DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Developing a Scalable, Secure, and Privacy-preserving Platform for Collection, Aggregation, and Analysis of Mobility Data

Dinh Le Khanh Duy

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Developing a Scalable, Secure, and Privacy-preserving Platform for Collection, Aggregation, and Analysis of Mobility Data

Entwicklung einer skalierbaren, sicheren, privatsphäre-erhaltenden Plattform zur Sammlung, Aggregation und Analyse von Mobilitätsdaten

Author:	Dinh Le Khanh Duy
Supervisor:	Prof. Dr.-Ing. Jörg Ott
Advisor:	Doan Trinh Viet
Submission Date:	15.03.2020

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.03.2020 Dinh Le Khanh Duy

Abstract

blablablbalblalbalbal

Contents

1. Introduction

1.1. Motivation

There are currently around 7.7 billion people living on earth with a declining growth rate at currently 1.1 percent per year. The UN is predicting that the population growth is going to be sinking steadily in the next decades, but despite that, the population is projected to increase until 2100. They estimate that the earth will hit approximately 10.9 billion at the end of the century, according to their diagram 1.1.

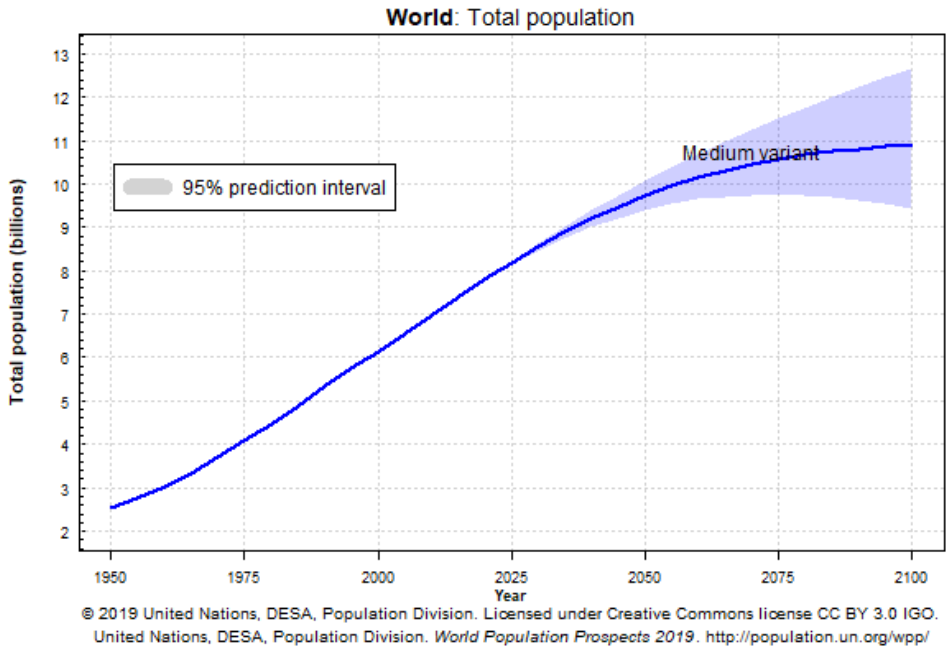


Figure 1.1.: Projections for population growth this century

Big cities and metropoles around the globe are inevitably growing denser year by year. According to the United Nations, urban areas around the globe have been experiencing an influx of people since the start of industrialization. While the same can be said for rural areas because of the general growth rate of the population, the increase in people moving into big cities is much higher compared to the countryside. Depending on how certain factors evolve, such as climate change, autonomous driving, and other technology and politics, it could decelerate or even accelerate growth.

To solve issues stemming from the increased demand in housing, traffic, and infrastruc-

ture, urban planning is one of the most important tools. Improving traffic control requires knowledge of the traffic flow and inefficiencies that are causing traffic jams. Knowing local population trends, movement patterns, and other factors could benefit the planning of housing and energy infrastructure. There are only a handful of companies that hold the monopoly on the much-required data, but acquiring them is expensive and can cause problems with data protection laws. Publicizing the information without anonymizing it, severely infringes on the privacy of the originator of the data. But the sole act of anonymizing the data is often not enough. The use of inference attacks may make it possible to associate data and re-identify the individual, as Sweeney has proven.

To solve some of the issues, we can leverage the widespread availability of portable computing power, and create a platform that provides the data on a need-to-know basis. Today, most smartphones are equipped with several sensors, including a GPS sensor, pedometer, and accelerometer, which can be used to collect a wide variety of mobility data. Many applications are already utilizing those sensors to implement location-based services and games; for instance, Google Maps, Uber, and Pokemon GO. But there has been a lot of controversy around products of that kind. "If you're not paying, you're the product" is a quote that is often cited around data-driven applications. It applies to a lot of free services offered in exchange for your data. Google Maps, for example, provides a free navigation service, while collecting your sensory data, such as GPS and accelerometer, staying informed about traffic information. Facebook provides a free social media platform for people to connect, in turn, using collected data to sell targeted advertising.

How they use your data is not in your control. So publicizing big anonymous data sets has its limits, and data collection itself is a hot topic in today's media.

Simon van Endern proposed a solution to return control over the data back to the users. He implemented a platform that used the power of crowdsourcing the data collection and aggregated the data on a central server on request.

1.2. Research Questions

When collecting data, it is crucial to find a balance between its usability and the privacy of the collecting party. It is also important to keep in mind, that there has to be a level of trust in both the data collector and the data analyst.

RQ1: What are the benefits and drawbacks in Simon van Endern's architecture?

We take a look at Simon van Endern's original idea and implemented architecture. We investigate both the advantages and disadvantages of his idea in regards to scalability, security, and privacy.

RQ2: What improvements are possible?

His work shows a minimal viable product on which we can expand further. We search for possible ways to improve on his architecture and explore possible aggregations.

RQ3: What information do the raw values for steps and activities reveal?

As Simon van Endern has shown, mean values expose less private information. We want to take a look into the raw data that makes up the mean values and analyze the privacy concerns regarding the distribution of the step values and activities values for the participants and if they can be used to link to other data.

RQ4: How much information does the aggregation of real location data expose?

Related work has shown that location data are very susceptible to leaking private information. We look into methods to disassociate the mobility data with the individual to provide anonymity while maintaining statistical relevance.

RQ5: Can we use the current state of P2P technology to remove intermediate third parties?

There have been a lot of advancements in direct communication frameworks, and we look into possible SDKs that can be leveraged to implement and enable a more decentralized or distributed architecture.

1.3. Contributions

Our research has the same structure as Simon van Endern's. He has already laid the foundation with his bottom-up approach of collecting mobility data using crowdsourcing and the aggregation of some basic data. Using his idea, we will into the aggregation of real location data and analyze them for privacy risks using a real test and a simulated one.

First, we review related work concerning risks and solutions to sensitive data and pertinent anonymization techniques. After that, we examine Simon van Endern's original idea and his final implementation and expand on his architecture. We propose our approach to solve the problems mentioned and increase the types of aggregation. Then, we explain the details of our implementation and the design decisions we made. In next chapter, we document our test setups and the deployment and evaluate our collected results for information leaks. Finally, we draw a conclusion on Simon van Endern's and our work and discuss further possible improvements and give instruction on how the project can be reproduced.

2. Related Work

2.1. Risks

Working with sensitive data brings out multiple attack vectors that have to be assessed before actions can be taken. We compile possible attack vectors and how they have been solved and how they could be solved.

2.1.1. Centralized

Because of efficiency, centralized information systems have always been a go-to infrastructure. The advantages of simple deployment, ease of maintenance, and less bureaucracy will always be an incentive for big companies and governments to consider it. In 1965, the US Social Science Research Council proposed a National Data Center to store all data in a central location for statistical data analysis. Still, in the end, the plans for the system were shot down because of the lack of privacy protection.

Today, several big tech companies are collecting information about their users and storing them in central databases. But centralization has a fatal flaw in protecting privacy. The collection of sensitive data in one location poses a high risk to their originator. Centralized databases that store private information are one of the weak points that have been under constant attack. In the first three quarters of 2019, there have been over 5000 data breaches, with almost 8 billion records exposed, 33% more compared to the number reported in the first nine months of the previous year. Around 10% of the breaches originated from the inside, from accidental leaks to intentional publications.

2.1.2. Reconstruction, Linking and Tracing

If we strip away all sensitive information, there remains a risk of private data leaking, which can be used by reconstruction, linking, and tracing attacks. The most severe problem is the danger of re-identification. This enables adversaries to deduce the identity of an individual using other publicly available data sets or auxiliary knowledge.

In her work, L. Sweeney was able to re-identify former Massachusetts governor William Weld by linking medical records from the Group Insurance Commission and the voter registration list.

The goal of reconstruction is to determine sensitive data from a data set that has been generalized or suppressed using publicly available information.

Tracing, on the other hand, is the ability to identify if an individual is present in a data set or not. This can expose information, which has previously been unknown.

2.1.3. Location Tracking

In regards to location data, the ability to infer the home and work location poses both risks for privacy and life and limb. Research has shown that it is possible to deduce the home address of an individual and even his workplace using historical location data. Location data is also able to hold more information than just spatial and temporal data. The association of a place with other sensitive information is possible. For example, the attendance of a political rally can tell an adversary about their political affiliation, or tracking the position of someone to a specialized clinic will expose private medical data. Besides, being able to analyze the movement pattern and predict the presence of people in a particular location may put their lives in danger. Predicting the presence of a person in their home alone or the presence might enable someone to break in and rob their house.

2.2. Countermeasures

To implement a secure platform that is not so vulnerable to the problems proposed in the sections above, we look into methods and design decisions to prevent them.

2.2.1. Distributed and Decentralized

The dangers of data breaches originate from outside the companies as well as inside, and their cause range from poorly implemented security and lax security policies to human error. Figure 2.1 depicts two more alternatives in place of a centralized architecture: distributed and decentralized.

When the internet was implemented as the Advanced Research Projects Agency Network (ARPANET) in 1969, it was a decentralized network of computers scattered across the United States. With the adoption of the TCP/IP in 1982, it became the internet, an interconnected network of networks. In 1989, Tim Berners-Lee introduced the world wide web as a read-only means of accessing information from other computers, and the commercialization turned it into the centralized web we know today.

In the last decade, we have seen a rise in attempts to reorganize the internet. This trend amassed attention in 2009, when the creator under the pseudonym Satoshi Nakamoto created Bitcoin, "A Peer-to-Peer Electronic Cash System" leveraging blockchain technology, followed by the Ethereum platform in 2013. The goal of decentralization is the separation of power from a single instance. In our case, it would be to take away control over our data from monopolies.

Distribution takes decentralization a step further by eliminating the central control. All instances have the same amount of power.

One of the protocols that arose from this niche is the InterPlanetary File System (IPFS). IPFS is a peer-to-peer hypermedia protocol to make the web more decentralized and distributed.

