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 round 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 increasing until 2100. They estimate that the earth will hit 10.9 billion at the end of the century 2100.

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

To solve issues stemming from the increase in housing, traffic and infrastructure, 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 planning housing and energy infrastructure. There are only a handful of companies that hold the monopoly on the much needed data, but acquiring them is expensive and 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 not enough. The use of inference attacks may make it possible to associate data and re-identify the individual as L. Sweeney has proven.

To solve some of the issues, we can leverage the widespread availability of mobile 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 vast 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 is 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, whilst collecting your sensory data, such as GPS and accelerometer, to stay informed about traffic information. Facebook provides a free social media platform for people to connect with each other, using your data to sell targeted advertisement.

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.

## 1.2. Research Questions

We try to find a balance between the usability of data, the privacy preservation of the data collecting user and the trust between them.

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

We take a look at Simon van Endern's original plan and implemented architecture. We investigate both 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: Do the raw values for steps and activities infringe on privacy?

We investigate the privacy concerns regarding raw step values for the participants and if they can be linked to other data.

### RQ4: Is it possible to query for more accurate location data while preserving privacy?

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 implemented to enable a more decentralized or distributed architecture.

## 1.3. Contributions

We will reexamine the results found by Simon van Endern and we will expand on his original idea while focusing on the scalability of the crowd sourcing platform and the privacy analysis of aggregation for more specific mobility data.

First, we will review related work concerning risks and solution to sensitive data and pertinent anonymization techniques. After that, we will propose our approach to an architecture to solve the mentioned problems. In chapter 4, we explain the details to our implementation and decisions. The next chapter documents our field test setup and evaluate our results.

Finally, we draw a conclusion on our work and discuss further possible improvements and show 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 have been solved.

### 2.1.1. Centralized

To improve 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 for the purpose of storing all data in a central location for statistical data analysis. In the end, the plans for the system were shot down because of the lack of privacy protection.

Today, there are several big tech companies, that are collecting information about their users and storing them in central databases. But centralization has a fatal flaw for protecting privacy. The collection of sensitive data in one location pose a high risk to their originator. Centralized databases that store private information are one of the weak points that have been on constant attack. In 2019, there have been over 5000 data breaches with almost 8 billion records exposed in the first three quarters, 33% more compared to the number reported in the first nine months of 2018. 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 harmful problem is the danger of re-identification.

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 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.

### 2.1.3. Location Tracking

In regards to location data, the ability to infer the home and work location pose 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 work place using historical location data. Being able to analyze the movement pattern and predict the presence of a person in a certain location may put his or her life in danger.

## 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 to lax security policies to human error. Figure [X] depicts two more alternatives in place of a centralized architecture: distributed and decentralized.

When the internet was implemented as 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, here for instance is to take away the control over data from monopolies.

Distribution takes decentralization a step further, by eliminating the central control. All instances have the same 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 removes the single point of vulnerability and lowers the the effort-to-reward balance and thus might deter malicious actors to try to steal data.

### 2.2.2. Homomorphic encryption

In a conventional way, encryption methods don't provide to anonymity. But one could argue, that in a way anonymity is provided, when the data is not readable or accessible. If an adversary manage to steal a data set while it is still encrypted, they won't be able to infer identity or any additional information, unless they are able to decrypt it beforehand. This protects the data set from intermediate parties, making them unable to derive private data.

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.

### 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 a malicious actor to link an individual to his data. One of the countermeasures to this problem, is providing k-anonymity. This is achieved when every query for a quasi-identifier returns at least k results. A quasi-identifier is an identifier or a combination of non-identifying attributes, that can 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 certain attributes into ranges. For instance instead of assigning the real age, an age range is used. This results in a loss of accuracy, but 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 don't 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 very powerful mathematical definition that is able to guarantee privacy. An algorithm is differential private as long as it disables tracing attacks, meaning that an output doesn't show signs of a particular individual being present in it or not. Both Google and Apple have implemented differential privacy to collect data.

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 composed 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 the true values. To avoid this, the default noise function 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 the noise function is applied on the 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 [X]. The advantages to 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.

#### Local Differential Privacy

Differential privacy can also be used on the entries of the data set. By running every entry through the mechanism, it creates a locally differentially private data set. The disadvantage to this method is the much higher noise, but in exchange the raw data itself is protected and doesn't require trust.

