University of Luxembourg

Bachelor Thesis

Autogenerating context to manage sensitive data in Android-phones

Author:
Xinxin ZHU

Supervisor:
Hugo JONKER

A thesis submitted in fulfilment of the requirements for the degree of Bachelor in Computer Science

in the

Security and Trust of Software Systems (SATOSS)
of the
Faculty of Science, Technology and Communication

June 2013

Declaration of Authorship

I, Xinxin ZHU, declare that this thesis titled, 'Autogenerating context to manage sensitive data in Android-phones' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"All human beings have three lives: public, private, and secret."

Gabriel Garcia Marquez

UNIVERSITY OF LUXEMBOURG

Abstract

Faculty of Science, Technology and Communication

Bachelor in Computer Science

Autogenerating context to manage sensitive data in Android-phones

by Xinxin ZHU

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too…

Acknowledgements

I would like to express my gratitude towards Hugo JONKER, the project advisor and local chief who helped me constructing the thesis and provided the ideas for this thesis.

Contents

List of Figures

Abbreviations

SDK	**S**oftware **D**evelopment **K**it
API	**A**pplication **P**rogramming **I**nterface
GPS	**G**lobal **P**ositioning **S**ystem
ISP	**I**internet **S**ervice **P**rovider
DRM	**D**igital **R**ight **M**anagement
WPA2	**W**i-Fi **P**rotected **A**ccess **2**
SIM-card	**S**ubscriber **I**dentity **M**odule card
AP	**A**ccess **P**oint
WOEID	**W**here **O**n **E**arth **ID**entifier
YQL	**Y**ahoo **Q**uery **L**language

Chapter 1

Introduction

In recent years the the number of users connected to the internet has grown. This is due to the popular smartphone which leads to the explanation of the popularity of the social networks [1].

The Figure 1.1 shows the rise of global social networking audience from 2007-2011.

Source: comScore Media Metrix, Worldwide, March 2007 October 2011

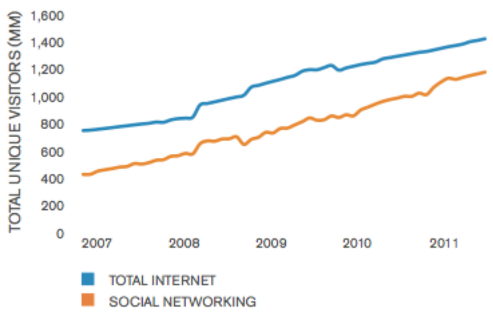


FIGURE 1.1: The Rise of the Global Social Networking Audience

In a time, where performing activities on social networks became a daily routine for the most people that are connected to these networks. The attention of developers who are active in the domain of software engineering should be focused on the privacy protection, since the internet presence has a significant impact in real life. For an example, let us assume a person attending his job

[1]

- Facebook with 51% of total internet users.
- Twitter with 22% of total internet users.
- Google+ 26% of total internet users.

Source: GlobalWebIndex, Steam social: Quarterly Social Platforms Update Q1 2013 interview and the interviewer checks the candidate on the internet and he finds pictures of the candidate. Some pictures show the candidate in embarrassing situations, hence the interviewer makes some assumptions and the job interview fails because the pictures indicate that the candidate is not a serious person.

To prevent the problem of having such a negative internet presence that is caused by the lack of attention from the users and partly popular social networks. The privacy of the users should be protected accordingly to the growth of their internet presence so that it doesn't affect their real life in a negative sense. For example Mark Zuckerberg, the creator and CEO of the well know social network *Facebook* never thought about how to enforce the privacy protection of the users because to make the business model from *Facebook* more successfully, user information must be gather as much as possible.

The default policy of privacy protection on *Facebook* did not improved at all but in contrary it deteriorated with the time. The Figure 1.2 displays the deterioration of user privacy in *Facebook* from year 2005 to 2010.

[2] `http://mattmckeon.com/facebook-privacy/`

Source: Matt McKeon[2]