Using distributed storage, in place of a centralized database, removes the single point of vulnerability and lowers the effort-to-reward balance and thus might deter malicious actors

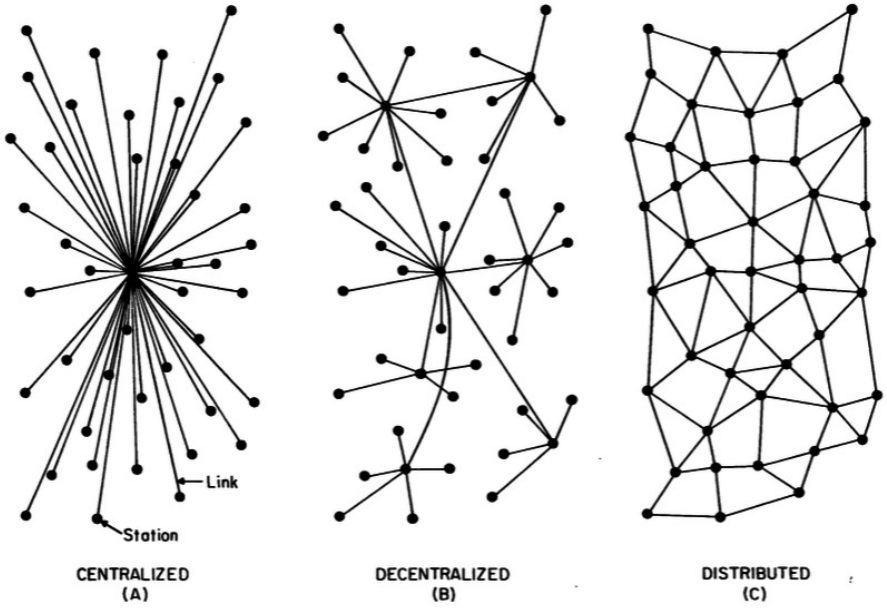


Figure 2.1.: Comparison between different types of centralization.

from trying to steal data.

2.2.2. Homomorphic encryption

Conventionally, encryption methods do not provide anonymity. But one could argue that in a way, anonymity is provided when the data is not readable or accessible. If an adversary manages to steal a data set while it is still encrypted, they wo not be able to infer identity or any additional information, unless they are able to decrypt it beforehand.

Homomorphic encryption enables the arithmetic operations on an encrypted data set. They are separated into three categories:

- **Fully Homomorphic Encryption (FHE)** allow multiple arbitrary operations, but have a lot of overhead and thus are expensive computationally.
- **Somewhat Homomorphic Encryption (SWHE)** support only selected operations to a limited number of times and are computationally more feasible.
- **Partially Homomorphic Encryption (PHE)** enables one type of operation any number of times.

This protects the data set from intermediate parties, making them unable to derive private data, while making it possible to work on it.

2.2.3. k-Anonymity

The above sections have shown that if a data set seems anonymous by itself, quasi-identifiers, such as ZIP code, birth date, or sex, still enable malicious actors to link individuals to their data. One of the countermeasures to this problem is providing k-anonymity. This is achieved when every query for quasi-identifier returns at least k results. A quasi-identifier is an identifier or a combination of non-identifying attributes, that can be used to link with external data sources to create new identifiers. For this, P. Samarati and L. Sweeney suggest the use of generalization and suppression.

Former is realized by expanding specific attributes into ranges. For instance, instead of assigning the real age, an age range is used. This results in a loss of accuracy, but a higher degree of anonymity and thus, privacy. For the latter, there are two methods of suppression. Attribute suppression removes attributes from the data set, reducing the number of possible quasi-identifiers by lowering the number of possible combinations. Record suppression, on the other hand, deletes entire entries in the data set to take out unique entries that do not meet the criteria of k-anonymity.

2.2.4. Differential privacy

Differential privacy can be used to prevent statistical databases from leaking private information. It is a compelling mathematical definition that is able to guarantee privacy. An algorithm is differential private as long as it disables tracing attacks, meaning that output does not show signs of a particular individual is present in it or not. Both Google and Apple have implemented differential privacy into their data collection methods.

Differential privacy guarantees three qualities:

- All data about an individual in the database is protected even if the adversary manages to learn auxiliary information about all other entries in the data set.
- Two differently differential private data sets can be combined, and the resulting data set is still differential private.
- It is possible to quantify the loss of privacy and information gained by the malicious actor.
- It can be applied to groups.

To achieve differential privacy, one basically adds noise to the data, but replay attacks can reveal actual values. To avoid this, the default noise function used is the Laplacian distribution mechanism by adding Laplacian noise. The operation can be used either on the aggregation of the user data or on the aggregated data itself.

Global Differential Privacy

If the noise function is applied to the entire aggregated raw data set, it is called global differential privacy. This works by using a noise function on the result of a query, as depicted

in figure 2.2. The advantage of this model is that the resulting data set is still very accurate, but on the downside, the whole raw data set has to be available to use this function.

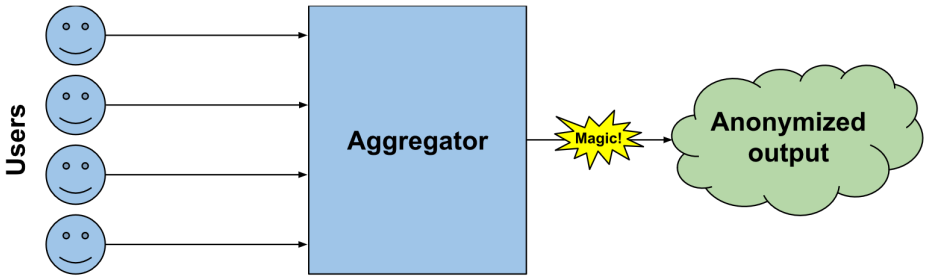


Figure 2.2.: Illustration of a global differential data flow

Local Differential Privacy

Figure 2.3 shows that differential privacy can also be used on the entries of the data. By running every entry through the mechanism, it creates a locally differential private data set. The disadvantage of this method is the much higher noise, but in exchange, the raw data itself is protected and does not require trust.

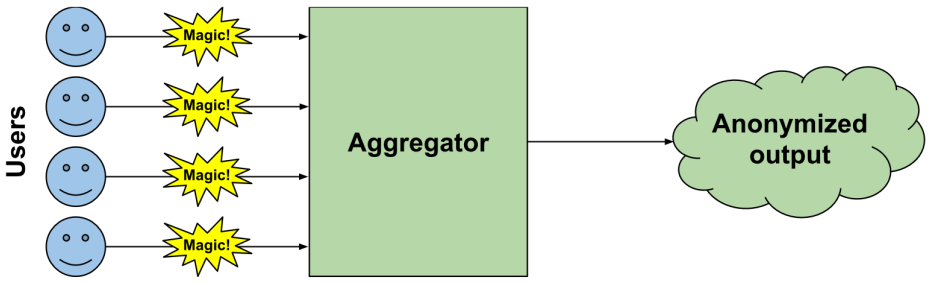


Figure 2.3.: Illustration of a local differential data flow

A typical example of this is the coin toss. A survey asks a question that can be answered with a yes or no. The function would toss a coin and would answer the question honestly

on heads and otherwise answer the question with a second coin toss, with heads resulting in yes and tails in no.

2.2.5. Spatial Cloaking

As mentioned above, spatial data can reveal a lot of sensitive information about a person. So location privacy should always be a high priority. To achieve that, there are a lot of algorithms that are used to hide the real location in a general area.

Casper

Casper is a grid-based cloaking mechanism, that organizes the location in squares. The level of detail is ordered like a pyramid. The coordinates are cloaked by using the lower-left corner and upper-right corners as a designated area of the location. The figure 2.4 would show the lowest level of detail with $\langle(0,0),(4,4)\rangle$, covering the whole area, and the highest level of detail with $\langle(0,2),(1,3)\rangle$, only wrapping a small portion of the area. To fulfill the requirements of k-anonymity, the algorithm looks for neighboring areas for other users and then spans the whole area when it finds one. For example, U_1 with its cloaked area $\langle(0,2),(1,3)\rangle$ would need the area $\langle(1,2),(2,3)\rangle$ to satisfy $k = 4$.

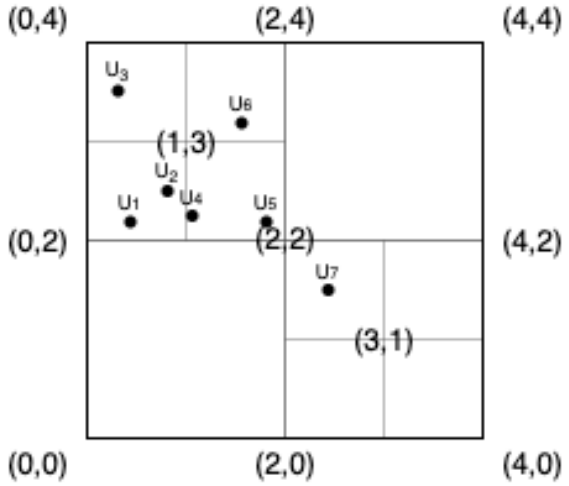


Figure 2.4.: Illustration of spatial squared areas

Interval Cloaking

Similarly to Casper, Interval Cloaking also uses rectangular cloaking areas to hide the real locations. But instead of using neighboring squares, it uses the lower level of detail to achieve k-anonymity. Using the example from above, Interval Cloaking would span the anonymized spatial area over $\langle(0,2),(2,4)\rangle$ to assure k-anonymity with $k = 4$ for U_1.

Hilbert Cloaking

Another famous spatial cloaking algorithm is Hilbert Cloaking. This uses the Hilbert space-filling curve to map the 2-dimensional area into a 1-dimensional representation. Then it groups together points depending on the provided k because points that are in proximity on the Hilbert curve are also close to each other in 2D. Instead of using predetermined rasterization of the area, the cloaked area is spanned by the users themselves.

3. Design

3.1. Trust, Usability and Privacy

In any platform, we expect a certain level of trust from each participant, may it be from the service provider or its consumer. How this trust is created can have many different sources. But the main reason a person or organization can be trusted is because of accountability. Companies with a strong market presence have the public's trust since they have been present for a longer period of time and ca not just disappear overnight Consequently, they can be held accountable for their actions. Today, Microsoft, Google, Amazon, and Facebook, for instance, are seen as trustworthy to a certain degree. But this trust is not invulnerable. Lies and secrecy can damage trust. Keeping the public uninformed about severe data breaches or selling sensitive data to third parties without their knowledge and permission has strained the public trust in recent years.

Even if the trust is in place, we as consumers require privacy. If we can guarantee anonymity, however, we can also guarantee privacy, as re-identification should be impossible at that point.

As chapter 2 has shown, it is a hard task to anonymize big data sets, as quasi-identifiers can be used to infer new information using linking attacks. So one way to achieve anonymity would be to strip away all attributes that could be used in another data set.

Here we hit a wall with the usability of the data itself. Cynthia Dwork says "de-identified data isn't," meaning that either that the data itself is not de-identified because of possible reconstruction of the data is not data anymore because it is not useful for analysis purposes. We see data itself is only as useful as its relationships.

