that it would be much harder to find out the original MAC-address. Due to the tight time frame and the fact that this problem isn’t a core problem for this thesis, no such method has been implemented. If the system developed in this thesis is to be used in a commercial setting or otherwise in the “wild”, the writer believes it would be advisable to change the way the anonymization is done in to a more secure way.

The communication between the application and the server before and once a buffer zone is entered is shown in Figure 7 below.
Figure 7
Communication between application and server before and once a buffer zone is entered.
If the device is outside a buffer zone and the device has exited an area of interest the communication will continue as shown in Figure 8 below.

### 6.5 Location Tracking

The application receives location updates from the device, by using the device GPS. When a location update is received the application calculates the distance between the previous location and the new location. The distance is saved in a counter and, when the distance traveled is greater than 15 meters the current location, longitude and latitude, is sent to the server. The server analyzes the received location data. If the device is inside a buffer zone the server responds with information about the area of interest and its points of interest. The response consists of:

- Area of interest center coordinates
- Area of interest radius
- Area of interest ID
- Points of interest center coordinates
- Points of interest radius
- Points of interest ID
- Points of interest event
- Data source for the events
- Points of interest GUI visibility

The application uses this data to construct geofences representing the area of interest and each point of interest as shown in Error! Reference source not found. below.

![Image of geofences in application GUI]

The picture shows how the geofences appear in the application GUI. The blue dot is the device, and the blue circle surrounding it shows the GPS error margin. The big black circle represents the area of interest and the smaller black circles the points of interest. The read markers show the actual points of interest. The black circles are used during the testing of the application and will be removed when the testing is done.

When the geofences are constructed the application stops to send location updates to the server. When the geofence representing the area of interest is entered the application starts saving the coordinates of the device along with a time stamp as “location history”. Entering into a geofence that is representing a point of interest, will as described earlier, trigger the event that the particular point of interest represents. When the device exits the geofence representing the area of interest, the application stops the logging of the device “location history”. The application once again starts consulting the server for information about its location, as described earlier. Exiting the buffer zone after exiting an area of interest will trigger the application to upload the “location history” to the server.
7 Discussion
The application prototype implemented in this thesis has shown that it can handle the following requirements in the tests that has been performed:

- Read device position
- Show multimedia presentations
- Sense hotspots – events triggered based on particular GPS values.
- Log device position.
- Send device movement history to server.
- Collect and send user data to server
- Show and submit form containing questions relevant to the area visited.

Https is used in the communication between the client and the server so the data sent is encrypted and can’t be read or altered while it is in transit. Https works using PKI and SSL certificates and was chosen as it’s a well-known and established protocol used by many websites and applications.

The SHA-512 algorithm was chosen to be used in the anonymization process because of the tight timeframe. It was already integrated in the development environment and ready to use. To greatly improve the strength of the anonymization the BCrypt hashing system could be integrated into the application. This would however have a negative impact on the application performance since the BCrypt system is constructed to use a computationally-intensive hashing algorithm. This means that it would take longer time to compute the unique identifier and that the user might have to wait longer when the server and application are exchanging data that involves the unique identifier.

When evaluating the strengths and weaknesses of the system constructed in this thesis and two guides that are used as comparison, it becomes clear that the system constructed in this thesis combines the strengths given from the hardware of the two other systems. For instance the system suggested by (Björk & Huhtanen, 2011) is stated to be easy to carry around due to that the system uses the uses the user’s smartphone. And the system used by the Digital Guide of World heritage in Falun (Bohlin & Brandt, 2013) is stated to use GPS to guide the users. When it comes to comparing the contents and the narratives of the different systems it becomes problematic because of the fact that the system developed in this thesis does not have any real contents or narratives ready to use.

The architecture of the system allows for the content and narratives to be completely dynamic since all that kind of data will be stored on the server and not in the user’s smartphones.

7.1 System Architecture
The application itself does not hold any of the geofence data or multimedia content that is showed to the user. The geofence data and the data containing the multimedia presentations are stored on the server and sent or streamed to the application when needed. The application itself consists only of the features that are needed to stream the data from the server and to track the device itself. This makes the application very dynamic and the content of the tours
easy to update, since only the information on the server has to be updated. No interaction is therefore needed by the end user to be able to view the new content or to add new areas of interests.

The fact that the application is dependent on the ability to stream content makes the system unusable in areas where the mobile phone network operators have no coverage. The application is also dependent of the GPS system and therefore needs to be able to receive signals from the GPS satellites in order to work. This makes the system unsuitable for indoor use. It will also be hard to get correct GPS readings in environments with much “noise”. This means that in such an environment it is not certain that the geofences will be triggered correctly, if they are triggered at all.

