DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Crowdsourcing mobility data with privacy preservation through decentralized collection and analysis

Simon van Endern

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Crowdsourcing mobility data with privacy preservation through decentralized collection and analysis

Crowdsourcing von Mobilitätsdaten ohne Einschränkung der Privatsphäre durch dezentrales Sammeln und Analysieren

Author:	Simon van Endern
Supervisor:	Prof. Dr.-Ing. Jörg Ott
Advisor:	Trinh Viet Doan
Submission Date:	30.06.2019

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

Munich, 30.06.2019 Simon van Endern

Acknowledgments

Abstract

We propose a method to publish location ata without raising privacy concerns.

As still this data could be useful for many stakeholders, we will investigate how on the one hand aggregated data can be published without imposing any privacy risk to the owners of the data and on the other hand develop a prototype of a mobile application through which this location data is aggregated in a decentralized manner so that the raw user data never leaves the users' device.

Contents

1 Introduction

1.1 Why we need an open-source location data approach

"Data is the new oil" is a quote many people agree with. It means that more and more businesses are based not on specific production capacities but on data, the ability to process it and the exclusive owenership over it. The success and monopoly of companies like Google, Facebook and Amazon can be attributed to this exclusive ownership to a significant extend.

While patents that used to power companies' success provide a balance through granting exclusive rights while having to make the knowledge public, many companies e.g. Coca cola have decided successfully not to go for a patent and thus not reveal their knowledge. If that approach is not compromised, it guarantees both - non-disclosure and also exclusive rights. Similarly, the non-disclosure of huge data sets collected by Facebook, Google and Amazon circumvent the balance intended by patents. The unavailability of huge amounts of data to the public is an impediment of innovation and increased growth. For example, cities would benefit from aggregated location data in order to optimize traffic scheduling as also highlighted by [23]. Nevertheless, the publication of raw data sets is impossible because it severly intrudes the privacy of the owners of the data.

So, even if companies would agree on a publication, a problem arises. There is a conflict between preserving user privacy and publishing user data.

Nevertheless, user privacy is already compromised even without publication of user data. Already the mere existence of central data sets pose a privacy risk to users, because security issues might allow for theft and unwanted publication of these data. An example is the theft of 14 million user data from facebook [22].

Some governments and other institutions already publish some of their data sets after anonymizing them e.g. through cloaking of data so that it achieves k-anonymity and there are crowdsourcing and open source approaches to make data available to everybody. Nevertheless, the applied anonymization is often not sufficient or at least critical if the resulting data set should still be useful. Research shows that inferences

can be drawn from the published data sets that violate the respecitve users' privacy. So, in addition to the main risk of a central data set, publishing anonymized data poses another risk to users privacy.

Furthermore, besides the remaining risk of inference attacks in published anonymized data sets, the anonymization through those algorithms always depend on a trusted server to collect the data from all users and then publish the results of any analysis applying privacy-preserving algorithms beforehand. So even if the data is only stored anonymized on the server, besides the remaining risk of inference attacks, this still imposes a high privacy risk to every user, as trust can be misued by the trusted server itself.

1.2 Research Question: No central raw data set but only aggregated data

RQ 1: What features does such a system require? RQ2: ... Nach dem Stil.

Clearly, in order to overcome the conflicit between privacy intrusion and (public) data availability, a solution is needed that gets along without storing raw data in a central data set. This solution should 1. eliminate the risk of leaking raw user data through theft from a centralized database and 2. eliminate the remaining risk of inference attacks on published believed-to anonymized raw data. So far, we have not seen an approach to fully solve this problem.

1.3 Contributions

For our solution, we will focus on the sub-area of location data and location privacy. We investigate the possibility of storing raw location data only decentralized on the collecting devices. On a central server available to the public, only aggregated data is stored, thus the main problem of privacy risk by a central database containing the overall raw data set is solved. Furthermore, the issue of trust is removed, as the aggregation process happens decentrally, thus the central server will never hold any other data than aggregated data. It will never know about the individual raw data.

In summary, our approach takes the opposite direction as todays standard. We do not first collect the whole data set and then reduce it to a data set meeting privacy-constraints but we start from the bottom up - first by performing analysis in a decentralized manner so that there never is an overal data set imposing a security risk on all the entries' users, and second by proposing a framework that only releases aggregated

data where no interference of any user information is possible. This data will then be available to the public. This gives us maximum possible feedback on eventual privacy problems, creates trust through transparency and fosters innovation through availability of data to everyone.

1.4 Outline

The structure of our research is organized as follows: First we re- view related work in the areas of location privacy and anonymisation techniques. In section **??** we describe our approach of decentralized data analysis to get along without a central database. Section XXX describes the setup in detail. Section XXX analysis the result from field-testing our application. Section XXX incorporates the results into our proposal of a possibility to achieve 100% privacy through all applications. Section XXX summarizes our work and points out further research possibilities.

2 Related Work

2.1 Classification of location data and apps that use it

In order to review existing approaches and research, we classify location aware services by the acceptable delay of the location information being available: Such a classification has already been made by [23].