So there is a need to find a way to find a balance between trust, usability, and privacy. Due to the limited scope of the thesis and trust being a vast research subject, we focus on the privacy of the crowd and the usability of their collected data. We assume that all participating devices and servers can be trusted, and possible adversaries only have access to the published end results.

3.2. Approaches

As mentioned before, there are advantages and disadvantages to Simon van Endern's system, and we explore how we can implement improvements and if they would be feasible with the current state of the art or if another alternative has to be chosen.

3.2.1. Simon van Endern

The proposed architecture implemented in his thesis was using a centralized approach with a distributed storage solution. The model can be seen in figure 3.1.

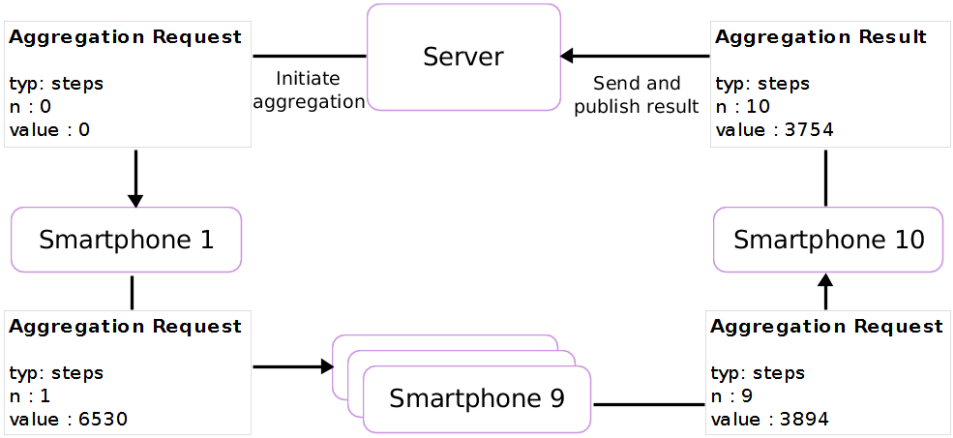


Figure 3.1.: Simon van Endern's original architecture.

In his thesis, he eliminates the need for a central database, by collecting and storing raw data directly on the users' smartphone, creating a distributed database. This gives each participant a lot of control over their own data, just by deleting the application they are able to opt-out of future data analyses.

In his original idea, the mobile phones would use peer-to-peer technology to forward the aggregation requests and finally send the data to the server for publication. But because of the lack of available technology, he opted to use the central server as an intermediary to send from mobile phone to mobile phone. To keep the data confidential and ensure anonymity from the server, he uses RSA and AES encryption. To forward the data to the next device, he implemented a polling solution. The application periodically asks for a new aggregation request targeted at the device from the server. If one is present, it fetches the request over the REST API, and after adding its own data, it posts it back to the server targeting the next device. In the finite scope, he managed to implement three aggregation types:

- Average number of steps over the course of a day for participants.
- Average time spent on an activity (walking, running, in a vehicle or on a bicycle).
- Average number of steps of a participant during the test period.

We examined his work and discovered a few flaws that we might be able to improve on. We will look into possible solutions to make the architecture more efficient, scalable, secure, and privacy-preserving.

3.2.2. Expanded Approach

Simon van Endern's original idea, as is, can hardly be scaled because of the nature of its forwarding chain. Adding n new devices creates a linear time complexity of $\mathcal{O}(n)$ and space complexity of $\mathcal{O}(n^2)$. For every device added, the aggregation takes up to an additional 15 minutes, and the data fetched and sent is the data of all previous devices combined, making it quite a burden on participant's data plan depending on their position in the chain.

Our first goal is to make the platform more efficient and scalable. We propose to parallelize the chains by building multiple groups for the aggregation, as shown in figure 3.2. Each chain in itself resembles the Simon van Endern's original idea in figure 3.1. With m groups, adding n devices, it only creates a time complexity of $\mathcal{O}(\frac{n}{m})$ and space complexity of $\mathcal{O}((\frac{n}{m})^2)$. So choosing the right m could change linear and quadratic growth to potentially constant growth.

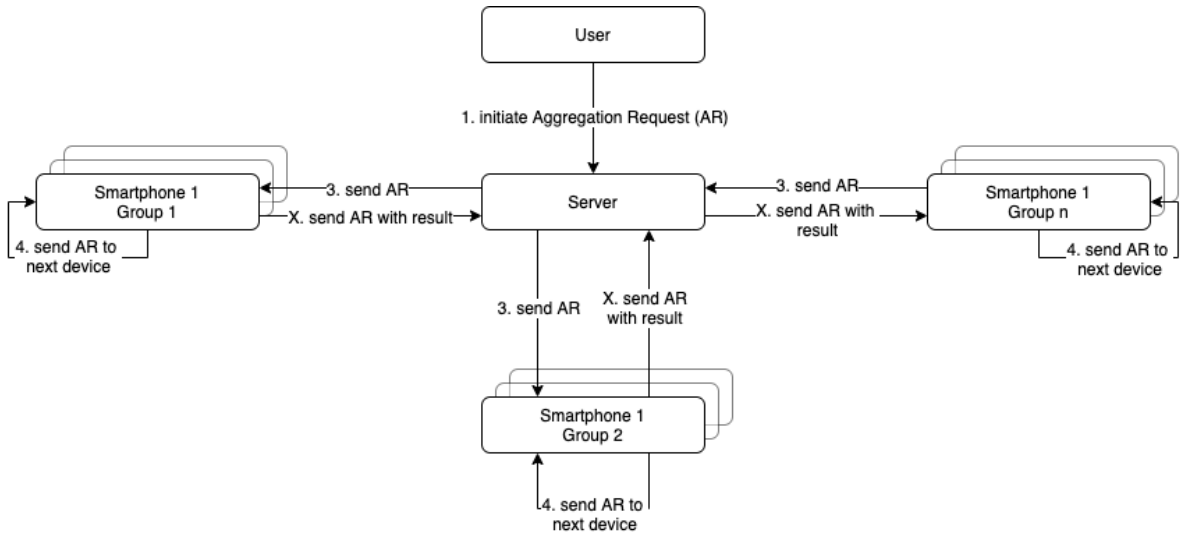


Figure 3.2.: Distribution of still raw values.

Another flaw to Simon van Endern's architecture is the constant polling for new aggregation requests. Each user asks for a request targeted at their device and will do so every 15 minutes. Depending on the number of participants in the project, this could mean up to millions of messages to the server every few minutes. To avoid this, we explore some peer-to-peer frameworks that can send data directly to the next target device to cut down on forwarding delays and remove intermediate third parties.

As we assume that there might possible privacy leaks, we also consider the inclusion of local differential privacy to protect the collected data. We look into the benefits and the disadvantages that the implementation of this mechanism could bring.

Beside architectural design changes, we also want to collect more data, especially real location data. In order to extend the aggregation options, we add the two following two types:

- Location of all participants in an interested area at a specific time
- Number of participants that were in an interested area in a specific time period

4. Implementation

The following sections describe the implementation and explain the design decisions we made during development.

4.1. Technology Stack

As our mobile platform, we have to decide between iOS and Android. While iOS has a more up-to-date operating system with 70% of all of their mobile phones running iOS 13 and 23% running iOS 12, Android has a much bigger market share with around 74% compared to Apple's 25%.

In terms of available development kits and libraries, both platforms provide plenty of choices. In the end, we choose Android, mainly because of their ease of deployment on devices for test scenarios. As for programming language, we opt for Kotlin instead of Java because of their intuitive syntax and compact and more readable codebase.

On the server-side of development, we select Node.js using Typescript combined with MongoDB's cloud storage solution, Atlas. Node.js provides a lot of flexibility with its vast ecosystem of third-party packages, and MongoDB, as a NoSQL database, easily stores and combines any type of data, allowing quick changes in the data model.

Our aggregation results can be fetched over the REST API or viewed directly in the MongoDB Atlas interface, given access privileges.

4.2. REST API

We use a REST API based on the JSON data exchange format, and the endpoints are shown in the table 4.1.

The endpoints require authentication either as a researcher or as a data collecting user. For the researcher, we currently only create admin access by manually entering the username and password into the database. To register as users as participants in the crowd, they have to send their id and their public key, as depicted in figure [X]. The server provides the participants with a random password for future authentication for sending the end result of their group.

4.3. Aggregation API

To start an aggregation, listing 4.1 shows the researchers have to send their *username* and *password* for authentication, the *requestType* they want to aggregate and additional search

HTTP call	Description
POST /crowd HTTP/1.1	Creates a new user and returns a password
POST /crowd/ping HTTP/1.1	Updates the latest timestamp to current time
POST /aggregationRequest HTTP/1.1	Initiates a aggregation request and forwards them to the device groups
POST /aggregationsteps HTTP/1.1	Sends the intermediate step results of a group to the server for further processing
POST /aggregationactivity HTTP/1.1	Sends the intermediate activities results of a group to the server for further processing
POST /aggregationlocation HTTP/1.1	Sends the intermediate loaction results of a group to the server for further processing
POST /aggregationpresence HTTP/1.1	Sends the intermediate presence results of a group to the server for further processing
GET /aggregationResult HTTP/1.1	Retrieves a the results of an aggregation request

Table 4.1.: Description of the API endpoints implemented in the server.

options as *request* depending on *requestType* to the server. The search options are visualized in the listings 4.2, 4.3, 4.4 and 4.5:

- *start* and *end* or *date*: time of interest
- *lat* and *lon*: coordinates of the point of interest
- *radius*: distance to the point of interest
- *type*: type of activity as encoded in the Google Activity API
- *accuracy*: position after the comma to round for GPS accuracy
- *anonymity*: number of k-anonymity for participants

```
{
    "username": "admin",
    "password": password,
    "requestType": "steps" or "activities" or "location" or "presence",
    "request": see listing 4., 4., 4. or 4.
}
```

Listing 4.1: Initial aggregation request from researcher

```
{
    "date": 1582945200000,
    "lat": 35.535751,
    "lon": 139.629355,
    "radius": 50000
}
```

Listing 4.2: Search options for steps

```
{
    "type": 0,
    "start": 1582941600000,
    "end": 1582945200000,
    "lat": 35.535751,
    "lon": 139.629355,
    "radius": 50000
}
```

Listing 4.3: Search options for activities

```
{
    "date": 1582945200000,
    "accuracy": 0,
    "anonymity": 2,
    "lat": 35.535751,
    "lon": 139.629355,
    "radius": 50000
}
```

Listing 4.4: Search options for locations

```
{
    "start": 1582941600000,
    "end": 1582945200000,
    "lat": 35.535751,
    "lon": 139.629355,
    "radius": 50000
}
```

Listing 4.5: Search options for presence

When the server receives an aggregation request of the described form, it will send out a more detailed request to the device groups for data collection. The messages sent to the first mobile phones can be seen in the listing 4.6.

```
{
  "to": "35e78c9a6072b5e81ac2a5",
  "data": {
    "encryptionKey": null,
    "iv": null,
            "requestHeader": {
      "id": "618k1ndk71bvw1h",
                "start": 1582941600000,
                "end": 1582945200000,
                "type": "steps" or "activities" or "location" or "presence"
            },
            "requestOptions": {
                "groupNumber": 0,
                "numberOfGroups": 1,
                "from": "token",
                "group": []
            },
            "requestData": see listing 4., 4., 4. or 4.,
            "data": {
                "n": 0,
                "raw": []
    }
  }
}
```

Listing 4.6: First aggregation request sent to the groups

After receiving and adding their data, the phones encrypt the *data* JSON object and forward a similar message as figure [X] to the next device in the list. After reaching the last, it sends the results back to the server. The messages to the server contain the password generated on registration for authentication.