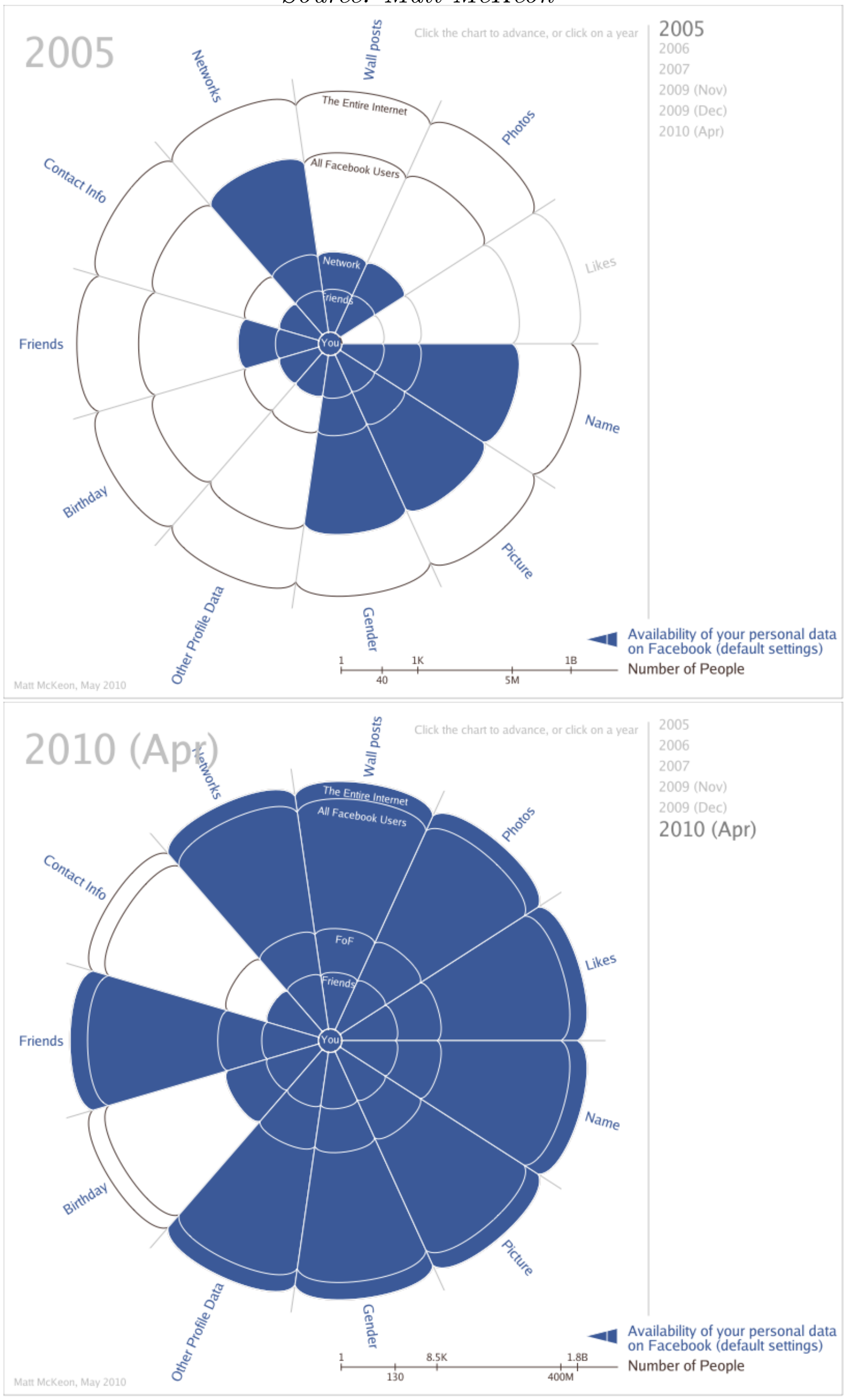


FIGURE 1.2: The Evolution of Privacy on Facebook 2005/2010

1.1 Motivation

When a memorable event occurs, one might like to have a memento of the event and share it with other people. The best way to do that, is to take a picture and share it, but depending on the moment one might consider when the picture should be viewable or not. For example a person is at a party and he took a photo which shows himself in an inappropriate position that could harm his reputation. This person still decides to share it with his friends who are at another party, since his friends know how he really is. This picture shouldn't be accessible any more when the receiver is at work or at school since the sender doesn't want to be seen in an inappropriate position by unauthorised people. Another example, at work the person attends a meeting and he took a valuable picture which he shares with his colleagues but this picture contains secret company information therefore the photo should only be viewable for his colleagues when they are working too.

Hence the question comes up now if it's possible to share moments of a certain event with the follow men. In a form of pictures taken by a device which are accessible if and only if the addressed unit is in similar circumstances as the device that took the picture. With this setup the receiver can only access the senders moments in similar contexts.

The procedure of identifying the circumstances or contexts are explained and shown in this thesis with Android SDK and java on an Android smart-phone.

1.2 Related work

1.2.1 SnapChat

Snapchat is a photo messaging application developed by four Stanford students. Using this application, users can take photos, record videos, add text and drawings, and send them to a controlled list of recipients. Users set a time limit for how long recipients can view their photos, up to 10 seconds. After the the countdown is due the picture should be deleted from the recipient's device and the company's servers.

This application is a good reference for this thesis since it has similar features and the same idea of sharing photos, but it lacks of policy controls since every Snapchat user can view the pictures at any given time. The meaning of thesis is to implement a system that allows the application to generate automatically contexts for the purpose of access control on the pictures to make them accessible only if the viewer has similar contexts.