A common example for this is the coin toss. A survey asks a question that can be answered by a yes or no. The function would toss a coin and would answer the question honestly and otherwise answer the question with a second coin toss, with heads resulting in yes and tails in no.

### 2.2.5. Spatial Cloaking

In contrast to many other types of private information, spatial data can reveal a lot of sensitive information about a person. For instance, visiting an abortion clinic leads to knowledge about a pregnancy, which in addition could be used to assume sexual orientation. So location privacy should be always be high priority. To achieve that, there are a lot of algorithms to achieve spatial k-anonymity.

#### 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 [X] would show the lowest level of detail with $\langle(0,0),(4,4)\rangle$ and the highest level of detail with $\langle(0,2),(1,3)\rangle$. To fulfill k-anonymity, the algorithm looks for neighboring areas for other users and spans the whole area when found. For example, $U_1$ with its cloaked area $\langle(0,2),(1,3)\rangle$, it would need the area $\langle(1,2),(2,3)\rangle$ to satisfy $k = 4$.

### 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 $k = 4$.

### 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 a predetermined rasterization of the area, it the cloaked area is spanned by the points themselves.

# 3. Design

## 3.1. Trust, Usability and Privacy

In any platform, we expect a certain level trust from each participant, may it be from the service provider or the service 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 trust since they have been present for a longer period of time and can't just disappear over night, consequently they can be held accountable for their actions. Today, Microsoft, Google, Amazon and Facebook for instance, are trusted to a certain degree. But this trust is not invulnerable. Trust can be damaged by lies and secrecy. Keeping the public uninformed about severe data breaches or selling sensitive data to third parties without their knowledge and permission have strained the public trust in recent years.

Even if 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-indentified because of possible reconstruction or the data is not data anymore because it is not useful for analysis. 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 very broad research subject, we focus on the privacy of the crowd and usability of their collected data. We assume that all participating devices and servers can be trusted to a certain degree and possible adversaries only have access to the published end result.

## 3.2. Approaches

As discussed before, there are advantages and disadvantages Simon van Endern's system and 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 propsed architecture implemented in his thesis by Simon van Endern was with a centralized approach with a distributed storage solution. The model can be seen in figure [X]. In his thesis, he eliminates the need for a central database, by collecting and storing raw data directly on the crowds's smartphone, creating a distributed database. This gives each participant a lot of control over their own data, just by deleting the application they can opt-out of future data analyses.

His original idea, the mobile phones would use peer-to-peer technology to forward the aggregation request and finally send the data to the server. 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 polls the server for a new aggregation request targeted at the device. If one is present, it fetches the request over the REST API and after adding its own data posts it to the server.

In the finite scope, he managed to implement three aggregation types:

- Average number of steps of a day over all 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.

In his implementation, we see a few flaws, that we try to improve in this thesis.

### 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.

To make the architecture more scalable, we propose to parallelize the chains, by building multiple groups for aggregation, as shown in figure [X]. With $m$ groups, adding $n$ devices, now 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 $m$ would change linear and quadratic to a potential constant growth.

We also intend to implement differential privacy because ...

Additionally to avoid unnecessary constant polling, we will explore some P2P frameworks that can send data directly to the next target device and cut down on forwarding delays.

To extend the aggregation options, we add the two following two types:

- Number of participants that were at a location in a specific time period
- General location of all participants in a location at a specific time

# 4. Implementation

The following sections describe the implementation and explain 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 choice. In the end, we choose Android, mainly because of their ease of deployment on devices for test scenarios and 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 table [X].

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 a 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, table [X] shows that the users have to send their *username* and *password* for authentication, the *requestType* they wants to aggregate (steps, activity, location or presence) and additional search options as *request* depending on *requestType* to the server. The search options are visualized in figure [X]:

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 message sent to the first mobile phones can be seen in table [X].

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 message to the server contains the password generated on registration for authentication.

The implementation of the different JSON formats are flexible and can accommodate more aggregation types. The next to sections will describe 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 by Android 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. 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 [X] 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 and communications.
- *Data Storage*: This part stores and fetches all the collected data.

On start 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 in the database. A screen for each type is implemented: steps, activities and GPS. Using Simon van Endern's Android application as 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 accepts all permissions, the *Main Activity* starts the *Background Service*. The *Background Service* in turn starts four tasks that have been shown in figure [X]. 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 the 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 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 *long*titude.