The implementation of the different JSON formats are flexible and can accommodate more aggregation types. The next sections will describe the features and the process of aggregation more in-depth from the Android device's and the server's point of view.

4.4. Android Application

As our target version of Android, we choose Android 10 (API level 29), which is currently the latest version of Android, and as a minimum version, we select Android KitKat (API level 19). The latest report shows that around 98% are on Android KitKat or later. To collect mobility data, we leverage the power of Google Play Services. For persistent storage, we make use of the Room Persistence Library, as it provides an abstraction layer over SQLite and is commonly used in a lot of Android applications.

Because the platform relies on the generation of data from the crowd, the main functions of the application are the collection of mobility data and providing the data on request. Figure 4.1 describes the android applications with three main packages:

- *User Interfaces*: This package handles the start of the application and the presentation of the stored data.
- *Background Services*: This starts all the processes in the background and manages all data collection and communications.
- *Data Storage*: This part saves and fetches all the collected data stored in the Room database.

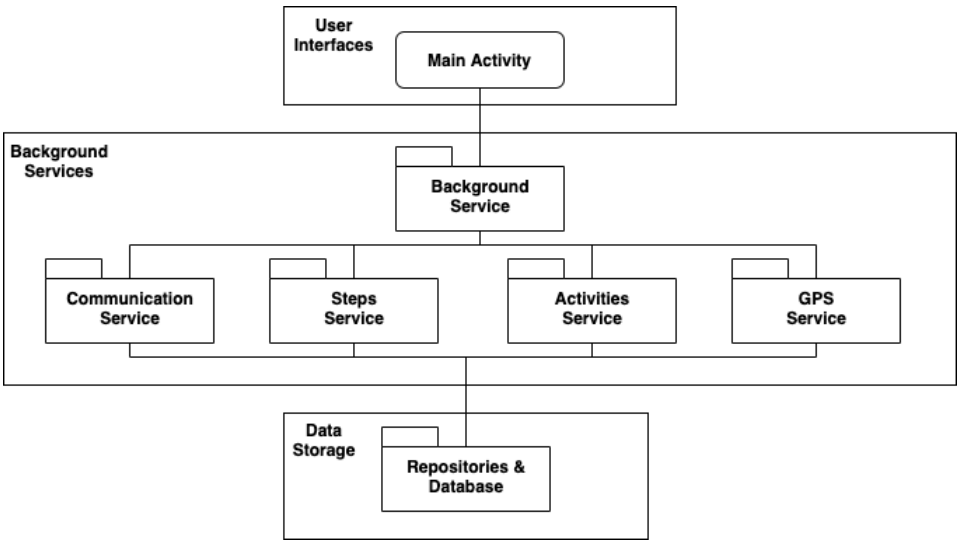


Figure 4.1.: Distribution of still raw values.

On starting up, the application opens into the *Main Activity*, where it asks the users for permission to access location services. On acceptance, the application starts the background services for data collection explained in the next section. It also serves the purpose of displaying stored data. A screen for each type is implemented: steps, activities, and GPS. Using Simon van Endern's Android application as a blueprint, we implement the same features to keep our application running behind the scenes. Because of background limitations Android

introduced with Android Oreo, we create a non-dismissible notification that is displayed in the status bar and the notification center. This turns the application into a foreground application, bypassing the limitations set in the newer Android versions. To keep user interaction to a minimum, we enable the application to reopen when it crashes or is closed, and when it is rebooted.

4.4.1. Data Collection

After the users successfully accept all permissions, the *Main Activity*, starts the *Background Service*. The *Background Service*, in turn, starts four tasks that have been shown in figure 4.1. The four modules are loosely coupled and have been separated by data type or task at hand. We split the application into data collection and data aggregation. The aggregation process will be further explained in the next section. The data collection has three modules:

- *Steps Service*: This module handles two classes, the step service, and the step logger. If the mobile phone has a pedometer, it registers the sensor to update the step count. The step logger receives steps from the sensor and stores it in the *steps_table* ever minute. Every step entry has an *start* and *end* timestamp and the number of steps taken in that time frame as *steps*.

- *Activities Service*: This service, similar to the *Steps Service*, is composed of the activities service and the activities logger classes. The activities service leverages Google's Activity Recognition API to identify the current activity of the mobile device. We are only interested in the main activities: still, walking, running, on a bicycle or in a vehicle. Therefore, for all possible activities, we register the transition type of entering or exiting the activity. Whenever we switch activities, the activities logger writes the *timestamp*, the *type* and wether we *enter*ed or not into the *activities_table*. Additionally, we store the exited activity into the *activitiesDetailed_table* with the *start* and *end* timestamp and the *type*.

- *GPS Service*: This part of the application has the same structure as the others described above. The GPS service accesses Google's Fused Location Provider API to collect location data. The API itself uses the GPS sensor as well as the network sensors in the device to determine device location. We make the location sample rate dependent on the current activity because the positional changes between *still* and *on a bicycle*, for example, are very different. So we set the interval to 5 minutes for *still*, 30 seconds for *walking* and 15, 5 and once per second for *running*, *on a bicycle* and *in a vehicle* respectively. Every once in a while, the GPS logger receives batches of data from the API and saves them into the *gps_table* with their *timestamp*, *lat*itude and *lon*gtitude.

4.4.2. Aggregation

The last part of the background services is the *Communication Service*. This is in charge of passing on data from the server or devices to the next target. For this interface, we have several ideas in mind.

IPFS

To take out the central server as an intermediary as implemented in Simon van Endern's version, we searched for libraries and SDKs that could be viable for our platform. We found the open-source projects, IPFS and libp2p, which are trying to be the foundation of the decentralized and distributed web. IPFS uses hashes to create content identifiers and turns them into blocks in a directed acyclic graph. To discover the peers with the content it uses a distributed hash table, using the modular P2P networking stack libp2p to communicate between nodes.

Textile leverages the strength of both technologies. Their main feature threads, a hash-chain of blocks that can represent any type of dataset, which we could use to send data directly from device to device. Unfortunately, after further investigation, the Textile SDK is not viable in its current state because it is unable to keep the node in Android alive after going into the background or turning off the device screen. It is essential for data collection and aggregation to keep communication channels available throughout the whole lifetime of the application.

Thus we opted to instead use a third-party push notification service as an intermediary to forward the messages between the server and other devices.

Push

To replace Simon van Ender's polling with an event-based action architecture, figure 3.2 shows our new design. For our push service, we select Pushy because of their free entry barrier and independence from big tech companies, as well as having implementations for both iOS and Android. We could also have used Firebase Cloud Messaging to forward messages. Of course, the modular architecture of the app enables us to swap this method of communication for a P2P solution as soon as one proves applicable.

The complete communication infrastructure is handled by the communication service, communication receiver, and communication handler. After starting the application, the *Background Service* creates the communication service that registers the device to the push service and receives a token as identification. Then it generates an asymmetric RSA key pair and registers itself to our platform with the Pushy token as *id* and the *publicKey* over the REST API. As a reply, the mobile phone receives a random password for future authentication purposes mentioned previously.

The communication receiver works as the entry point for push notifications from the push service and adds relevant data to the message. Upon receiving an aggregation request like in figure [X], it checks for encryption and, if required, decrypts the data field depicted in figure [X]. The application then aggregates data according to the *type* sent in the *requestHeader*:

- **steps**: We add up the number of steps from the start and end of the specified day and add it to the list of raw values.
- **activities**: In this case, we sum up the time spent on the specified activity in the defined time frame and add that to the list of raw values.

- **location**: Here, we look for locations with the closest timestamp to the date specified in the header, but also inside a reasonable time, i.e., 10 minutes to cover the fact that still activities only logs the GPS data every 5 minutes. Then we spatially cloak the GPS coordinates and add it to the list of raw values as the hidden GPS position.
- **presence**: We just check all the GPS coordinates in the specified time frame, if they have been inside the range of the point of interest and add a 1 for if it has been and a 0 it has not.

After that, the communication handler takes the message and prepares it for the next participant in the *group* array encoded in *requestOptions*. If the device was the last in the list, it sends the results back to the server; otherwise, it uses the hybrid encryption scheme to encrypt data with the credentials of the subsequent device. So the current device generates a symmetric AES key and encrypts the *data* and afterward does the same to the symmetric key using the public RSA key of the selected participant. Next, the communication handler forwards it over the push service, which in turn sends a push notification to the specified mobile phone. This repeats in each group until the last device has been chosen.

Bypassing inactive users in the aggregation chain

In the case, that the aggregation is stuck because of any reason, for instance, a device does not have an internet connection, it is turned off, or the participant deleted the application, we have to be able to bypass the inactive mobile phone and select a new one.

With our selected push notification service, the mobile phones periodically ping the push notification server, signaling that they are online and active. Using this information, we only send aggregation requests to the participants that are marked online by the service, bypassing devices that have been offline and have not contacted it in a while. But it could also be the case that the mobile phone is not available after the aggregation has already started.

So after the users forward their encrypted results to the following device, they start a sleeping thread that will skip the next device and select the one after that. The subsequent user that receives the message from the push service will aggregate the data as usual but will send a short confirmation message back to the preceding participant, which cancels the sleeping thread, signaling that aggregation was successful. The device does not need to be skipped.

By avoiding offline devices and actively confirming that a request has been fulfilled, we are able to bypass inactive users and make the aggregation more efficient and reliable.

Spatial Cloaking

To be able to provide k-anonymity for location data, we have to be able to generalize the GPS coordinates. By using spatial cloaking algorithms, we can hide the true position and hide the whereabouts by assigning the user to a rough area. We consider using Casper, Interval Cloaking, or Hilbert Cloaking. We select a variant of the Interval Cloaking algorithm

by rounding down and rounding up the digit after the comma defined in *accuracy* in the *requestData* to create a cloaked rectangular area. To conserve data, we calculate the midpoint between the left lower corner and right upper corner of the area as a representative. It is important to find a good value for *accuracy* because a too high value would result in a high suppression rate and too low value in too broad data.

Differential Privacy

We decide against the use of differential privacy for our use cases because of the main flaws that it provides. Our decision applies to both global and local differential privacy. On the one hand, global differential privacy provides accurate data representation, but the algorithm needs the real values of the data infringing on our privacy-preserving requirement. Local differential privacy, on the other hand, can provide privacy without the intermediate data collector because of its composition feature. A disadvantage that local differential privacy brings is the necessity for a large data set. Since the noise addition to every entry in the set creates a much higher total noise level than global differential privacy and a huge number of participants is needed to cancel that out.