1.3 Research scope and approach

The scope of this thesis is to do a exploration research on context information that can be collected by an Android phone, further on, these collected data from different kind of sources are going to be processed into context which is assigned to the user-generated data. Using this approach, further researches in context modelling and context similarity checking are done.

The researches are related to the work package“Privacy-aware sharing” from the project “PRIVACY-BY-DEFAULT-IN-DIGITAL-LIFE” of prof. dr. Sjouke Mauw, where the objective is to create a method for comparing two contexts for privacy and a scheme that preserves the context of user-generated content.

1.4 Contribution

The thesis is contributed in two main parts. In the first part, we describe the possible information sources on the smart-phone and which and how we are going to process the data. This part includes Chapter2 and Chapter3.

In the second part we will show the implementation as an application on a smartphone with the necessary features and sensors. This part consists of Chapter4.

1.5 Structure

Chapter 2

Information sources

In this Chapter possible information sources that can be retrieved by an Android smartphone are listed and explained by which source it gets leveraged from. These raw data is later on processed into an context in Chapter 3.

2.1 General

The information are collected from different sources, for example sensors of the device running the application and/or provided from different services on the internet. After a lot of research on how to extend the scale of these information, Yahoo provided some good APIs which may be used to leverage further context information from. Possible data that could be used to parse contexts are listed in this section.

The sources of context information that can be obtained with an android smartphone are divided in two main categories: "Direct sources" and "Indirect third party sources". Context information that can be obtained by the "Direct sources" that are known and can be leveraged by the android SDK are listed in the Table 2.1. While all context information that were retrieved from "Indirect third party sources" are listed in the Table 2.2. We make the distinction between these two categories because the direct sources are available within the Android API and indirect sources needs to be fetched from the internet via HTTP[1] requests.

Direct source:

[1]HyperText Transport Protocol

- Temperature
- Surrounding noise level
- Acceleration of the device
- Air pressure
- Phone number of the user
- Schedules
- Available wifi AP
- Contact list of the phone
- Date and time
- Movement Speed

Indirect third party source:

- City Name
- Country
- Zip code
- Wind speed
- Atmospheric temperature
- Air pressure
- Atmospheric humidity
- Weather Condition
- Sunrise time
- Sunset time

2.2 Direct sources

Direct sources are sources provided by the Android SDK and by the sensors of the mobile phone. Further information and details about the direct sources are explained in this chapter.

GPS sensor

By measuring the signal strength from the GPS sensor to the GPS satellites, one can derive if the device is located inside or outside of a building. The device movement speed can also be leveraged from Android using the GPS sensor. With the movement speed one we can define if the user is currently travelling or not.

Location

The location information consists of the geographic latitude and longitude and is a key factor for the indirect third party sources, this information can be obtained via the GPS-provider or by the network-providers. This information is provided by the Android API using GeoCoder[2] class.

Light sensor

With the light sensor we can measure the ambient light intensity of the surrounding environment. Dark rooms and well lit rooms can thus defined.

Humidity sensor

The humidity sensor measures the ambient humidity. Thus context where water is nearby can be defined, for example “beach”.

[2]`http://developer.android.com/reference/android/location/Geocoder.html`

Temperature sensor

The current smartphones have two thermometer which measure the CPU and battery temperature expect the latest smartphone from Samsung, the Galaxy S VI, it features also an ambient temperature sensor that measures the room temperature. With the temperature we can define whether the device is in an environment where people are dressed lightly or not. For example when it is less than 18C people are dressed more than when the room temperature is 30C.

Microphone

The microphone measures the surrounding noise level. Lively places like a party are really noisy but calm places like a library, there is almost no noise.

Accelerometer

The accelerometer allows the Android device to measure it's own acceleration. This sensor is rather useless to determine context when photos are taken. Since taking pictures in motion, they get blurred.

Gyroscope

A gyroscope is a device for measuring or maintaining orientation. The use of this sensor for context reasoning could not be found yet.

Compass

A compass shows directions in a frame of reference that is stationary relative to the surface of the earth. The use of this sensor for context reasoning could not be found yet.

Barometer

With the air-pressure data measured from the barometer we can derive an approximate weather condition. With the air pressure we can derive if it's going to rain.

Phone number of the user

To identify the uploaded data of the devices, the phone number could be used since the phone number is unique in every country.

Google Calendar

The Google Calendar is a free time-management web application offered by Google and its events are accessible via the Android SDK. Based on the event one can derive if the user is in a meeting by looking up when the event is hold.

Wifi Adapter

When there is no AP available nearby, the device is situated in an out skirt location, thus we deduct that the user is currently travelling.

Contact-list of the phone

Using the data from the contact-list, for example the phone number of the contact, the access policy takes a further step.

Date and time

The date and time can be easily retrieved from the Android system itself. The internal clock of the Android OS is used to determine the time and date. With the time, we can derive in combination with the sunset and sunrise information if the photo was taken during day or night time.