1. Almost no delation tolerance: e.g. an application showing a pop-up about a nearby venue e.g. a coffe shop when a pedestrian passes
2. Some delay e.g. one minute is acceptable: An application e.g. google maps derives the information of congested traffic from devices reporting their GPS data which show lower than usual speed. As congestions worth reporting last longer than one minute, some delay in the device's information reaching the server is acceptable.
3. Significant delay of hours, days or even weeks is acceptable for historical and statistical use of location data e.g. to find out about popular visiting times

Most research investigates user's privacy in case 3 [Citations!!!] or 2. For case one there are already solutions available. We will first review research tackling location privaacy in case 3 and 2 and then briefly point out the findings for case 1.

2.2 Research has identified the following privacy problems

2.2.1 Central databases itself pose a risk due to possible theft

Centralized databases also expose the users to a security risk (through theft) [44, 24].

- [28] proposes the use of P2P over WIFI and Bluetooth to decrease the need of central instances.
- [29] proposes a secure approach where the raw data is hidden from the central instance but still the aggregated data can be obtained by using encryption methods. This approach is very close to our work. Also [24] is close to our work and uses encryption.

- [24] proposes an approach to handle user authentication.

2.3 Inference attacks on published data

2.3.1 Inferring home and work location from consecutive data samples

Research has shown, that even from a location data set that is pseudonymous, i.e. the identifiers have been stripped or anonymized from the data, it is still possible to infer the home location of single users through inference attacks [31, 15, 18, 24, 54]. The same problem arises when using data collected through crowdsourcing [29].

2.3.2 Inferring identity from home and work location

Furthermore, this location coordinates can then be combined with publicly available information e.g. reverse map coding of coordinates to addresses and then searching for entries in telephone books to infer the users identity from it's home location [31, 18, 24]. This identity can then be linked to other sensitive data. This problem also arises in the area of IoT [44, 24]. Often (though with usually lower probability) also the work address in addition to the home address can be inferred and makes linking the data to identities even easier [15, 18].

2.3.3 Solutions / Countermeasures to prevent inference attacks

Spatial cloaking: Achieving k-anonymity by dropping data points or perturbing them or dropping all data points around a random point around the home location [31]. More sophisticated approaches: [25]

2.3.4 Problems still after countermeasures

Data suppression algorithms have only limited success and can only reduce, but not eliminate the risk [24].

Data is useless if k-anonymity is guaranteed

Insufficient accuracy / the data set becomes useless [31, 15, 40, 47, 46].

Countermeasures not effective in sparsely populated areas

Anonymization techniques might score well in densly populated areas or areas with high traffic but poorly in sparsely populated areas especially where a single address

can be mapped to a single person or family [25, 3, 24] [location-privacy correct paper or cited wrong paper???] or might not work for individuals whos work and home location are further away than average [18].

Still privacy breaches possible

- More advanced privacy breaking algorithms
- Taking other sources into accouont, e.g. history of location data Extending the time period over which data is collected generally increases the risk.
- quasi-identifiers not thought of

2.3.5 All methods depend on trust to a third party or the provider itself

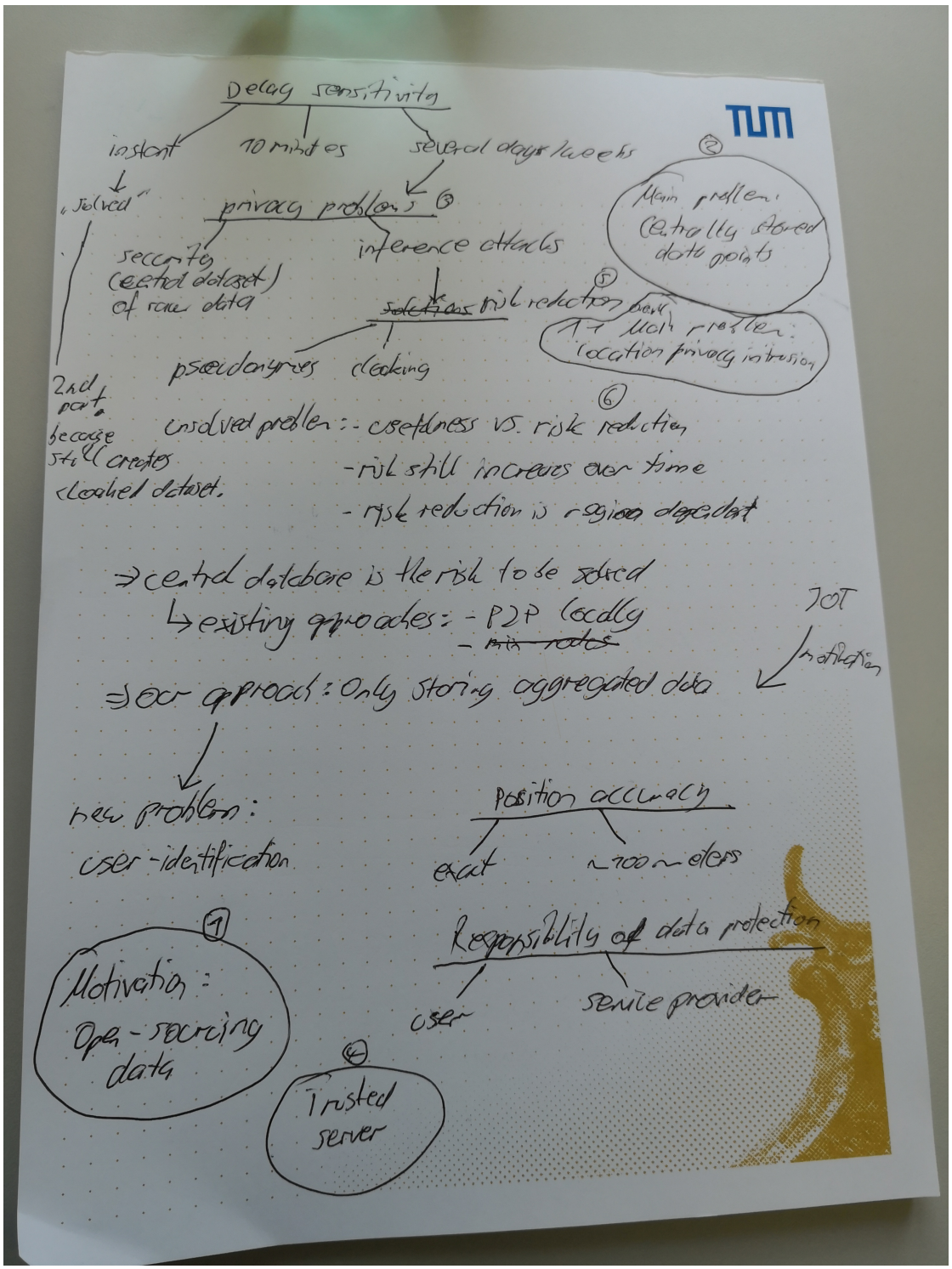
Still all approaches depend on first centrally collecting the original raw data and then before querying [47] applying anonymization techniques.