Differential privacy heavily depends on the sensitivity of the data. That is the maximum value a single entry in the data set can change the query. For example, a query for the count of true and false answers; the sensitivity is one as the removal of one participant only removes one response of true or false. But the higher the sensitivity, the greater the noise that has to be added to the data to guarantee privacy, which in subsequently makes the data less accurate.

We considered it for location data and presence data, but rejected the idea, on the basis that the differential private algorithms can currently only applied to simple data schemes, such as sums and counts. This would work for on our previous data, such as the average number of steps and time spent on an activity. Still, because of the high sensitivity of those values, it would lower the quality of our collected data too much, and we were not able to find a large number of participants.

4.5. Server

For our server, we are using Node.js, on version 12.12.21 with the express middleware, a minimal web application framework. Our MongoDB Atlas database instance receives data over the database communication module. The architecture can be seen in figure 4.2.

As our programming language, we select Typescript because of its versatility. As a superset of Javascript, it supports all Javascript libraries natively and allows for object-oriented programming paradigms. Also, it provides optional static typing and better code structure. But to run the server, we transpile our Typescript code to Javascript code to run the webserver.

When we start the server, *app.ts* registers the routes described in the table [X] and connects the database object to the cloud storage instance. All calls to MongoDB are handled over

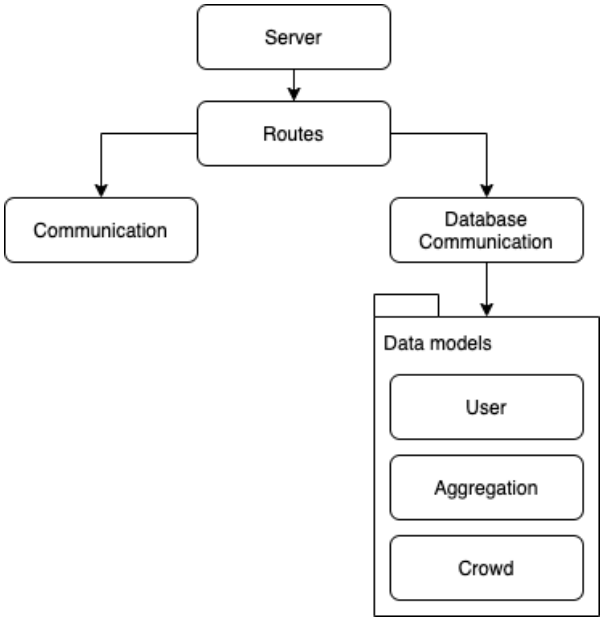


Figure 4.2.: Distribution of still raw values.

that object, while the route handler manages calls to the endpoints.

4.5.1. Registration

Upon starting the application, it registers to the server using the JSON shown in figure [X] over the */crowd* route. To update the latest timestamp, we also have an obsolete route */crowd/ping*, which has been replaced with the ping our push notification service already provides. But this route can be modified to ping the devices for their current status if another communication model is adapted.

4.5.2. Aggregation

We create one route to which the researchers can send their aggregation requests. After receiving a JSON in the mentioned form in the listing 4.1 in route */aggregationRequest*, the server will get all the devices, ping them using the push service, select all the available participants and calculate the groups. Before assigning the groups, we use the Fisher-Yates shuffle to mix the order of the users. We calculate group sizes depending on a designated minimal length, so the aggregation has enough members in each group for anonymity purposes, but small enough to stay efficient. After all that, the server sends the aggregation request to all groups using the push service.

On request creation, the server creates a temporary aggregation object that stores the id, number of groups, and how many it already has received. We have four additional routes for the final aggregation, */aggregationsteps*, */aggregationactivity*, */aggregationlocation* and

/aggregationpresence. They each manage the POST requests from the Android devices and calculate data or ensure anonymity.

4.5.3. Anonymity

To achieve privacy with data as sensitive as whereabouts, we apply k-anonymity. For steps and activity, we do not see any immediate privacy issues and thus refrain from using k-anonymity as that would needlessly sacrifice the usability of the data without benefits. As opposed to location data, we merge the received spatially cloaked raw values and suppress all coordinates that do not fulfill our defined k.

4.6. Limitations

Even with the proposed solutions, the architecture is still far from perfect. While the data should be confidential because of the state-of-the-art encryption, we still have to rely on a third party to deliver messages, making us dependent on their availability.

With Simon van Endern's aggregation chain, the most vulnerable participants would be the first because next in line would always be able to see the raw data they added. Unfortunately, homomorphic encryption is still computationally expensive, and there are currently no available libraries for Android to implement it.

Another issue that can lead to privacy issues is centralization. While the raw data itself is distributed, the control over aggregated data, its storage, and its collection is still under one central authority.

5. Performance and Evaluation

5.1. Field Test

To deploy our project in the field, we used IBM Cloud's free hosting service to run our node.js server and connect it to our MongoDB, as mentioned earlier Atlas instance. We tested the system over the course of eight days from Wednesday, the 29.01.20 to Wednesday, the 05.02.20. To find participants, we asked friends, acquaintances. We tried to recruit people in the vicinity and over private social media channels. Still, unfortunately, because of privacy concerns, we only found 13 volunteers to participate in the trial, of which only 5 to 7 devices were available over the whole test period. Because of Android's own battery management system and other OEM's aggressive battery saving algorithms, a lot of devices were not reachable after they have been unused for a longer period of time, entering Doze mode. Additionally, most of our users were fragmented globally, and with the low number of active participating devices, we had to adapt our aggregation parameters. To start aggregation processes, we use Postman, a tool for sending among others REST requests directly to an address.

We started with aggregating similar data to Simon van Endern, such as steps data and activities data, and afterward, we looked into location data and presence data. The collected data set can be found in the Git repository [X] and partially in the sections below.

5.1.1. Data Consumption

Running the application by itself should not require a lot of data, as the only data consumption only comes from the initial registration request to the server, the periodical ping to the push service, and the aggregation requests. We let the application run for the first few days and aggregated data irregularly starting after the 29.01.20. But most of the requests were sent towards the end of the field test, on the 05.02.20 and the 06.02.20. In the figure 5.1, the distribution of the aggregation can be seen. We were able to request proof of data consumption from 4 participants, but are only able to confirm that some of them were actively providing data in requests. The provided screenshots can be viewed in the Appendix, and consumption is shown in table [X].

Date	Location	Users

Table 5.1.: Location values collected over the test period each day at 12 pm JST

We can see in all cases that data consumption did not exceed 10MB and, in most cases, did not even use more than 5MB. We could extrapolate that the consumption of data should

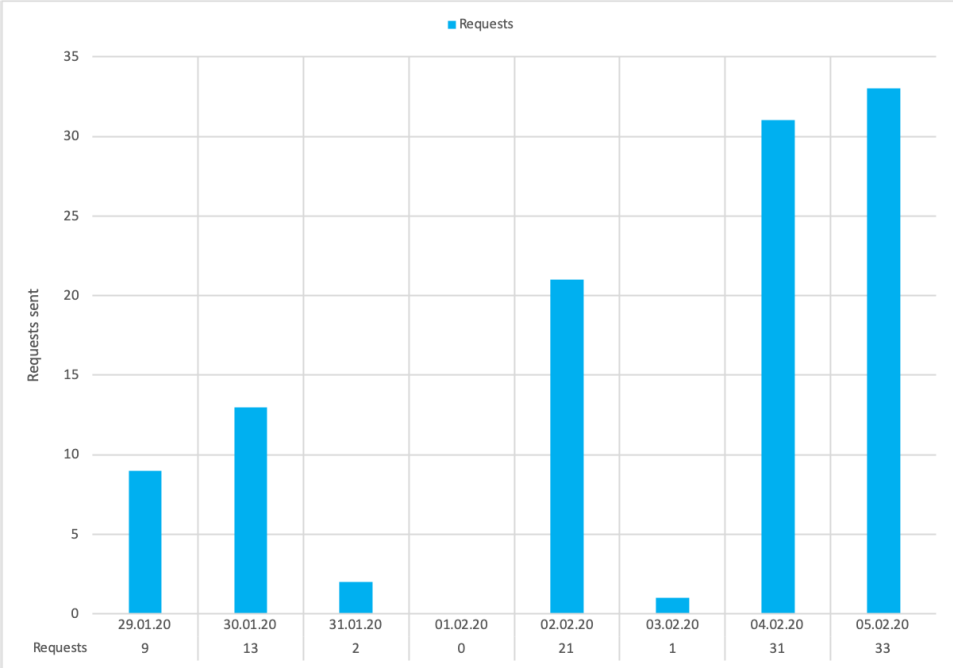


Figure 5.1.: Simon van Endern's original architecture.

not be a lot more than twice, depending on the number of aggregation requests sent because the parallelization of the data collection should always aim for small groups.

5.1.2. Results

We sent more than a hundred aggregation requests during the field test. We collected the average number of steps taken and the average time spent on the activities still, walking, biking, and in a vehicle. The start and end date parameters for aggregating these types were to 12:00 am JST for each day. We also requested for location and presence of the devices over the globe at 12:00 pm CET and 12:00 pm JST, 11:00 am, and 3:00 am in the GMT zone, respectively. Each aggregation had consistently from 3 to 7 participants.

We recorded the average time spent motionless was from 577 to around 1106 minutes. For individual participants, we have a wide gap with a minimum of 0 and a maximum of 1310 minutes, almost 22 hours, spent still, as depicted in figure 5.2. This can be explained on the basis that for counting activities, we only use finished activities that have a start and an end timestamp. For the two cases, it was highly probable that the phone has not been used touched for two full days, and thus the start date of the activity was before the 31.01.20 and the end after the 01.02.20. For the low values of 16 minutes and 28 minutes, we have not found an explanation.