2.3 Indirect third party sources

Indirect third party sources are obtained from other companies that provides information which could be parsed into context information. These sources require internet to be available. Below two such sources are discussed, there exist others such as Google Places API[3] or Weather Underground [4], these weren't chosen because the number of calling to these APIs were limited or they weren't free.

2.3.1 PlaceFinder API by Yahoo

The Yahoo PlaceFinder API takes longitude and latitude as input and generates an XML document and returns it as one string. The document contains information about the location for example which country it belongs, the WOEID of the location and even more precise, the street name if this information is accessible for the provided longitude and latitude.

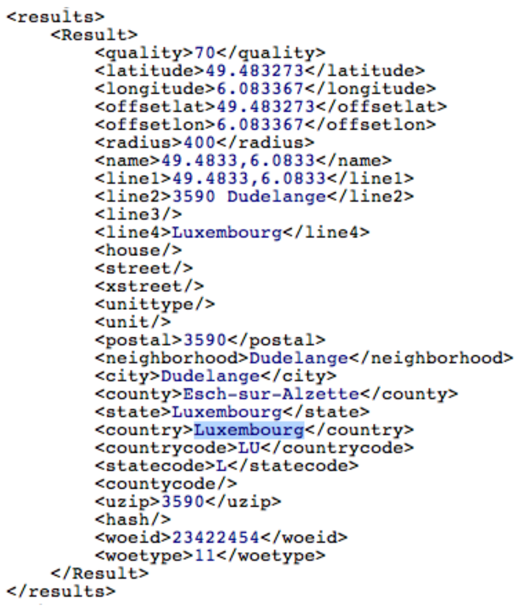
The API is accessed via HTTP Request by using a query in YQL[5] for example:
*select * from geo.placefinder where text="49.4833,6.0833" and gflags="R"*

40.4833 is the latitude value and 6.0833 is the longitude value of a location, for this example the longitude and latitude from the city Dudelange were taken. This query returns an XML formatted string which is shown in Figure 2.1.

2.3.2 Weather API by Yahoo

[6] Weather information were initially accessible via a Google API but Google abandoned their weather API along with iGoogle, therefore the Yahoo weather

[3] `https://developers.google.com/places/documentation/search`
[4] `http://www.wunderground.com/weather/api/`
[5] `http://developer.yahoo.com/yql/console/`
[6] `http://developer.yahoo.com/weather/`



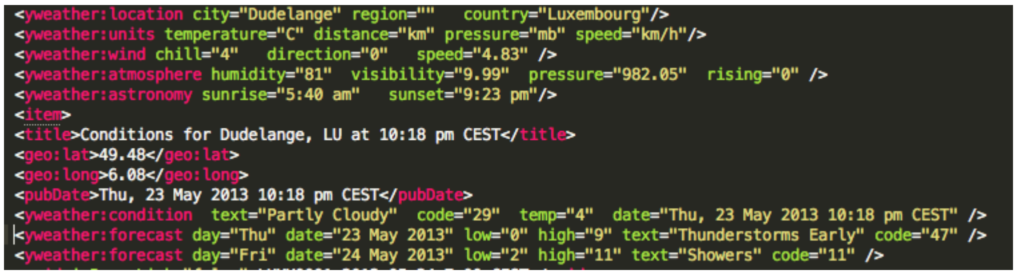
```xml
<results>
    <Result>
        <quality>70</quality>
        <latitude>49.483273</latitude>
        <longitude>6.083367</longitude>
        <offsetlat>49.483273</offsetlat>
        <offsetlon>6.083367</offsetlon>
        <radius>400</radius>
        <name>49.4833,6.0833</name>
        <line1>49.4833,6.0833</line1>
        <line2>3590 Dudelange</line2>
        <line3/>
        <line4>Luxembourg</line4>
        <house/>
        <street/>
        <xstreet/>
        <unittype/>
        <unit/>
        <postal>3590</postal>
        <neighborhood>Dudelange</neighborhood>
        <city>Dudelange</city>
        <county>Esch-sur-Alzette</county>
        <state>Luxembourg</state>
        <country>Luxembourg</country>
        <countrycode>LU</countrycode>
        <statecode>L</statecode>
        <countycode/>
        <uzip>3590</uzip>
        <hash/>
        <woeid>23422454</woeid>
        <woetype>11</woetype>
    </Result>
</results>
```

FIGURE 2.1: XML from PlaceFinder API.

API is considered as an alternative solution. This API takes a WOEID as input, where the WOEID a unique 32 bit reference identifier is that is assigned by Yahoo to identify any featured place on earth. This API generates an XML document and returns it as a string, the document contains weather information about the location, which is identified by the WOEID.