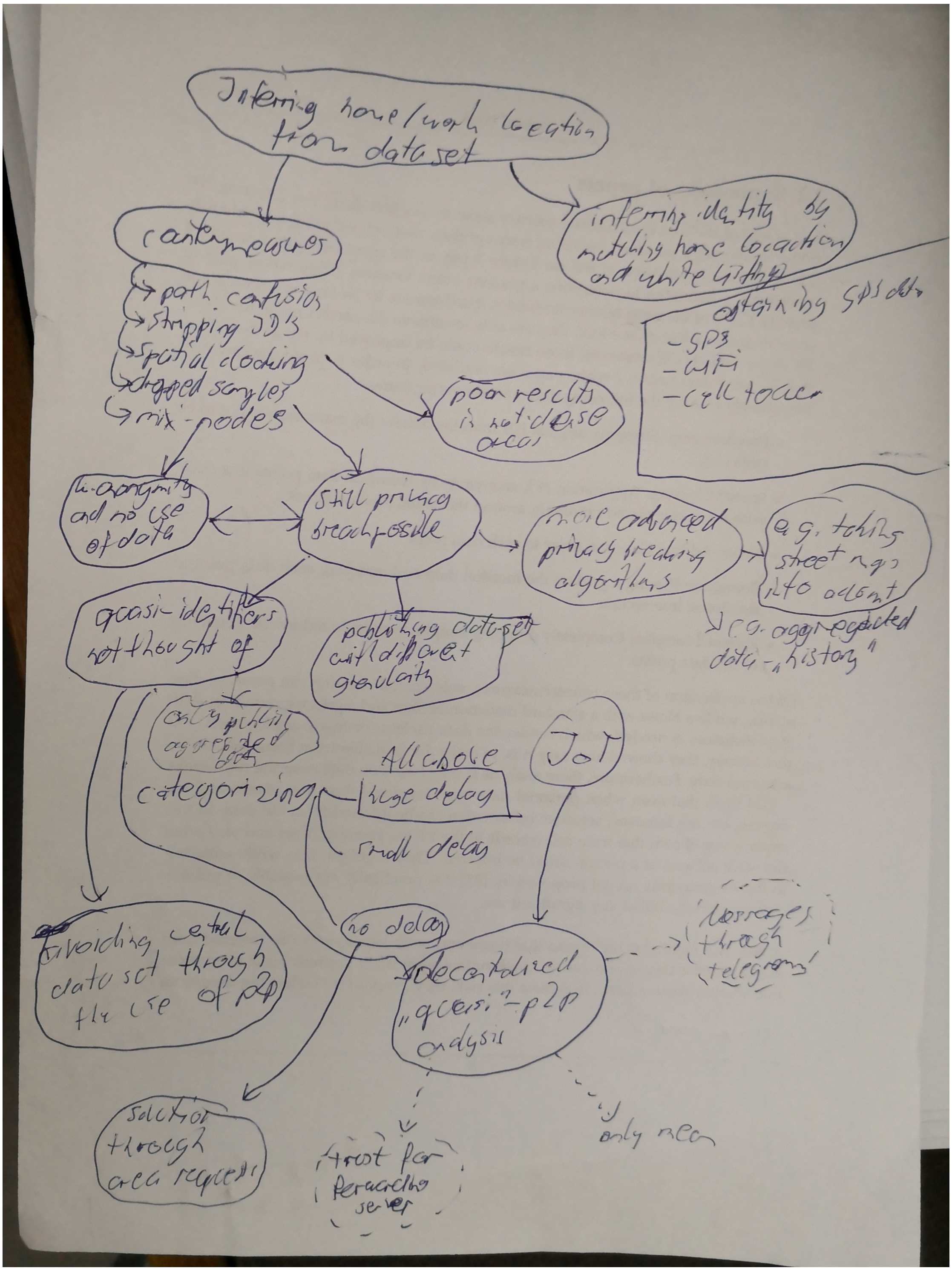
2.4 Category 1 location data use: instant

[3, 4] introduces mix-nodes, that can nevertheless not guarantee privacy and also depnds on a trusted third party. Also [36] proposes a solution (close to our summary) how to enable privacy for instant use of location data.