As for walking, figure **??** shows a mean time between 25 and 75 minutes, from participants only walking 2 minutes up to 158 minutes. Without having the running time at hand, it is possible to infer that with the high number of steps recorded on half of the days, that one

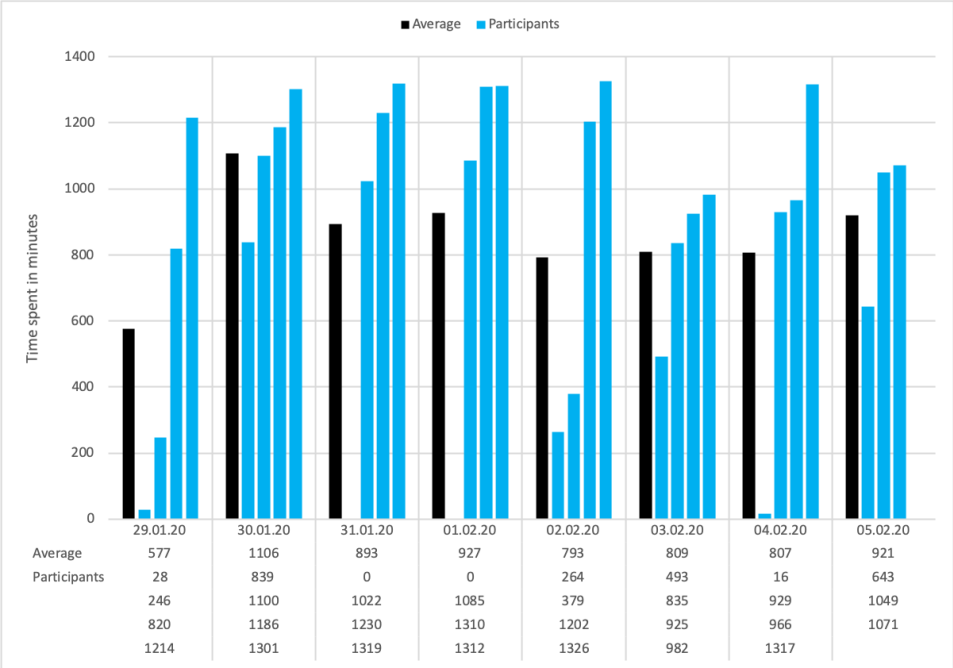


Figure 5.2.: Distribution of still raw values.

person is running regularly if the steps can be assigned to the same user.

On average, the participant uses a vehicle for up to 106 minutes. On occasion, many users do not use any transportation at all. Figure **??** shows the data a user spent in a vehicle. Registered activities from 1 to 2 minutes are with high probability credited to escalators and elevators.

For cycling, on some days, we only had one value of around 7 minutes. We can assume with confidence that it is one specific participant that is using the bike on occasion.

Now we take a look at the location and presence data. In the tables 5.3 and 5.2, we can see the GPS coordinates that we get after the server suppresses them for k-anonymity. We do this because there is a lot of information if only one person in a greater area is present. If we collect only spatially cloaked area in the vicinity and we do this for several time points, we can assume with high confidence, the general trajectory of that user posing a risk to privacy.

According to the aggregated data, we have 2 to 7 participants throughout the field test. Table 5.2 shows the location at 12 pm JST and table 5.3 at 12 pm CET, which corresponds to 4 am CET and 8 pm JST in the other time zone. As already mentioned before, because of the fragmented participant pool, we set the desired location at any GPS coordinates. Still, the range to 50,000km and the accuracy to 0, which results in gives us a rough estimate in a 111km range.

At 12 pm JST or 4 am CET, we have only two devices located in the greater Munich area with a midpoint of $(48.50108260577674, 11.495067909974292)$ which corresponds to the territory covered by $\langle(48, 11), (49, 12)\rangle$. On the 30th and 31st of January, we detected three devices in the same region and, additionally, 2 participants close to $(35.50103138028429, 139.49688751384306)$

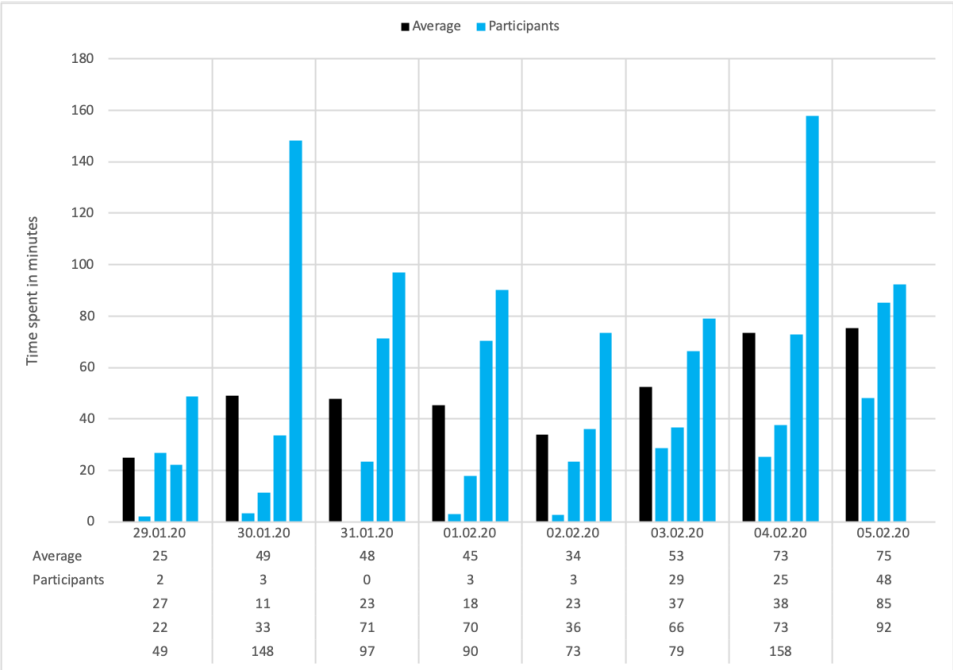


Figure 5.3.: Distribution of walking raw values.

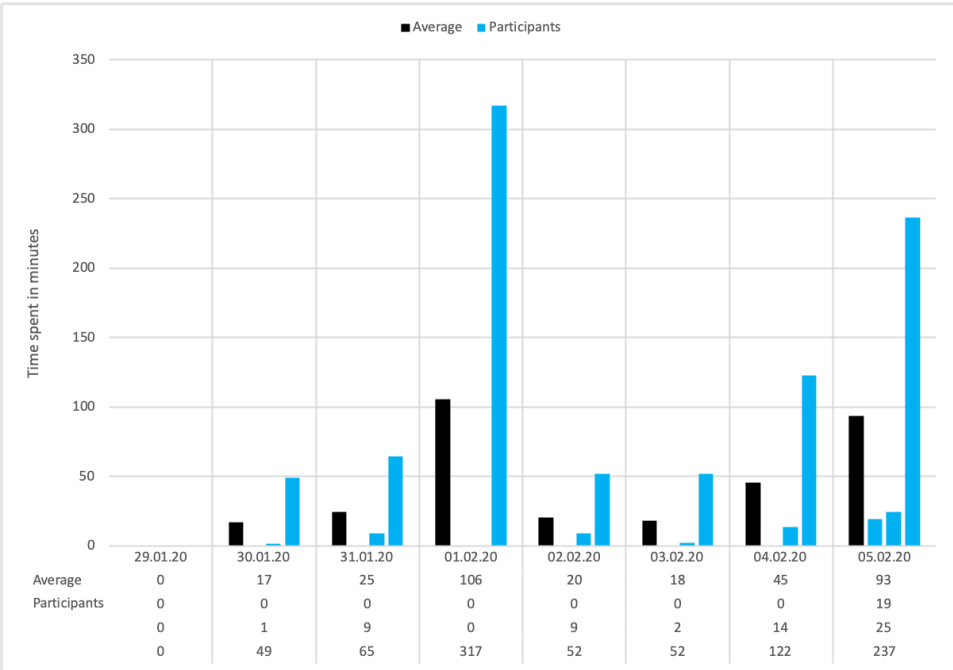


Figure 5.4.: Distribution of in vehicle raw values.

Date	Location	Users
29.01.20	(48.50108260577674, 11.495067909974292)	2
	suppressed	1
30.01.20	(48.50108260577674, 11.495067909974292)	3
	(35.50103138028429, 139.49688751384306)	2
31.01.20	(48.50108260577674, 11.495067909974292)	3
	(35.50103138028429, 139.49688751384306)	2
01.02.20	(48.50108260577674, 11.495067909974292)	2
02.02.20	(48.50108260577674, 11.495067909974292)	2
	(35.50103138028429, 139.49688751384306)	2
03.02.20	(48.50108260577674, 11.495067909974292)	2
	suppressed	1
04.02.20	(48.50108260577674, 11.495067909974292)	2
	(35.50103138028429, 139.49688751384306)	2
	suppressed	1
05.02.20	(48.50108260577674, 11.495067909974292)	2
	suppressed	2

Table 5.2.: Location values collected over the test period each day at 12 pm JST

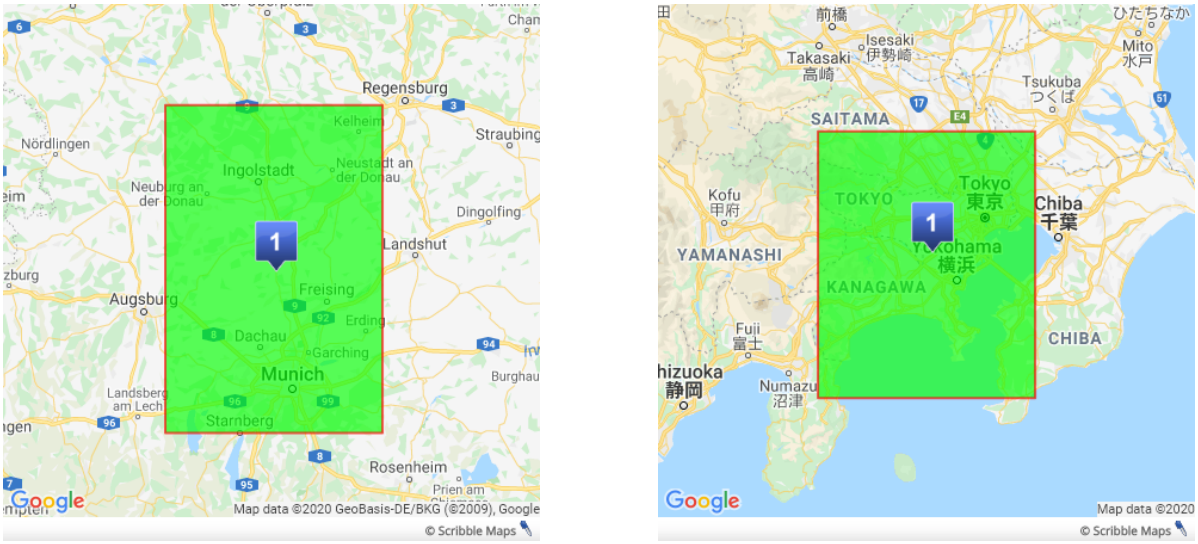
Date	Location	Users
29.01.20	*suppressed*	*suppressed*
30.01.20	*suppressed*	2
	(35.50103138028429, 139.49688751384306)	2
31.01.20	(48.50108260577674, 11.495067909974292)	3
	(35.50103138028429, 139.49688751384306)	2
	suppressed	1
01.02.20	(48.50108260577674, 11.495067909974292)	3
	(35.50103138028429, 139.49688751384306)	3
	suppressed	1
02.02.20	(48.50108260577674, 11.495067909974292)	3
	(35.50103138028429, 139.49688751384306)	3
	suppressed	1
03.02.20	(48.50108260577674, 11.495067909974292)	3
	(35.50103138028429, 139.49688751384306)	3
	suppressed	1
04.02.20	(48.50108260577674, 11.495067909974292)	3
	(35.50103138028429, 139.49688751384306)	3
	suppressed	1
05.02.20	(48.50108260577674, 11.495067909974292)	2

Table 5.3.: Location values collected over the test period each day at 12 pm CET

which represents the space between $\langle(35, 139), (36, 140)\rangle$. For the rest of the field test, we can determine two devices in the general area of Munich and 2 in Tokyo most of the time.