It is accessed via a HTTP request

In the Figure 2.2 all context information that can be retrieved with the Yahoo Weather API are shown in an XML formatted text.



```xml
<yweather:location city="Dudelange" region=""   country="Luxembourg"/>
<yweather:units temperature="C" distance="km" pressure="mb" speed="km/h"/>
<yweather:wind chill="4"   direction="0"   speed="4.83" />
<yweather:atmosphere humidity="81"  visibility="9.99"  pressure="982.05"  rising="0" />
<yweather:astronomy sunrise="5:40 am"   sunset="9:23 pm"/>
<item>
<title>Conditions for Dudelange, LU at 10:18 pm CEST</title>
<geo:lat>49.48</geo:lat>
<geo:long>6.08</geo:long>
<pubDate>Thu, 23 May 2013 10:18 pm CEST</pubDate>
<yweather:condition  text="Partly Cloudy"  code="29"  temp="4"  date="Thu, 23 May 2013 10:18 pm CEST" />
<yweather:forecast day="Thu" date="23 May 2013" low="0" high="9" text="Thunderstorms Early" code="47" />
<yweather:forecast day="Fri" date="24 May 2013" low="2" high="11" text="Showers" code="11" />
```

FIGURE 2.2: Yahoo Weather XML Document.

Chapter 3

Context

In this chapter my definition of context, the way of how to model context from the raw data provided by the sources mentioned in Chapter 2 and the comparison of the contexts are going to be explained.

3.1 Context definition

3.1.1 General

Contexts are ideas and meaning interpreted from a set of information and thus context derived from the set of information may vary depending on the unit that evaluates the set. For example in natural language, a text can be interpreted in different ways depending on the person processing it. The variation of context poses a problem in computing domain since a computer can not argue with other computers what the real context of an information source is.

Thus the processing and similarity comparison of contexts are the main parts of this chapter. Another definition has been made by Anind K. Dey and Gregory D. Abowd [1].

Their definition of context:

'Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.'"

3.1.2 Impact of context in privacy protection

Assume the scenario of a set of people attending a party. A person who is a member of the people that attends the party, takes a picture during the party which contains privacy sensitive data that could cause problems to him if unauthorised people see it. For example the picture shows content that would damage his reputation. The best solution to this problem would be not to take pictures at all but he wants to keep some memorials of that party that he wants to share with authorised people later on. Another solution would be to delete the picture after he has shown to authorized people, this solution eliminates the potential threat posed by the picture but it would also eliminate the idea of memorial.

A better approach would be to infuse context to the photo that describes the privacy level of the data, so that access control rules can be defined to protect the privacy sensitive data from unauthorized access.

3.2 Processing context information into context

Android SDK and the sensors from the the smartphone provide a large set of context information and on top of that the context information from third party sources provides further extensions to this set. One context information can have different magnitude in different use cases of the contexts.

Every country has their own DRM laws for example in Germany, GEMA [1] forbids every video on *Youtube* that has copyrighted audio content in Germany. This restriction from GEMA is only valid in Germany therefore in Luxembourg these videos with copyrighted sound content are available.

This idea can be leveraged and implemented to graphical files, thus infusing context information to the files and applying the DRM laws. In this use case the context information about the origin of the data are valuable. Taking the country name for example. This information contains a string which holds no information about the privacy of the user but the origin of the content. Thus comparing this information from two contexts would be meaningless in the case of privacy control but this information holds a greater value in the example with Germany.

[1]

- DE: Gesellschaft fr musikalische Auffhrungs- und mechanische Vervielfltigungsrechte.
- EN: Society for musical performing and mechanical reproduction rights.

To prevent GI/GO(garbage in/garbage out) the large quantity of context information must be therefore weighted coherently to the use case before they are processed into context.

3.2.1 Context model

To analyse context similarity in the Android environment, a suitable context model is required. Thus the question: "How to model context in a generic way that a computer can process the context data?" comes to mind.

The multiple sensor data fusing model is used in this thesis. In this model the context information from different sensors and sources are gathered firstly, afterwards they get analysed and processed further into a context. This model is simplistic but it is well suited for the Android environment, due to the big quantity of sensors and sources information available on Android smartphones.

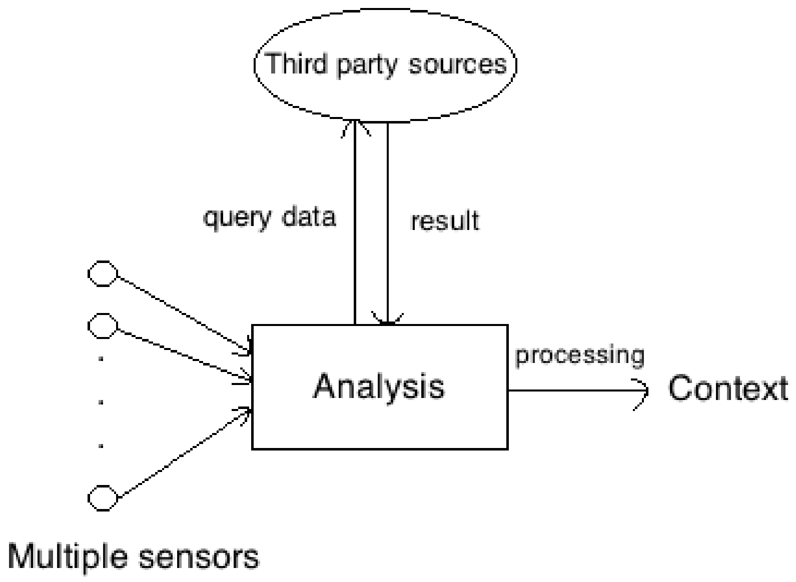


FIGURE 3.1: Mutiple sensor data Model.