Decentralized methods for data analysis are also motivated from the area of IoT [44].

TODO: Relate to [47]


Delay sensitivity
instant
10 minutes
several days/weeks
"solved"
privacy problems
inference attacks
security (central dataset) of raw data
risk reduction
pseudonyms
cloaking
2nd part, because still central dataset
unsolved problem: - coarseness vs. risk reduction
- risk still increases over time
- risk reduction is region dependent
⇒ central database is the risk to be solved
↳ existing approaches: - P2P (costly
⇒ our approach: only storing aggregated data
IOT
motivation
new problem:
user-identification
position accuracy
exact
~100 meters
Responsibility of data protection
user
service provider
Motivation: Open-sourcing data
trusted server


Inferring home/work location from dataset
inferring identity by matching home location and white listings
countermeasures
path confusion
stripping ID's
spatial cloaking
dropped samples
mix-nodes
poor results is not dense area
obtaining GPS data
- GPS
- WiFi
- cell tower
k-anonymity and no use of data
still privacy breach possible
more advanced privacy breaking algorithms
e.g. taking street maps into account
e.g. aggregated data "history"
quasi-identifiers not thought of
publishing dataset with different granularity
only publishing aggregated data
All above
huge delay
categorizing
small delay
IoT
no delay
avoiding central dataset through the use of p2p
decentralized "quasi"-p2p analysis
messages through telegram
solution through area requests
only near

3 Methodology

We aim to solve two problems: First the raw dataset being available centrally itself and second the risk of inference attacks. Most research is based on datasets that span only over a few days or weeks. When collecting location data over months, the chance of successful inference attack increases in all cases with more data available.

In order to address the problem and test our hypothesis, we developed a framework in which we collect and locally aggregate location data on end user devices (smartphones). The raw data will stay on each device and will only be used to serve aggregation requests from a central server. The aggregation requests have to be defined upfront. An example is the determination of the average steps per day by each user. An incoming aggregation request might look like

```
{
"n" : 3
"value": 2000
}
```

And the data contained in the outgoing response after processing the request might look like