To spot deficiencies in the implementation in the application code, a feature of the development environment called “monkey” could be run. The “monkey” feature is a stress tester that exposes the application to random events that can happen on a smartphone, for example the “monkey” may change some phone settings or send some random input to the application. In other words the monkey will do everything in its power to try to make the application crash. At the time of writing the application has not been tested with the “monkey” and this should be done before releasing the application in to the “wild” to make sure that the application is as stable as possible.

The system that has been developed would benefit from a webpage or some similar feature where the users could get an overview of where the different areas of interests are available and what sort of attraction the areas of interest represents.

During the live testing a bug was detected. The markers on the map that show where the points of interest are located disperse from the map when the user returns from viewing a YouTube clip. This bug was detected on one of the test device running Android 4.0.4, “Ice Cream Sandwich” API 15. But it does not show up when run on the other test device running “Cyanogen mod 10.2.0” Android 4.3 API 18. Further work is required to find out what is causing the bug and how to correct it.

7.2 Scalability
In an attempt to make the system scalable, computational heavy operations, like the computation of the unique identifier, is done on the client side. The server’s workload mostly consists of SQL queries to the backend database and of sending responses to incoming requests.

The audio streaming feature of the system could potentially put a heavy load on the server if many clients were to stream audio content simultaneously. But the audiofiles intended to be streamed consists of navigational directions with short durations and will probably not be used to a greater extent. The audio streaming feature should therefore not have a great impact on the server performance. The primary points of interests will be the ones that trigger video
streaming. These events are handled by YouTube’s servers and won’t put any load on our server.

The server load could be reduced by caching the audio content on the device upon streaming it the first time. This would make it possible to reuse audio files once they have been streamed, reducing traffic to the server and decreasing load times. On the other hand, it might be possible to work out where the device has been by dumping the memory of the device and analyze these audio files. To overcome this threat more cleanup of the device and the applications memory would be needed, this would use more system resources, which in turn would lower the application performance and have a negative impact on the user experience.

7.3 Destroying Location Data

To avoid the possibility that someone could find out where the user has been by recreating the “location history”, using for example file carving on the device storage, that data is never saved to file. The information is kept in the volatile memory of the device as long as it is needed. This means that it will be stored in plain text in the memory of the device. After it has been sent to the server or if the user declines the request to upload the data, it is overwritten with junk consisting of random letters and numbers before the memory assigned to the variables is handed back to the system. This is done to prevent attempts to recreate the deleted “location history” by dumping the memory of the device.

The way that Android handles memory makes it impossible for the programmer to decide when the memory of a variable should be reclaimed by the system. All that can be done is to terminate all references to the variable. The Dalvik virtual machine will then decide when the garbage collector will make its run and reclaim the parts of the memory no longer referenced. Even if this is done there is no way to know when the memory previously used by the variable will be overwritten with new information. As long as the memory range previously used by the variable is unused, the old information will still be in the memory and it would be possible to recreate this data by analyzing the dumped memory.

This raises other problems as well. The uploading of the location history is done using a separate thread. When the data is prepared to be sent to the server it is processed, and has to be stored temporarily in a key value variable before it can be sent to the server using the POST method. There is no way to overwrite this key value variable. All that can be done is setting the variable reference to NULL. This means that the data that resides in this variable will remain in the memory, although unreferenced, until it is overwritten.

The variable that contains the location history is a list variable and each time the location history is updated a new entry in the list is made. When it is time to clean the location history list variable, a loop goes through each entry in the list and replaces the current contents with a new string of equal length that contains random numbers and letters.
for(int i = 0; i < locHistory.size(); i++)
{
    // Replace the current location data with random junk
    locHistory.set(i, mumboJumbo(locHistory.get(i).length()));
}

// Clear the location history
locHistory.clear();

The string containing the random numbers and letters is produced by another loop which does one iteration for every entry in the string that is being produced until the string has the desired length.

// Variable used to hold the junk
String mumboJumbo = "";

// run operation until desired length is reached
for(int i = 0; i < length; i++)
{
    // append a random character to the string
    mumboJumbo += ALPHAS.charAt(r.nextInt(ALPHAS.length()));
}

// return the junk string
return mumboJumbo;

This means that the time and resources needed to make the cleanup depends how much location history that has been produced and the length of each entry in the history. The length of the entries depends on the accuracy of the GPS reading for the coordinates that are being stored. Higher accuracy produces longer entries, since the GPS coordinates contains more digits when the accuracy is higher.