As for the aggregation at 12 pm CET or 8 pm JST, we get similar results with three devices in the greater Munich area and also three devices in the proximity of Tokyo for four days. On the 30.01.20, we only see 2 users the area covered by $\langle(35, 139), (36, 140)\rangle$ and on the 05.02.20, we only see 2 users between $\langle(48, 11), (49, 12)\rangle$.

Figure 5.5 depicts the areas that are spanned by the corner points for Munich and Tokyo.



(a) Munich $\langle(48, 11), (49, 12)\rangle$

(b) Tokyo $\langle(35, 139), (36, 140)\rangle$

Figure 5.5.: Spatial areas in Tokyo and Munich with an accuracy of 0

5.1.3. Privacy Evaluation

After collecting the data, we analyze it for privacy issues. We first put the steps data and activity data under the microscope, and then we tear down the location and presence information. Examining possible reconstruction, linking, or tracing attacks.

Raw Values of Steps and Activities

Looking at the steps and the activities, we do not see any vulnerability in linking attacks as there are not a lot of possible quasi-identifiers to connect. Being able to connect temporal similarities would be a potential vulnerability. Steps data is collected over the course of the whole day, but could also be modified to use a specific time frame instead. But the issue is, that all delivered data from every device has the same time data they get from the request. The same applies to activities. As every query by itself is a statistical database, without any quasi-identifiers, it is improbable to achieve a linking attack. Tracing attacks, however, can be used to identify if a user is used in the query, possibly. Taking the gathered steps

data, with auxiliary data as knowing the exact number of steps a person took, we can infer that the person has participated in the aggregation request with a high chance. But because our data collection architecture has a possibility that a person does not provide data, there is plausible deniability. Scaling the number of participants up, would result in a normal distribution of the mobility data, as shown by Althoff et al. With a lot of possible users, inferring the participation of one user, even with the ability to match the number of steps to exactly one value in the query result, we are unable to guarantee the user took part. We think we can assure privacy with the raw values, the same as Simon van Endern promised privacy for median values.

Location and Presence

As for the more specific mobility data, we ca not build any connections to the other data with confidence. We are incapable of linking our location or presence to steps or activities without auxiliary data. By suppressing unique locations, that could show individuals, and we have a k-anonymous data set. But the same problems apply to it as for steps and activities. Missing quasi-identifiers make it hard to re-identify the devices that the data came from. But in contrast to reconstruction or linking attacks, tracing attacks could be used. Because of the lack of participating devices, we are sure that it is the same people participating in the query. But with the low accuracy, we are unable to build any trajectories. To examine the possible calculation of paths, we need more data and more accurate data.

5.2. Simulated Test

Because of the small sample size in our field test, we decided to run one more test in a more controlled environment for more reliable data. We use 10 Android Emulators on different versions of the operating system using Android Studio. We selected four small landmarks in a small area from which the devices travel from one of the other landmarks as a destination. This will ensure that we have devices that have been alone and together over the test period. For 10 minutes, each device generates location data along a predefined route in an interval of one second, totaling in 600 GPS coordinates. We use the emulator's route simulation, which leverages Google's Map API, giving us a travel speed of exactly 4.5km/h. We then aggregate the location data over the ten minutes for each minute and gather data using different accuracy parameters, giving us different approximate areas. This gives us more data to evaluate privacy issues, especially concerning historical location data.

5.2.1. Results

First, we chose an accuracy of 2 decimal places, which translates to real-world distances of around one kilometer. We were able to track 7-8 devices consistently in the areas depicted in figure 5.6. The markers 1 and 2 span the area of $\langle(35.54, 139.63), (35.55, 139.64)\rangle$ and the other two markers are $\langle(35.53, 139.63), (35.54, 139.64)\rangle$. Table 5.4 shows how many devices were in which area at each time step.

Minute	Location	Users
1	(35.54500010319573, 139.63499968824985)	4
	(35.535000103183386, 139.63499968836484)	4
	suppressed	2
2	(35.54500010319573, 139.63499968824985)	4
	(35.535000103183386, 139.63499968836484)	3
	suppressed	3
3	(35.54500010319573, 139.63499968824985)	3
	(35.535000103183386, 139.63499968836484)	4
	suppressed	3
4	(35.54500010319573, 139.63499968824985)	3
	(35.535000103183386, 139.63499968836484)	4
	suppressed	3
5	(35.54500010319573, 139.63499968824985)	3
	(35.535000103183386, 139.63499968836484)	4
	suppressed	3
6	(35.54500010319573, 139.63499968824985)	4
	(35.535000103183386, 139.63499968836484)	4
	suppressed	2
7	(35.54500010319573, 139.63499968824985)	4
	(35.535000103183386, 139.63499968836484)	4
	suppressed	2
8	(35.54500010319573, 139.63499968824985)	4
	(35.535000103183386, 139.63499968836484)	4
	suppressed	2
9	(35.54500010319573, 139.63499968824985)	4
	(35.535000103183386, 139.63499968836484)	4
	suppressed	2
10	(35.54500010319573, 139.63499968824985)	3
	(35.535000103183386, 139.63499968836484)	4
	suppressed	3

Table 5.4.: Location values collected over the simulated test period for each minute with an accuracy of 2

Figure 5.6.: Distribution of still raw values.

Because the region we specified is large, we can almost always find multiple devices in the area.

For our next aggregation, we selected a precision of 3, which corresponds to roughly 100 meters. The data in table 5.5 reveals that we can find two mobile phones most of the time, with aggregations, in which we were able to find three devices or none at all. The areas are visible in figure 5.7.

Finally, we send requests with an accuracy of 4 or 10 meters. In this scenario, we were almost unable to register any devices, except for minute 10, see table 5.6. At this time point, we have two users that pass each other in $\langle(35.5372, 139.6336), (35.5373, 139.6337)\rangle$ as seen in figure 5.8.

5.2.2. Privacy Evaluation

Looking at the documented results, we can see where devices meet each other. With low precision, we always register 3-4 users. Using k-anonymity, we are unable to find out more than the provided data. We only know the number of people present in a specific area.

After we increased precision to 3, we mostly see pairs. According to the table 5.5, in the first 3 minutes, we can assume that two devices pass each other in the same area. Later one of them leaves the zone, and we stop registering that region. Without auxiliary information, we are unable to find out in which direction the users traveled or who left the area first. In minute 5 to 9, we register two adjacent regions. We can assume that there are more than two phones in the vicinity. Analyzing the historical spatial data, we suggest that we have two devices in area 2 and one device in zone 3 at minute 5. In the sixth minute, we register a new domain and loose the old one, indicating that one device moved from region 2 to 3 while the

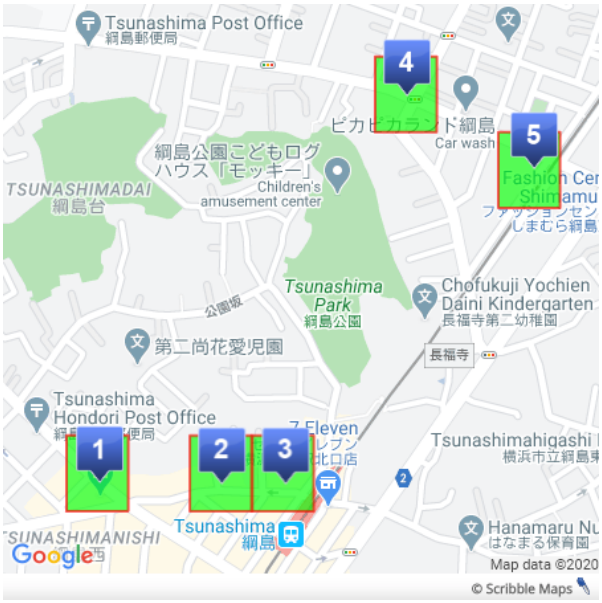


Figure 5.7.: Distribution of still raw values.

Figure 5.8.: Distribution of still raw values.

Minute	Location	Users
1	(35.537500001031866, 139.63149999688338)	2
	suppressed	8
2	(35.537500001031866, 139.63149999688338)	2
	suppressed	8
3	(35.537500001031866, 139.6334999968834)	2
	suppressed	8
4	-	-
5	(35.537500001031866, 139.63449999688333)	2
	suppressed	8
6	(35.537500001031866, 139.63549999688334)	2
	(35.54250000103193, 139.6364999968828)	2
	suppressed	6
7	(35.537500001031866, 139.63549999688334)	3
	(35.54150000103191, 139.6384999968829)	2
	suppressed	5
8	(35.537500001031866, 139.63549999688334)	2
	suppressed	8
9	(35.537500001031866, 139.63449999688333)	2
	suppressed	8
10	(35.537500001031866, 139.63549999688334)	2
	suppressed	8

Table 5.5.: Location values collected over the simulated test period for each minute with an accuracy of 3

Minute	Location	Users
1	*suppressed*	10
2	*suppressed*	10
3	*suppressed*	10
4	*suppressed*	10
5	*suppressed*	10
6	*suppressed*	10
7	*suppressed*	10
8	*suppressed*	10
9	*suppressed*	10
10	(35.53725000001032, 139.63364999996884)	2
	suppressed	8

Table 5.6.: Location values collected over the simulated test period for each minute with an accuracy of 4

other two are still in their respective areas. In minute 9, we then see the old area again and the new one vanishing, suggesting that the device that was formerly in area 3 walked into zone 2. We were able to infer the travel direction of one device, but are still unable to point out the device.

Using an accuracy of 10 meters, we are unable to collect any useful data with the lack of devices.

5.2.3. Usability

By stripping away all of the quasi-identifiers, we sacrifice a lot of possible usability in the data. But for the loss of value, we are able to preserve a lot of privacy and even provide anonymity. On the one hand, we wo not be able to use the collected data for detailed medical research that requires sex, age, and other information. On the other hand, however, we can gather population density data and activity data, which are very handy in urban planning. This provides a good starting point to find a balance between the usability of data and the privacy of the data provider.

5.2.4. Possible Improvements

We achieved a lot of scalability and a lot of privacy in our implementation, but there are still immediate improvements possible in both the application and the server. For the spatially cloaked location data, our design only applies static accuracy provided by the request itself. It may be possible to send different accuracies of the concealed areas and match them on the server, to get smaller anonymizing regions and, at the same time, still maintain k-anonymity.

The application should also be tweaked to be able to handle multiple requests at the same time. It is thus making data aggregation, in general, more efficient. The same applies to the server. At the moment, it does not differentiate the messages it receives from the groups. It is forcing researchers to wait for an aggregation to finish before they are able to send a new request.

As homomorphic encryption is still in its infancy, an alternative to protect the first participants from the subsequent is to add dummy data that can be removed at the end. This method would hide the primary data added to the aggregation and make it harder for the second device in the chain to determine the real input from the fake.