### 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 intermediary as implemented in Simon van Endern's version, we researched 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 and 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 is 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.

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

**Push**

To replace Simon van Ender's polling with an event-based action architecture, figure [X] shows our new design. For our push service, we select Pushy because of their free entry version 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 to swap this method of communication for a P2P solution as soon as one proves viable.

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 for as identification. Then it generates a 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 the random password for future authentication.

The communication receiver works as the entry point for push notifications from the push service and adds data to the message. Upon receiving a aggregation request like in figure [X], it checks for an 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 hasn't.

After that, the communication handler takes the message and prepares it for the next participant in the *group* array in *requestOptions*. If the device was the last in the list, it sends the results 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 afterwards does the same to the symmetric key with 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.

### Bypassing inactive users in the aggregation chain

In the case, that the aggregation is stuck because of any reason, for instance a device doesn't 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 the their 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 push service bypassing devices, that have been offline and haven't contacted it service in a while. But it could also be the case, that the mobile phone isn't available after the aggregation already has 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.

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.

### Differential Privacy

### Spatial Cloaking

To be able 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 generalize the whereabouts by assigning the user to an 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 general 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 general data.

**Differential Privacy**

We decide against the use of differential privacy for our use cases because of the main flaws that it provides.

## 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 [X]. 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. In addition, it provides optional typing and better code structure. But we transpile our Typescript code to Javascript code, to run the web server.

When we start the server, *app.ts* registers the routes desribed in table [X] and connects the database object to the cloud storage instance. All calls to MongoDB are handled over 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 routes to which the researchers can send their aggregation requests. After receiving a JSON in the mentioned form in figure [X,X,X,X] 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 use k-anonymity. For steps and activity, we don't see immediate privacy issues and thus refrain from using k-anonymity as that would needlessly lower 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 don't fulfill the 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 computationally very expensive and there are currently no available libraries for Android to implement this encryption scheme. Another possible solution would be to add dummy data for each group and remove it afterwards, so that the subsequent devices can't discern the real data of the first participant from the decoy data.

Another issue, that can lead to privacy issues is the 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 aforementioned MongoDB 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 and tried to recruit people in the vicinity and over private social media channels, but 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 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 afterwards, we looked into location data and presence data. The collected data set can be found in the Git repository [X] and partially in the the sections below.

### 5.1.1. Data Consumption

Running the application by itself shouldn't 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 table [X], the distribution of the aggregation can be seen. We were able to request prove 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]. We can see in all cases that data consumption didn't exceed 10MB and in most cases didn't even use more than 5MB. We could extrapolate, that the consumption of data should 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 and 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 time zone respectively. Each aggregation had consistently from 3 to 7 participants.

We started with aggregating steps with around 5 to 7 participants being online according to our push service. The average number of steps we recorded, range from 7323 to 11362 with 3 to 4 users contributing to the data as the rest were did either not have an internet connection or no pedometer built-in. For the participants, we have a span from 0 to 21128 steps in a day. The data is visualized in figure [X].

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 [X]. 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 hasn't been used touched for two full days and thus the start date of the activity was before the 31.01.20 and 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 [X] 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 person is running regularly if the steps can be assigned to the same user.

On average, the participant uses a vehicle up to 106 minutes. On occasion many users don't use any transportation at all. Figure [X] 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 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 [X] and [X], 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 great confidence, the general trajectory of that user posing a risk to privacy.

According to the aggregated data, we have 2 to 7 participants over the course of the field test. Table [X] shows location at 12pm JST and table [X] at 12pm CET which corresponds to 4am CET and 8pm 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, but the range to 50,000km and the accuracy to 0, which results in gives us a rough estimate in a 111km range.

At 12pm JST or 4am CET, we have only 2 devices located in the greater Munich area with a mid point 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 3 devices in the same region and additionally 2 participants close to $(35.50103138028429, 139.49688751384306)$ which represents the space between $\langle(35, 139),(36, 140)\rangle$. For the rest of the field test, we can determine 2 devices in the general area of Munich and 2 in Tokyo most of the time.

As for the aggregation at 12pm CET or 8pm JST, we get similar results with 3 devices in the greater Munich area and also 3 devices in the proximity of Tokyo for 4 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 [X] and [X] depict the areas that are spanned by their corner points, for Munich and Tokyo respectively.