3.2.2 Context information analysis

As mentioned in the example before, a context information can bear different contextual information values, therefore the supplied context information don't suffice to define a contextual situation. Thus these information are regarded as raw input data and therefore they need to be analysed on their contextual information value and refined. The analysis is conducted in three steps.

Step 1:

Assume S is the set of context information gathered from the different sources and x is an element of S such that $S = \{x : x \in S\}$. Then $\forall x$ are analysed independently from each other. In this step those context information that bears valuable information determined by the analysis, are inserted to a new empty set of context information *S'*.

Step 2:

After the first step, combinations of different x are considered. When a combination bears valuable contextual information, it is considered as a new element *x'*. The newly formed element *x'* is then added to the set *S'*.

Step 3:

In this third and last step, the elements from the newly created set *S'* are weighted for their importance in context derivation. The importance of each element in *S'*, is then transformed into a priority system with different level values. Assume n is the number of elements in *S'*, then the number of priority level $p \leqslant n$ and each priority level has an different impact level on the context determination. To model the impact, values depending on the level of impact are assigned such that the highest model has the highest value and the lowest impact as the lowest value. For example in locational context, the country's name has a bigger impact than the the atmospheric temperature and thus the country's name has a higher value.

Unfortunately the context information analysis can not be done automatically due to the inability of context reasoning for the coherent use case of the contexts by the computer. Thus the analysis is done by human hand for now and the context is determined from the analysed context information combined with the coherent priority level.

3.3 Context similarity

With the analysed context information and its according priority level, their impact values are compared from the current context ContextA and the "to be compared context" ContextB.

For every context information from both contexts, a different tolerance level is considered based on their impact to the context. For example, to determine the locational difference of the contexts, a tolerance of 100 meters is considered for the similarity checking. Another example, to determine whether night time or day time, there is zero tolerance. The task now is to decide whether the ContextA is similar, equal or different compared to ContextB. For this purpose a point scheme is implemented. Thus a differential value that defines how much two contexts differs from each other in implemented. The differential value can vary from zero to a maximum value. The maximum value is the sum from the evaluated impact values of each analysed context information in a context. An acceptable limit for the differential value is thus defined to identify whether ContextA is similar or different to ContextB.

$ContextA \equiv ContextB$

When both contexts are equivalent, then the differential value would be zero.

$ContextA \not\equiv ContextB$

When both contexts are completely different, then the differential value would be maximum.

$ContextA \sim ContextB$

When ContextA is similar to ContextB, then the differential value is in an acceptable limit that is predefined and adjust on the use case. Similar contexts are considered as equivalent contexts.

Chapter 4

Case Study: PrivyShare

4.1 SnapChat

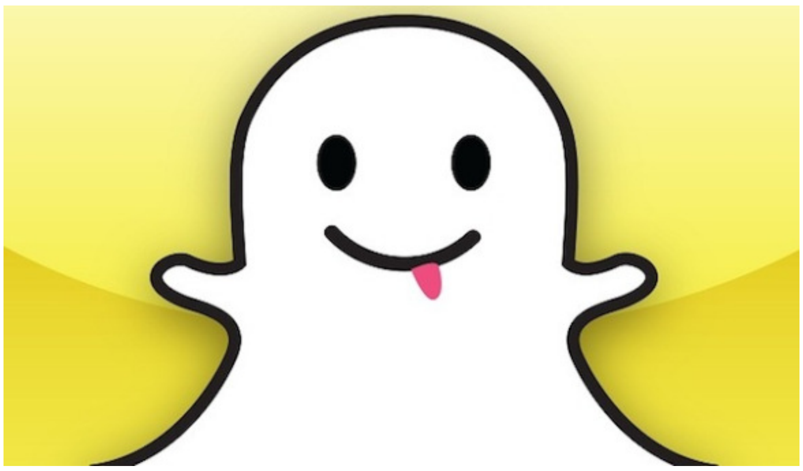


Figure 4.1: SnapChat Logo.

As mentioned before in Chapter 1. SnapChat is a popular application to share pictures and videos with friends but it has several issues and these issues are described in this section which are handled in the use case of this thesis.

4.1.1 Security issues

Unfortunately since the application was released, several hacks were published that allowed the user to save the picture or videos before it's deleted from the servers and the receivers local phone.