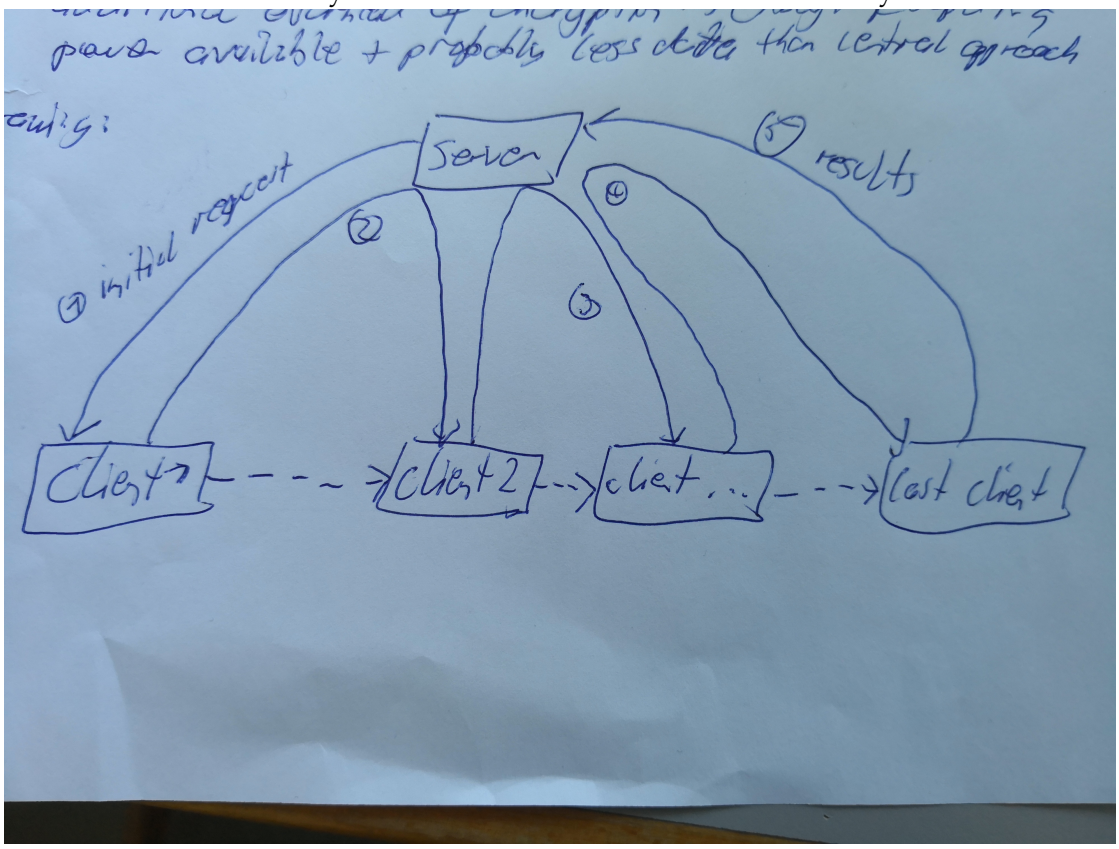
```
{
"n" : 4
"value": 2500
}
```

In order to protect the user's privacy and shield the raw data from the server, it would be necessary to pass the request P2P from one device to another until the last device finally publishes the result to the server. P2P on mobile phones though is hardly possible. Nevertheless, we found a way to oslve this problem: On installation, a public-private key-pair is generated and every installed application registers at the server with this public key. The corresponding private key is stored locally. When one end user device needs to send the processed aggregation request to the next phone, it encrypts the data using that phones public key in the standard hybrid encryption[1] approach

[1] In hybrid encryption as used in SSL, the message itself is encrypted with a synchronous key while this key itself is encrypted using the public key

leveraging the benefits of synchronous keys. This way the next phone will be able to decrypt the request and process the data while the server is unable to read the data until the aggregation request is finally send in plain text for publishing to the server. This process is depicted in figure XX.

The aggregated results are available only to our research team in order to protect the research participants privacy in case there is a privacy risk we have not thought of but the setting allows for them being available to the public. The aggregation requsts themselves are controlled by our research team and inserted on a daily basis.



4 Design and Implementation

4.1 Technology Stack

In order to implement the architecture proposed in chapter 4 we chose Android as end user device platform. Android has the highest market share among mobile devices and offers a healthy ecosystem of frameworks and libraries that simplify development. Furthermore we opted for an implementation in Kotlin to reduce boilerplate code and improve readability.

As server side technology we chose node.js in conjunction with a mongoDB object store database. A NoSQL database like mongoDB provides flexibility and easy adaption of data schemes without much overhead and thus perfectly fits our prototyping purpose. We chose node.js out of the same reasons. In contrast to a statically typed language, javascript provides more flexibility and ease of change. Furthermore node.js is often used in combination with mongoDB and probably provides the best integration of it.

The results can be made available to the public either via a dedicated route of the server or directly through granting right access to the respective collection of the database.

4.2 Android Application

The android application can be grouped into three main parts:

- A module responsible for collecting location data
- A module responsible for locally aggregating the raw data
- A module responsible for handling aggregation requests

The application is programmed with API level XX as target level and a minimum Android API level of XX. Approximately XX% of devices support this minimum API level which allows our application to be installed on almost any Android device. Furthermore, the application leverages google services. Without google play services, the application will not work. In order to ease future adaptability, we chose to use the Dagger2 framework for dependency injection. Further use of frameworks and libraries will be explained in the following respective sections.

4.2.1 Separation of concerns regarding location data

We chose to separate the aggregation and collection of location data in order to decouple the modules and provide the possibility to extend the model of aggregated data in the future without the need to chose the raw data model. Vice versa the data collection process can be modified without impacting the aggregation process.

4.2.2 Overall Android Architecture

The application has only one Main Activity in order to ask the user to grant location access. Apart from that the only Activity does not serve any specific purpose. It displays the 10 most recent database entries for some tables, used during the development and testing process. The application itself is structured into loosely coupled modules taking care of retrieving and saving the raw data, locally aggregating the data and communicating with the server. The local aggregation as well as the polling of new requests from the server happens on a 15 minute interval. The Android Workmanager controls this periodic work. For the App in order to have maximum possibilities collecting esspecially GPS data and preventing the Android operating system from shutting down when not interacted with by the user (which is usually never the case), a non-dismissible status notification is displayed at all time. (Compare to the non-dismissible status notification displayed by Google Maps when the navigation system is active). Furthermore, the application, respectively each module is heavily unit tested in norder to guarantee functionality and facilitate further development by other research teams. Unit tests are based on JUnnit 4 for Android and expresso.

4.2.3 Data collection

We use the Android Room Persistence library which is a layer on top of the standard LiteSQL database used in most Android applications. We collect three types of data:

- Steps: If available, the phone internal step sensor provides updates on a regular basis choosen by the user. We chose XX minutes as update interval. The step sensor always retrieves the total number of steps since last reboot. Upon each time we receive data from the step sensor, this data is stored directly in the "steps_raw" table.
- Current activity: The google services activity recognition API leverages different data and sensors available on the phone in order to inform about the current activity as one of [STILL, WALKING, IN_VEHICLE, ON_BYCICLE]. Whenever there is a change detected, two events are fired - one for exiting the former and one for entering the current activity. The events might not be dispatched instantly

but contain the timestamp of the exact occurence. Upon each received event, this data is stored directly in the "activity_raw" table.