Things could be done to improve the execution speed of cleanup code. One such thing would be to make the variable controlling the loop iteration in the for-loops static. As the code is written now, a method call is made each iteration, when the loop iteration count is compared to the desired loop length, this slows the execution:

for(int i = 0; i < locHistory.size(); i++)

Removing this method call and replacing it with a static variable would speed up the process. Faster code:

int maxIterations = locHistory.size()
for(int i = 0; i < maxIterations; i++)

This change would not be noticeable on a desktop computer but on a device like a smartphone this could make a big difference if the loop is doing many iterations.

The cleanup is only performed when the user is done taking a tour and the location history has been sent to the server. It is performed on a separate thread, so even if the cleanup would take
some time the GUI will not freeze, the user will be shown a progress dialog. The cleanup should therefore not have a big impact on the overall application performance.

If the volatile memory of the device was dumped before the application has sent the “location history” to the server, it will be possible to retrieve the location history from the dumped memory.
This situation will occur if for example the application is put to the background before the application has exited the area of interest and/or the buffer zone. If the application is in this state the location history will not be sent, although the device has left the buffer zone. This is hard to avoid, due to that the users approval is required to start the upload of the location history.
An improvement to the application would be to make the user observant of that the area of interest and buffer zone has been exited if the application is in the background by sending a notification to the user.

Due to the tight timeframe the result of the countermeasure towards the dumping of volatile memory implemented in the application has not been verified by actually dumping the memory of a device and analyzing it.

If there are other applications/services running on the device that utilizes the GPS it’s possible that some, or the entire, device location history will be available through this application/service memory. This is nothing that can be prevented.
There are also other methods of tracing a smartphone. For instance wireless networks that the phone encounters can be logged. It is also possible to use the properties of the GSM network to trace the phone. If an application/service like this are running on the phone it would probably be possible to recover the phones location history by examining these applications/services.
In other words it is extremely hard, if not impossible, to remove all data from the device that might indicate where the device has been.

7.4 Google Maps Trouble
The Google maps API used by the application caches map information to the device external storage. This cache cannot be cleared by the application; the user has to clear it manually. This cache may therefore compromise the user’s integrity, because the user’s movement history could be recreated by using the information in the cache. A step towards cleaning the data from the device that can indicate where the user has been would be to clear the Google maps cache. However at the time of writing there are no way to do this programmatically, so this is a possible way to find traces of where the user has been after the application has been used, despite the actions taken to try to remove this kind of data from the device.
7.5 Application Comparison
The application produced by (Björk & Huhtanen, 2011) features an application for smartphones based on a Webb interface. The user can view pictures and read short texts about different point’s interests in Elfsborg, a part of the World heritage of Falun. The application uses a static map of the area that is covered by the guide. Points of interest are marked on the map allowing users to navigate there by the selves. Clicking the on a marked point of interest on the map will display the information available for that point of interest. In their thesis, Björk and Huhtanen compare their application to another digital guide. The application developed in this thesis is evaluated and compared using and extending the comparison made by Björk and Huhtanen in their thesis as shown in Table 1 found in Appendix B Tables.

8 Conclusion
The application prototype implemented in this thesis has shown that it can handle the following requirements in the tests that has been performed:

- Read device position
- Show multimedia presentations
- Sense hotspots – events triggered based on particular GPS values.
- Log device position.
- Send device movement history to server.
- Collect and send user data to server
- Show and submit a form containing questions relevant to the area visited.

The data sent between the application and the server are encrypted through the use of https, and it is therefore difficult to gain access to this data while it is being sent over the network. Attempts has been made to clear the memory of the location history once it has been sent to the server. The implemented cleanup method has not been tested and has not been proven to work as intended in an empirical fashion.

The implemented cleanup method is believed to have flaws since at least one of the variables used, when sending the location history to the server, is unaffected by the cleanup process and might therefore leave traces of the location history in the volatile memory of the device. Furthermore the implementation of the Google maps API in the application makes it possible to gain access to map information through the Google map cache. There are no way to programmatically empty this cache and this information will therefore remain on the device after the application has been used. This information could be used to retrieve information about where the user has been.

Furthermore there might be other applications/services running on the phone that are gathering data that could be used to retrieve the location history of the phone.

In other words it is hard if not impossible to clean the device from all the data that can revile where the device has been after the application has been used.

The implementation of the memory cleanup method can be altered to make the code run faster. The application performance or the user experience is not believed to be affected in any major ways by the cleanup.
8.1 Things To Address

The bug discussed in 7.1 System Architecture needs to be corrected in order for the application to work properly on all supported API platforms. To further ensure the applications stability, the monkey feature of the development environment needs to be run, this would surely reveal further bugs that also would need to be corrected.