SnapChat has a feature that enables the user to take a screenshot while viewing the photo and the sender will be notified after the screenshot was taken but there is a possibility to take a screenshot without the sender being notified.

This glitch only works on iPhone devices; directly after the screenshot is taken, the user needs to open the task manager by pressing the home button twice. By doing this, the sender notification code won't be executed since the iPhone won't recognize that a screenshot has been taken in the process of task manager opening.

Another problem posed by the application is the feature called poke, poke is simply a video version of the pictures but the privacy protection of the pokes are worse than the pictures. Using iPhone directory browser applications, one can access the video file directly from the PC since the videos are also saved locally. Further on the user just need to copy the file before it's opened because after opening it, the local copy will be deleted with the copy on the server.

4.1.2 Other issues

Apparently the photos on the mobile phone are not deleted after they are viewed in SnapChat but instead, they are renamed into".NoMedia" file extensions. This extension makes the files inaccessible for the device but the files are still on the storage of the phones.

4.2 PriviShare

PriviShare is a use case application to demonstrate the context generation and privacy handling of the sensible data with contexts. This application is written with Android SDK, which features java as programming language. The key feature is the sharing of graphical contents in a secured way that avoids all the issues posed by SnapChat. It also has an enforced access control that enables the policy to control whether the content is accessible or not.

To identify the sender and the receiver, IDs are needed. At first, the owners phone number of the SIM[1] card should be used to identify the sender but the problem here is that not all SIM cards sold have its own number registered in the memory because the operator decides which initial contacts the SIM card should contain.

Therefore an user ID is set on the first run of the application, this ID is unique since it is used to identify users. Before a picture is taken, the receiver is indicated by the sender. Android has a feature to make an user interface of an application private so that no screenshot can be taken. This feature is used to avoid illicit screenshots made by the receiver after the pictures were opened.

[1]SIM: Subscriber identity module

The contact list from the Android phone should be referred as the list of possible receivers but with the problem stated above the list can not be used. Hence, the assumption that senders know the receivers ID is made.

When pictures are taken or opened, the current context information must be analysed, contexts generated and encoded, these steps are explained further in the subsections with the encryption and file format part.

For demonstration purpose the user ID can be changed to simulate multiple users during the run and only those pictures which are designated to the user ID are listed, this simulates the sharing feature. This application is build under the assumption that the phone has internet access, at least Android API verison 4.1 and the required sensors.

4.2.1 List of context information used in PriviShare

1. Information from direct sources
   (a) Geographical coordinates.
   (b) Current time.
   (c) Wifi APs availability.
   (d) GPS signal strength.
   (e) Ambient light intensity.
   (f) Ambient temperature.
   (g) Ambient humidity.
   (h) Ambient noise level.
2. Information from indirect sources
   (a) Atmospheric temperature.
   (b) Atmospheric humidity.
   (c) Weather condition.
   (d) Sunrise time.
   (e) Sunset time.

4.2.2 Context Analysis

Unfortunately the context information *Ambient temperature* and *Ambient humidity* couldn't be used in the use case because the test device didn't support the required sensors. With the combination of these two information, contexts with near water property for example *beach* or *bath* could be derived.

The first idea was to classify the contexts into event categories based on the time when the picture was taken. Thus four events are created.

Morning event

A/a

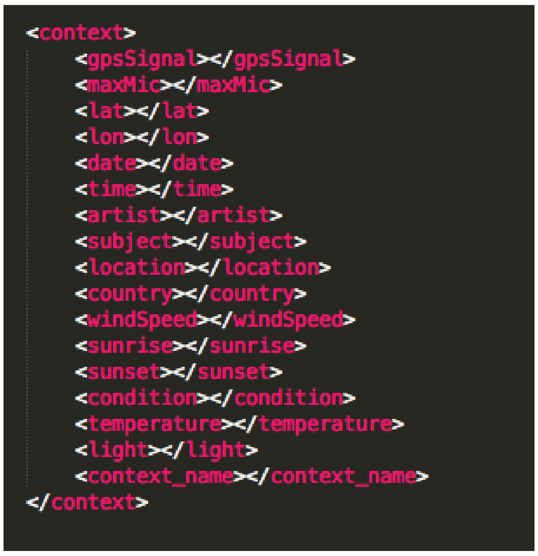
A/a

A/a

night event from 23h00-6h00, afternoon event from 12h00-18h00, evening event from 18h00-23h00 and morning event from 6h00-12h00 ... ...