- GPS positions: GPS data is collected via the GPSFusedLocationProvider which leverages cellphone-tower and WIFI data apart from GPS to determine the position. In order to limit battery consumption, GPS data is only requested every 5 minutes if the device is idle. [1] If the device is in state WALKING (see last item), the interval is set to XX seconds and in any other state, the interval is set to every second. The data is stored in the linked tables XX and XX. We chose to separate the GPS point itself from the timestamp having in mind that future aggregations might need or leverage the separation of spatial data and time. Especially regarding the high load of data - up to several 10 thousand GPS points per day.

4.2.4 Local data aggregation

Steps are simply aggregated on a daily basis and stored in the steps table. The exit and enter events received via the activity recognition framework and stored in the XX table are matched in order to compute activities with start and duration. Those are then saved in the "activity" table. GPS data is currently only used to compute trajectories via the following algorithm:

1. When there are more than 10 minutes between two subsequent GPS points in the sequence of GPS points to be processed, the sequence is separated into two separate possible trajectories and each is processed separately in the next step.
2. First we identify still moments - periods of no movement - as follows:
   a) For each GPS point, the next point that was registered at least two minutes after the first one is identified.
   b) If the average speed between those two points was below 0.6 m/s, the pair is added to a list.
   c) After iterating over all GPS points, the list resulting from the last step is fused into sequences as long as possible.
3. Those sequences / still moments are cut out of the original sequence. The remaining subsequences are our trajectories.

[1] Neertheless, if other applications request a GPS position, our application also receives this data, even if it occurs on a faster interval than our set interval

The start and end location as well as time of those trajectories is then saved in the "trajectory" table.

Example of algorithm:

```
{
Original dataset:
Latitude Longitude Time
44  11    10:11:03
44.5  11.1    10:11:15
44.4  11.05    10:12:12
44.3  11.07    10:33:00
44      11.2    10:34:00
}
```

4.2.5 Serving aggregation requests

We use the retrofit2 framework based on OKHTTP to handle communication with our REST server. An HTTP Interceptor is used to modify incoming and outgoing requests. In the former case, the interceptor decrypts the requests body using the private key of the installation before the body is parsed into Java Objects. The latter is used to add authentication to outgoing requests. The app polls for new aggregation requests on a regular basis. New aggregation requests are first saved locally to the database, then processed and again stored locally and finally send back to the server. This separation of concerns is useful especially in case of an interrupted communication during processing the aggregation request. //TODO: Image with repository. The aggregation itself takes the type parameter to specify which actions to take on the three fields (nn:int, value:Float, valueList: List<Float>) shared across all aggregation request types. In case of the tyes "steps" and "activity_X" the field value contains the mean and the field n is the number of participants so far. In case of "stepsListing" only the field "valueList" is used and filled with the mean value of each user. In case of "trajectories", only the field "valueList" is used. Four subsequent elements of the list always represent one trajectory as of latitude of start, longitude of start, latitude of end and longitude of end.

4.2.6 API

The API is designed as a pure JSON API. The API consists of the following endpoints:

- POST /user - for creating a new user
- GET /requests - for retrieving aggregation requests for a user

- POST /forward - for sending a processed aggregation request back to the server
- GET /aggregations - for retrieving all completed aggregation results
- POST /admin/sampleRequest - for starting a new aggregation request.

The routes used for interaction with the application (except the one for creating a new user) are secured and can only be used if the users can successfully be authenticated. The routes for starting new aggregation requests and retrieving all results require administration authentication. The route for the results was designed to be available to the public without authentication but restricted in our setting to avoid privacy issues that might evolve due to the experimental request. Figure XX depicts an example usage of each of those endpoints. There are some other routes that are only used for debugging purposes.

4.3 Data aggregation design

All aggregation requests have in common that the underlying data is specified through start day and end day. Thus the minimum timespan for an aggregation is 24 hours. All other timespans are multiples of 24 hours. The following aggregation requests were implemented:

1. Computing the average number of steps walked.
2. Computing the average time spent walking, in a vehicle or on a bicycle. [2]
3. Aggregating a list of the average number of steps walked during the timespan of each user.
4. Aggregating a list of all trajectories registered by the users phone.

We had no doubt that aggregation 1 and 2 do not pose any privacy risk to the users participating in the aggregation. Aggregation 3 was implemented to test whether this type of aggregation, which allows for more advances statistical computations as median values and distribution properties, still completely preserves privacy. Aggregation 4 imposes a privacy risk discussed in XX on the user and was implemented in order to get an overview of the data quality of our setup and validate the feasability of the other aggregation requests proposed in XX.

[2] https://developers.google.com/android/reference/com/google/android/gms/location/DetectedActivity

4.4 Server Design and Implementation

The server is build using the event-driven node.js verion 10.15.3 leveraging the express [3] web-server framework and using the mocha [4] testing framework in combination with the chai [5] assertion library for unit and integration testing. The layered architecture starts with the server - server.js handling authentication and updating the respective users lastSeen property. Afterwards routes.js connects each API endpoint to the respective function in requests.js which contains the overall logic of the endpoint. The repository module than handles those requests according to the underlying datamodels and persists data in a mongoDB object store. server.js also invokes a scheduled task which re-routes stale requests where the user has not proceeded with the pending request either due to being offline or due to a problem handling the request. When new aggregation requests are started, the lastSeen timestamp of users is taken into account to exclude users that have not connected for a certain time. Furthermore, the list of users who are selected to serve the new request is ordered by the time the user was last seen.