The anonymization process would benefit from implementing the BCrypt system. This would result in that the anonymized data would be much harder to trace back to its source.

Besides from correcting bugs in the application, the implemented code would also benefit from a review to find sections of code and implementations that can be improved in order to enhance the application performance and reduce system load, memory footprint, battery usage and other similar things that are effecting the applications performance.

The application would also benefit from the implementation of some kind of feature that would assist the users, for example a webpage where it is shown where the areas of interest can be found and some basic information about the areas of interest could be assimilated. Otherwise the users would have to know where the areas of interest are located before they start using the application.

8.2 Feature work

As mentioned in 7.3 Destroying Location Data the countermeasures taken towards attempts to recreate the user’s location history by dumping the volatile memory from the device has not been confirmed to work. This needs to be done in order to evaluate the countermeasures used in an empirical manner.

Furthermore the system needs to be tested in some kind of a larger live test to see how the system handles different user loads and to and determine how well the system scales.
9 Bibliography


Appendix A UML Diagram

Main Activity
Extends: ActionBarActivity
Implements:
  LocationListener
  ConnectionCallbacks
  OnConnectionFailedListener
  OnAddGeofencesResultListener
  OnRemoveGeofencesResultListener

ReceiveTransitionsIntentService
Extends: IntentService

interface MPInterface
void playFile(String location);
void pause();
void stop();

PrivacyPolicy
Extends: Activity

UserInfoCollection
Extends: PreferenceActivity

UserDataSender
Extends: AsyncTask<Void, Void, String[]>

AudioPlayer
Extends: Service
Implements:
  OnCompletionListener
  OnPreparedListener

LocationTracker
Extends: AsyncTask<Location, Void, String>
Private Class UserLocationHistorySender
Extends: AsyncTask<List<String>, Void, String>

TheYouTubePlayer
Extends: YouTubeBaseActivity
Implements:
  YouTubePlayer.OnInitializedListener

ViewForm
Extends: Activity

interface MPInterface
void playFile(String location);
### Appendix B Tables

<table>
<thead>
<tr>
<th>E-guide</th>
<th>Prototype Elfsborg</th>
<th>Digital Guide of World heritage in Falun</th>
<th>TrackIT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Geographic navigation</strong></td>
<td>Map without guidance</td>
<td>GPS with guidance</td>
<td>GPS with possibility of guidance</td>
</tr>
<tr>
<td><strong>Instructions / help</strong></td>
<td>Instructions integrated in the GUI</td>
<td>Pauses for continuous instructions</td>
<td>Possibility of guidance through Sound</td>
</tr>
<tr>
<td><strong>Dimensions</strong></td>
<td>The user's smartphone, with the relative size of the application</td>
<td>PDA in a large shell</td>
<td>The user's smartphone</td>
</tr>
<tr>
<td><strong>Structure of content</strong></td>
<td>Ability to search on the map or in lists</td>
<td>Narratively presented content</td>
<td>Narratively presented content</td>
</tr>
<tr>
<td><strong>Access</strong></td>
<td>Download to users smartphone</td>
<td>Retrieve from a museum</td>
<td>Download to users smartphone</td>
</tr>
<tr>
<td><strong>Audio</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Multimedia</strong></td>
<td>Yes, some</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Navigation (own choice)</strong></td>
<td>Yes completely</td>
<td>Yes, to some extent</td>
<td>Yes to some extent, it depends on the narrative structure</td>
</tr>
<tr>
<td><strong>Extended Features</strong></td>
<td>Access to the built-in features of the smartphone, such as the camera.</td>
<td>No</td>
<td>Access to the built-in features of the smartphone, such as the camera.</td>
</tr>
<tr>
<td><strong>Update content</strong></td>
<td>Dynamically updated on the server automatically downloaded</td>
<td>Performed individually on each PDA</td>
<td>Dynamically updated on the server automatically downloaded</td>
</tr>
<tr>
<td><strong>Contents</strong></td>
<td>Transparent, memorable, evokes curiosity</td>
<td>Perceived as being too massive, but entertaining</td>
<td>Depends on the information made available through the server</td>
</tr>
<tr>
<td><strong>Weaknesses</strong></td>
<td><img src="#" alt="List of weaknesses" /></td>
<td><img src="#" alt="List of weaknesses" /></td>
<td><img src="#" alt="List of weaknesses" /></td>
</tr>
<tr>
<td><strong>Strengths</strong></td>
<td><img src="#" alt="List of strengths" /></td>
<td><img src="#" alt="List of strengths" /></td>
<td><img src="#" alt="List of strengths" /></td>
</tr>
</tbody>
</table>

Table 1: Comparison between different digital guides

32