### 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, 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 don't 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 certain 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 possibly identify if a user is used in the query. 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 doesn't provide data, there is a plausible deniability. Scaling the number of participants up, would result in a normal distribution of the mobility data as shown by T. Althoff et al. And with a lot of possible users, inferring if one has participated even if we are able 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 can't built 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, 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 in participating devices, we are sure, that the it is the same people participating in the query. But with the low accuracy, we are unable to build any trajectories. In order to examine possible calculation of trajectories, 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 using Android Studio. For ten minutes, each device generates location data along a predefined route in an interval of one second, totaling in 600 GPS coordinates. We simulate the travel speed with walking and cover a small area, so we can use a higher accuracy parameter. We then aggregate the location and presence data over the ten minutes for each minute. This gives us more data to evaluate privacy issues, especially concerning historical location data.

### 5.2.1. Results

### 5.2.2. Privacy Evaluation

#### Location and Presence

#### Historical location data

### 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 for most cases. On one hand, we won't be able to use the collected data for detailed medical research that requires sex, age and other conditions, on the other hand however, we are able to gather population density data and activity data, which are very handy in urban planning.

### 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 still maintain k-anonymity.

The application could be tweaked to be able to handle multiple requests at the same time. Thus making data aggregation in general more efficient. The same applies to the server. At the moment it doesn't differentiate the messages it receives from the groups.

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.

# 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 reexamining, we found that Simon van Endern managed to collect useful data while preserving privacy. Additionally his idea provided a very secure communication infrastructure using a hybrid encryption scheme. His proposal however, isn't able to scale because of it, creating long wait times for data collection. Because of the single aggregation chain, the data consumption would grow quadratic and our current data plans can't carry that burden. The polling requests are also draining the battery whenever so slightly. 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 endangering privacy.

### RQ2: What improvements are possible?

We chapter 3, we proposed a few improvements and extend further data collection. To help aggregation, we changed the polling mechanism to a straightforward event-driven mechanism. 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 greatly on communication time. We also suggested to split the single aggregation chain into multiple groups, creating several smaller chains to collect the data instead of one long one, parallelizing the process and cutting down on collection time.

### RQ3: Do the raw values for steps and activities infringe on privacy?

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 are able to 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.

#### RQ4: Is it possible to query for more accurate location data while preserving privacy?

We found that similar to other types of data, the number of participants is vital to protect their identity or at least their association.

#### 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 the mobile application. We have identified Textile as viable candidate for our project, but unfortunately one of the most important aspects didn't meet our expectations. There is a lot of progress 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 direct communication.

## 6.2. Limitations

## 6.3. Future Work

### 6.3.1. Mobile

### 6.3.2. Server

### 6.3.3. Additional Information

### 6.3.4. Decentralization and Blockchain

### 6.3.5. Reproducibility

# 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 |

$$\begin{tikzcd}
R_1 \arrow[r, dash] \arrow[d, dash] & R_2 \arrow[r, dash] \arrow[d, dash] & R_5 \\
R_3 \arrow[r, dash] & R_4 &
\end{tikzcd}$$

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.

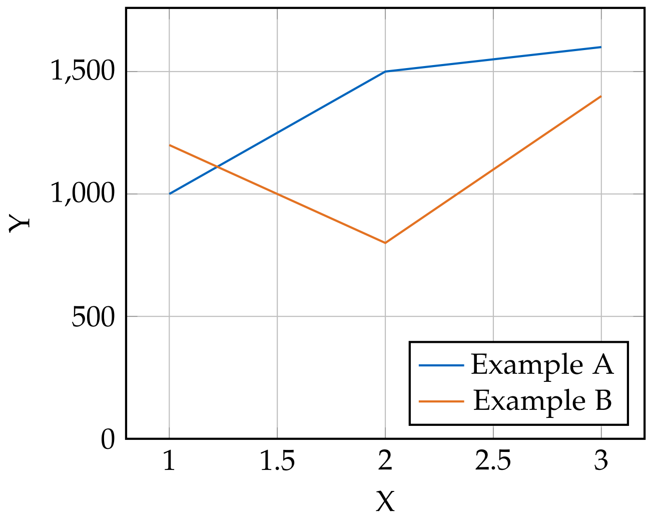

Figure 7.2.: An example for a simple plot.

```sql
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