4.4.1 Data Model

We organize the data in four collections[6]. The user collection stores the user data which is the public key, the hashed password and "lastSeen" - the timestamp of the last interaction of the user with the server. Aggregation requests are split into two collections. The collection "rawAggregationRequests" stores the initial aggregation request inserted through the admin interface containing the fields start, end, type - the type of the request, the three fields n, value, valueList reused across all aggregations to pass data, the timestamp when the request was filed to the server and a flag indicating whether this request has been started yet[7]. Upon start of the aggregation request, a list of the 10 most recently active users is retrieved in order to serve this request. The request body is then encrypted with the first users public key and stored in the collection "aggregationRequests". Each time, a user requests an aggregationRequest, proceeds with it and sends the results back to the server, the result is inserted into the database as a new aggregationRequest. The fields of this collection are

- rawRequestId - The id of the related rawRequest. This field is not available

[3] https://expressjs.com/
[4] https://mochajs.org/
[5] https://www.chaijs.com/
[6] A collection in an object store is the equivalent to a table in a SQL database
[7] E.g. when the end data of a newly inserted aggregation request is in the future, the request will be started only when this day has passed

through the API.

- started_at - The timestamp, when the request has been started
- publicKey - The public key of the user that should proceed this request
- nextUser - The public key of the user that will receive the request afterwards. This is necessary so that the user that should proceed this request can encrypt the processed request with the public key of the next user.
- previousRequest - The id of the previous request. This is null, if it is the first request in the chain. This is used for the mechanism taking care if a request is not processed by the user it is pending for.
- users - the list of public keys of the following users that will proceed with this request. This field is not available through the API.
- encryptionKey - A synchronous key, encrypted with the public key of the user the request is aimed at.
- iv - the initialization vector used for synchronous encryption and decryption of the actual aggregation request.
- encryptedRequest - The actual aggregation request encrypted with the synchronous key.
- timestamp - The timestamp when this object has been created.
- commpleted - A flag indicating whether this aggregation request has already been proceeded by the respective user and the resulting aggregationRequest has been received by the server.

The last collection called "aggregationResults" is used to store the results of an aggregation request. Once there are no more users to serve an aggregationRequest, the last user sends the final data unencrypted to the server where it is stored as an aggregationResult. It contains the same fields as the rawAggregationRequest except the started flag and additionally a field started_at and timestamp - indicating when the aggregation request referenced through rawRequestId was started and when it was completed.

5 Performance and Evaluation

5.1 Deployment

The proposed Android application and server have been tested on 16 devices for one week from 30.05.2019 until 06.06.2019. The server was deployed in the IBM Cloud as an 128 MB node.js instance. The database was hosted as a free version at mongodb.com. We ran each of 8 requests on a daily basis and also for each timespan of several days within this period. The raw results can be found at XXX[1]. We used this testing period also to improve the performance of the server as well as the Android application and to find and remove bugs.

5.2 Results

The results support our hypothesis that data can be analyzed decentrally and that aggregated data can be published without any privacy concerns. The aggregations of mean values clearly leave no doubt about full privacy protection as there even is no personal data involved anymore. The listing of mean values of average number of steps per participant allows for more advanced statistical analysis while at the same time the values cannot be mapped to persons. Even when conducting the same request twice, due to users being chosen dynamically, one could most probably see if the same user participated in the second request but nothing else. When aggregating over another time period (that might have an intersection with the other one), there is not even the chance to identify whether the same user participated in both aggregations. As requests are started not at the same time (TODO!!!), and users are in a future setting allocated dynamically to the request, there is also no chance to link the data from different aggregations. From the number of steps one could infer the time somebody spent walking, but as it is not given whether the steps where conducting walking, running or both, this linking would result in a very poor performance and also only reveal that to a very low probability, the user with X steps in one aggregatioin is the same user spending X minutes walking the same day. The listing of trajectories created

[1]Only 7 of the 8 aggregations are available as it is. The results of aggregation 8 where modified in order to protect user priacy

a dataset that clearly shows the vulnerability highlighted in XX and XX. Nevertheless, this is no aggregation but just a collection of raw data with stripped of identifiers and timestamps anonymized to a daily basis. The results show clearly that our setup is sufficiently accurate to field test the other aggregations proposed in XX and further prove our thesis. The data shows, that e.g. A change of transportation system can clearly be identified.

6 Conclusion and Discussion

We have shown that our hypothesis holds but only for aggregated data. This is fine because except in one experimental setting as with trajectories, there is no need for (anonymized) raw data. Also the experimental setting could be replaced by directly implementing the aggregations and testing with a greater user base. In theory, the anonymity and preserved privacy that hold for the tested aggregations hohlds also for the other proposed aggregations and aggregations we have not evaluated here. In order to follow with this research, we provide the setup to easily implement and test those and further aggregations in future research in order to further support our hypothesis.

6.1 Limitations

- As mentioned in XX, our system is based on trust. In case of user data being compromised, this significantly impacts some of the results. Nevertheless, collecting the list of mean values is far less error prone as the outlier could also be identified.