4.2.3 Context Generation and Encoding

Two possibilities were considered to encode contexts. Encode the context as an object which is serialized into a binary file but this solution is not scalable and therefore the contexts are encoded in XML[2] format, since XML is both human and computer readable. As in the name already stated, this format is extensible and thus its scalability property allows the implementation of future extensions. The structure of the encoded context is shown in Figure 4.2.



```
<context>
    <gpsSignal></gpsSignal>
    <maxMic></maxMic>
    <lat></lat>
    <lon></lon>
    <date></date>
    <time></time>
    <artist></artist>
    <subject></subject>
    <location></location>
    <country></country>
    <windSpeed></windSpeed>
    <sunrise></sunrise>
    <sunset></sunset>
    <condition></condition>
    <temperature></temperature>
    <light></light>
    <context_name></context_name>
</context>
```

FIGURE 4.2: Context structure.

[2]XML: Extensible Markup Language

On the senders end, the main idea was to implement a generic way of context generation, such that every time a picture has been taken, no user interaction is needed to generate a context. On the receivers end, the similarity check is done on the context before its coherent picture can be viewed. Depending on context similarity, the access is granted or denied.

The problem of this idea is that different contexts have similar context information. After a lot of thoughts and investigations, I concluded that with the current smartphones and technology, it is impossible to identify if one is at a party or watching a film in a movie theatre. Both have similar context information for example it is very noisy, dark and there is a lot of people attending this event. Another example would be *Home* and *Office*, the only thing that differentiate them is the location and the time when the picture was taken. These information do not suffice to make a difference in the similarity analysis, hence making these two contexts similar when they are clearly not. The analysis results in a false positive.

To resolve this problem a naming mechanism is introduced. An initially empty list of known contexts is implemented. This list is scalable and has redundancy property, meaning that it can contain multiple contexts with the same name . Every time a picture has been taken the current context information of the phone is checked with the contexts in this list. When no exact context is found, a set of similar contexts from this list is proposed and the user selects the appropriate one. If the current context is different from any entry of the list or all suggested contexts are wrong, then a new entry is made from the user by providing the context name. If the current context is found in the list, then no action to the list occurs. The context found or created is thus infused to the picture.

Due to the problem and the naming mechanism mentioned above, the similarity comparison when a picture should be opened, has changed. In this case there exist no longer similar contexts but only equivalent or not equivalent contexts. The received context is no longer compared directly with the current context. Firstly the known context list is looked up for it and if the list does not contain the received context, a new entry in the list is made for the received context. Afterwards the current context is checked with the list and upon finding the entry that corresponds exactly to the received context name, the found context is then checked with the current context. Decided by the check of current and found context in the list, the permission to open the picture is hence granted or denied.

This solution introduces a new problem. Every user has a different naming policy. For example I call my home *Heem* in my mother tongue but a user whose mother tongue is Spanish he would call his home *Casa*. Thus same contexts may become different ones.

One solution to this problem would be predefining context names to avoid such confusions. For this purpose five context names were predefined:

- Party
- Home
- Meeting
- Dinner
- Pool Session

This solution is not optimal but for now it should be enough do demonstrate the context generation and sharing of sensitive data. A better solution to this problem is describes in the future work section.

4.2.4 File encryption

In SnapChat the pictures could be copied form the phone before it was opened. The problem can can not be avoided so in PriviShare they are encrypted to make the illicit copy unreadable even if they get copied.

PriviShare uses RSA[3] public-key encryption. The public key is derived from a password and an arbitrarily selected 8-byte salt sequence. The selected salt sequence is hard coded and it is the same in every instance of this application. The pass phrase is a combination of the receivers user id and the title of the picture.

The combination is formed by twisting the user ID and the subject of the picture together. For example: the user ID is *"User1"* and the subject is *"M club"*, then the resulting pass phrase is *UM scelru1b*. To make it even harder to guess, special characters are inserted after each vowel.

A/a
After an A/a, "," is added.

E/e
After an E/e, "." is added.

[3] RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described the algorithm in 1977.

I/i
: After an I/i, "-" is added.

O/o
: After an O/o, "!" is added.

U/u
: After an U/u, "?" is added.

Thus the passphrase results in "*U?M sce.lru?1b*".

4.2.5 Container format

After the encryption the context information is bound to the picture. This is done by implementing a container format. The first idea was to develop a proper container format but after some research I concluded that a new container format should only be developed when no file format that fits all the need is found from the thousands of file formats that already exists.

Hence the ZIP[4] file format is used to list the pictures with its corresponding context. A positive side effect to this format is that the ZIP algorithm also compress the added files.

When taking a new picture, a temporary image file and an XML file that models the current context is created. The temporary files are directly deleted from the phone after the ZIP file has been created. When listing the received pictures, the content of the received files are extracted to a temporary directory which is deleted after the termination of the program or every time the listing method is called.

4.3 Threats to Validity

4.4 Conclusion

[4] `https://en.wikipedia.org/wiki/Zip_(file_format)`

Chapter 5

Conclusion

5.1 Future work

Bibliography

[1] Anind K. Dey and Gregory D. Abowd. Towards a better understanding of context and context-awareness. In *In HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer-Verlag, 1999.