We also have an issue with the confirmation approach. This method is right now implemented using a sleeping thread that prepares to resend the data collected so far to the subsequent device after time out, but which in turn creates another sleeping thread. It may lead to multiple threads that use a lot of resources and might cause the application to freeze or crash.

6. Conclusion

While it is hard to strike a balance between anonymity and usability, we think we found a starting point on which we can build for a more scalable, secure, and privacy-preserving platform for collecting, aggregating, and analyzing of mobility data.

6.1. Research Questions

RQ1: What are the benefits and drawbacks in Simon van Endern's architecture?

After reexamination, we found that Simon van Endern managed to collect a lot of data while preserving privacy. Additionally, his idea provided a very secure communication infrastructure using a hybrid encryption scheme. His proposal, however, wasn't able to scale because of it, creating long wait times for data collection. Furthermore, the single aggregation chain leads to to a quadratic growth in data consumption, and our current data plans can't carry that burden. On the upside, the lone series of collection hides the connection between the data and the device. On the downside, his implementation of forwarding requests using a polling system is also a drain on the battery. Another flaw in the design is the selection process of the next device. Giving the server the ability to choose the following target allows it to choose compromised users and to endanger privacy.

RQ2: What improvements are possible?

In chapter 3, we proposed a few upgrades and extended further data collection. To help aggregation, we changed the polling mechanism to a straightforward event-driven one. Using a third-party push notification service, we are able to send requests directly to the participating mobile phones instead of giving them the job to fetch the requests from the server, improving significantly on communication time. We also suggested splitting the single aggregation chain into multiple groups, creating several smaller series of devices to collect the data instead of one long one, parallelizing the process, and cutting down on collection time. We also added the ability to request more accurate mobility data, such as the raw values of steps and activities and information about the location of devices or the presence of a device at a location.

RQ3: What information do the raw values for steps and activities reveal?

We added raw values to the aggregation of steps and activities as we saw no infraction on privacy by providing these values if there are enough participants. It is crucial to expand the

number of users to a certain number to provide privacy because as long as the numbers are low, we can trace the contribution of a participant in an aggregation. At a certain point, the steps and activities data should take the form of a uniform distribution, which then hides the individual raw data inside the collected data set. When the number of participants is reached and the inability to determine if a user participated in the collection request, we are confident that the privacy of the participants is secure.

RQ4: How much information does the aggregation of real location data expose?

We found that similar to other types of data, the number of participants is vital to protect their identity or at least their association. Using suppression, we are unable to hide devices, that are on their own. In chapter 5, we tested the platform in the field and a controlled setting. We were unable to draw any reliable conclusions regarding the usability of the collected data because of the lack of participants in our deployed field test. The privacy, however, we don't see any way how data can use to infer more sensitive data or identity. In the simulated test, we are able to collect much more reliable data. We were able to deduce the travel direction of a device but were unable to trace it back to other information.

RQ5: Can we use the current state of P2P technology to remove intermediate third parties?

Chapter 4 has shown our attempts to implement P2P technology into a mobile application. We have identified Textile as a viable candidate for our project, but unfortunately, one of the most important aspects didn't meet our expectations. There is a lot of progress in that technology, but some work still has to be done before it can be used in production. We should also look into decentralized communication standards, such as matrix, that can be used as an alternative to our initial direct communication approach.

6.2. Limitations

As mentioned before, there are a lot of short-term obstacles that we identified that could be improved in the short-term and some that warrant further research in the long-term.

First, both our field test and simulated test were executed on a very limited number of devices, giving us only a glimpse in our solution to scalability and privacy. The number of participants in an aggregation request can range from a few people in a rural area to a few thousand or even a few million in crowded metropoles. Further tests should be performed to collect a more extensive data set, and we expect them to confirm our assumptions.

In chapter 3, for the sake of the scope of the thesis, we assumed that the server and the mobile devices are trusted, and a malicious actor only has access to the aggregated data. The possibility of a compromised server and compromised mobile phones must be taken into consideration. Given an untrusted server, it is able to view the groups and their participants and the data received from each group. We have suspicions that the server is able to connect the values sent to the mobile device it was sent from by resending the same

request multiple times and observing the changes in the grouped collection. In case that there are compromised participants, the integrity of the data is in danger. The privacy of some devices might be jeopardized, but we believe that the mixed groups sent from the server cause enough change so that the malicious devices aren't able to build a profile. In the event that the server and participants are malicious, we aren't able to guarantee any protection.

6.3. Future Work

We have shown that the work we have done looks very promising, but a lot of issues were left untouched because of the limited constraints of the thesis. Below, we suggest a few subjects that could be considered in the future.

6.3.1. Mobile

The biggest extension would be to implement the mobile application for iOS platform. This opens up questions to cross-platform or native development. Furthermore, the Apple App Store has a lot of restrictions regarding the privacy and could eventually hinder the deployment.

With the trend in decentralization, P2P technology is being developed hand-in-hand by many organizations. Being able to communicate with the next devices directly, instead of involving an intermediary, will cut out a possible point of failure and provide a more secure communication channel.

As discussed in the earlier chapters, homomorphic encryption could be an excellent tool to keep data confidential while enabling arithmetic operations on them. Using this method allows for the calculation of values without knowing the raw data, preventing sensitive information from prying eyes. This technology might not pertain to privacy and anonymity directly but would still make the platform more secure.

Another technique to strengthen privacy would be to give the users the control to participate in the data collection themselves. An option in which they can decide what level of privacy the collected data holds. I.e., either setting the collection to very private, so that all data gathered with the option is protected from aggregation or just changing the privacy level of the data itself.

Also mentioned in the section above is that trust is a topic that can't be handled lightly. Either intentionally providing false information or trying to trace or link private data to a device are possible motives for a malicious user. To combat misinformation and privacy leaks, there has to be a way to establish trust. Pouryazdan et al. suggest a score to measure the reputation, which in our eyes could either help with data reliability or create a quasi-identifier for an adversary to exploit.

6.3.2. Server

Trust is also an issue for the server. With scandals constantly being reported on, the public trust in corporations and organizations collecting sensitive data is waning. Establishing the server as a trustworthy instance that doesn't mishandle the collected data by enhancing privacy or guaranteeing anonymity would be a step in the right direction. Relinquishing control over the data from a centralized architecture into a decentralized one could also help with this issue.

6.3.3. Blockchain

Leveraging the power of blockchain, the platform can incentivize the user to participate in aggregation. Ming et al. [X] suggest CrowdBC, a decentralized framework for crowdsourcing using blockchain.

On the one hand, compensating the crowd for participating on the platform would result in more data collected. On the other hand, however, the existence of monetary value would draw in the attention of malicious intentions, making it hard for data to be trustworthy.

6.4. Reproducibility

Further research on the collection, aggregation, and analysis of mobility data with the use of a scalable, secure, and privacy-preserving platform can be done using our implementation. In the README file, we noted down instructions for the installation and local deployment of the server. The necessary tools for the setup are also documented. Comments in the code will indicate the places additional changes, such as API keys for the push notification service and URLs for the push notification server and the database instance, have to be added to run it correctly. As for storage, we built the platform with MongoDB's cloud solution in mind, so utilizing an alternative might require changes to the database module. Section [X] shows our deployment option, but the server can be hosted on any platform, and the database could be deployed locally.

On the client-side, we have mobile phones. The application can be built using the Android Studio IDE and installed on any compatible Android device. Similar to the server, the comments specify the locations where URLs have to be modified. The application was built from the ground up with modularity in mind, making it possible to replace communication schemes.

First, the server and the database has to be started. After that, the Android applications can be started, triggering the registration process automatically when the permissions have been accepted. With everything running, aggregation requests can be initiated by consuming the REST API detailed in section 4.2, and the end result should be stored in the deployed database instance.

7. Introduction

Use with pdfLaTeX and Biber.

7.1. Section

Citation test (with Biber) [**latex**].

7.1.1. Subsection

See Table 7.1, Figure 7.1, Figure 7.2, Figure 7.3, Figure 7.4, Figure 7.5.

Table 7.1.: An example for a simple table.

A	B	C	D
1	2	1	2
2	3	2	3

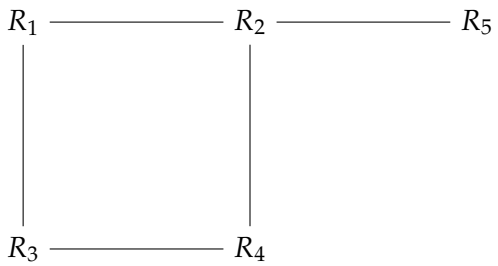


Figure 7.1.: An example for a simple drawing.

This is how the glossary will be used.

Donor dye, ex. Alexa 488 (D_{dye}), Förster distance, Förster distance (R_0), and k_{DEAC}. Also, the TUM has many computers, not only one Computer. Subsequent acronym usage will only print the short version of Technical University of Munich (TUM) (take care of plural, if needed!), like here with TUM, too. It can also be –> hidden[1] <–.

[(Done: Nevertheless, celebrate it when it is done!)]

[1]Example for a hidden TUM glossary entry.

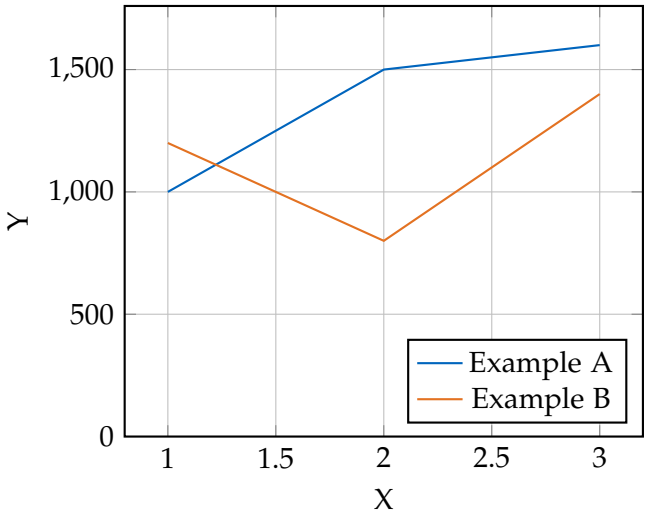


Figure 7.2.: An example for a simple plot.

```
1 SELECT * FROM tbl WHERE tbl.str = "str"
```

Figure 7.3.: An example for a source code listing.

Figure 7.4.: Includegraphics searches for the filename without extension first in logos, then in figures.

Figure 7.5.: For pictures with the same name, the direct folder needs to be chosen.

(a) The logo.

(b) The famous slide.

Figure 7.6.: Two TUM pictures side by side.

8. Second Introduction

Use with pdfLaTeX and Biber.

A. General Agenda

If there are several additions you want to add, but they do not fit into the thesis itself, they belong here.

A.1. Detailed Addition

Even sections are possible, but usually only used for several elements in, e.g. tables, images, etc.

B. Figures

B.1. Example 1

✓

B.2. Example 2

✗

List of Figures

List of Tables