6.2 Future Work

- While XX has found that inference attacks can be based on the same dataset being published two times with different anonymization techniques applied and XX shows that anonymized datasets that overlap poses a risk, it still has to be investigated whether overlapping aggregated data as in our case can pose a risk.
- Some of the techniques identified as useful, such as spatial cloaking, ... should be applied in our setting.
- Our framework would also allow to pre-populate (simulated) smartphones with artificially generated or otherhow collected data in order to test and verify the functionality.
- Another use of our framework is the area of decentralized computation. Problems might be solved locally and collected by the server afterwards and the pieces put together still with anonymity for the users.

7 Design

7.1 Overall Design

Our architecture comprises the following:

1. An Android Application
2. A server application
3. A public database (with a visualizing website)

7.1.1 Android Application

The Android Application needs the following:

1. Create a public/private key pair
2. Automatically collect Location Raw Data for the scheme Timestamp, Location –> TODO: Implement database in Android Application
3. Ensure that even when the app is not running, data is collected.
4. Create activities from the location raw data for the scheme From, To, Duration, type, ...
5. Algorithm to infer home and work location (or ask for home / work location (rough specification)).
6. For each day? week?, ... compute the areas where the user was active. (This can be done as an aggregation request as well)
7. Stage 2: (or even stage 1?): After each day, send anonymously, thus over several nodes, in which area the device has been active! ~~(that is a breach of our privacy!!! find solution) in order for the server to know to whom send requests regarding locations.~~ The most possible fine level of user location will be stored alongsied the id

8. Send registration request to server application (including most coarse location). If successful, less coarse location will be send until unscuccessful. The server will store the request for the least coarse location area and count the following requests. Once a threshold is met, it will "unlock" this location.
9. Mobile might send (privacy?!?!?!) most coarse location after one week etc. again. TODO: Think about least coarse area useful. (city, department, ... level) The location can be more coarse than needed for the aggregation request. This will create network overhead, thus a balance has to be found.
10. receive Aggregation request from server application
11. calculate response to aggregation request
12. Apply logic when to abort aggregation request e.g when incoming n is 0 and next device ID is empty.
13. send aggregation response
14. At least one screen displaying some text / information about the application and hinting to turn off battery saving mode
15. Automatic hard-coded push notification to check for an update at date xxx
16. Automatic hard-coded deletion of all data after test-phase including a push notification to ask for the deinstallation of the app.

In stage two, the App will also be able to send e.g. traffic alerts to the server.

7.1.2 Server application

The server application needs the following:

1. A database containing counters for each level of reported active location to "unlock them".
2. A database listing all registered devices following the scheme public device key, Google Cloud messaging key (for reaching the device), most accurate possible last location.
3. A database containing all possible aggregation requests
4. A scheduler to start / send out aggregation requests

5. A handler for an ongoing aggregation request (forward it to the next one, until done or aborted).
6. Store a requests result in the public database

In stage 2, the server also processes data send by the client on its own behalf. The server aggregation task does the following

- ~~Randomly select a fake-start-n (out of the range of lets say 1-5) in order~~ With encryption not necessary
- ~~Select fake-start-value~~ Not necessary with encryption
- Devices list for the aggregation request (Will be made dynamic in stage 2).

7.1.3 Public database

The public database comprises the following schemes for aggregation requests. Furthermore, it handles incoming data e.g. traffic alerts. The aggregation schemes will all contain at least the following fields:

- Current n
- Current mean / value **Use json (or xml) for value passing**
- ID ?? (necessary?)
- Next device's public key for encryption of the data (and n?)
- Timestamp start
- Tmestamp end –> duration

7.2 Specific designs

7.2.1 Standard user story

Our user is called Hans

1. Hans somehow gets motivated to go to the playstore and install our appliation
2. In the playstore, Hans sees some photos and information about the application
3. Hans clicks on the install button in the playstore to install our application. The installation process starts.

4. Hans is curious about the application and opens the application.
5. Hans sees the first and only screen of the application that tells hime what the application does. It also contains a link to view the results stored in the public database.
6. (In stage 2, maybe Hans can even see his data and what has been send, ...)
7. Hans leaves the application (the application must still go on in the background).
8. Hans uses his task manager to quit the application (the application must still go on collecting data in the background).
9. After some days, without Hans being involved, an aggregation request is started and send to the appliation running in the background.
10. The application receives and processes the request and sends the results to the server without Hans noticing anything.
11. One week after the installation, tha application creates a push notification and asks hans to update the application. It also says that the update is less than 1MB and asks him to install it right now, as it is so few data and in order not to forget to do it later
12. Hans updates the application and thus installs all the fixes we have done in the meantime.
13. After the end of the testing period, the application automatically deletes all data. Hans gets shown a push notification informing about this and is asked to deinstall the application. A thank you is displayed as well.

7.2.2 Data aggregation schemes

We will in stage one ask all devices and only in stage 2 allow for limiting to specific areas. The minimum n is always 5 (choosen out of "Bauchgefühl"). TODO: evaluate which n values are necessary. When computing average and median, n is at least 10 (Bauchgefühl). When reporting a full box-plot, n is at least 20 (Bauchgefühl). For reporting skewness and standard deviation as well, 50 is the minimum for n (Bauchgefühl).

We definitely need a strict approach for overlapping, etc. to also take care of cases we have not thought about.

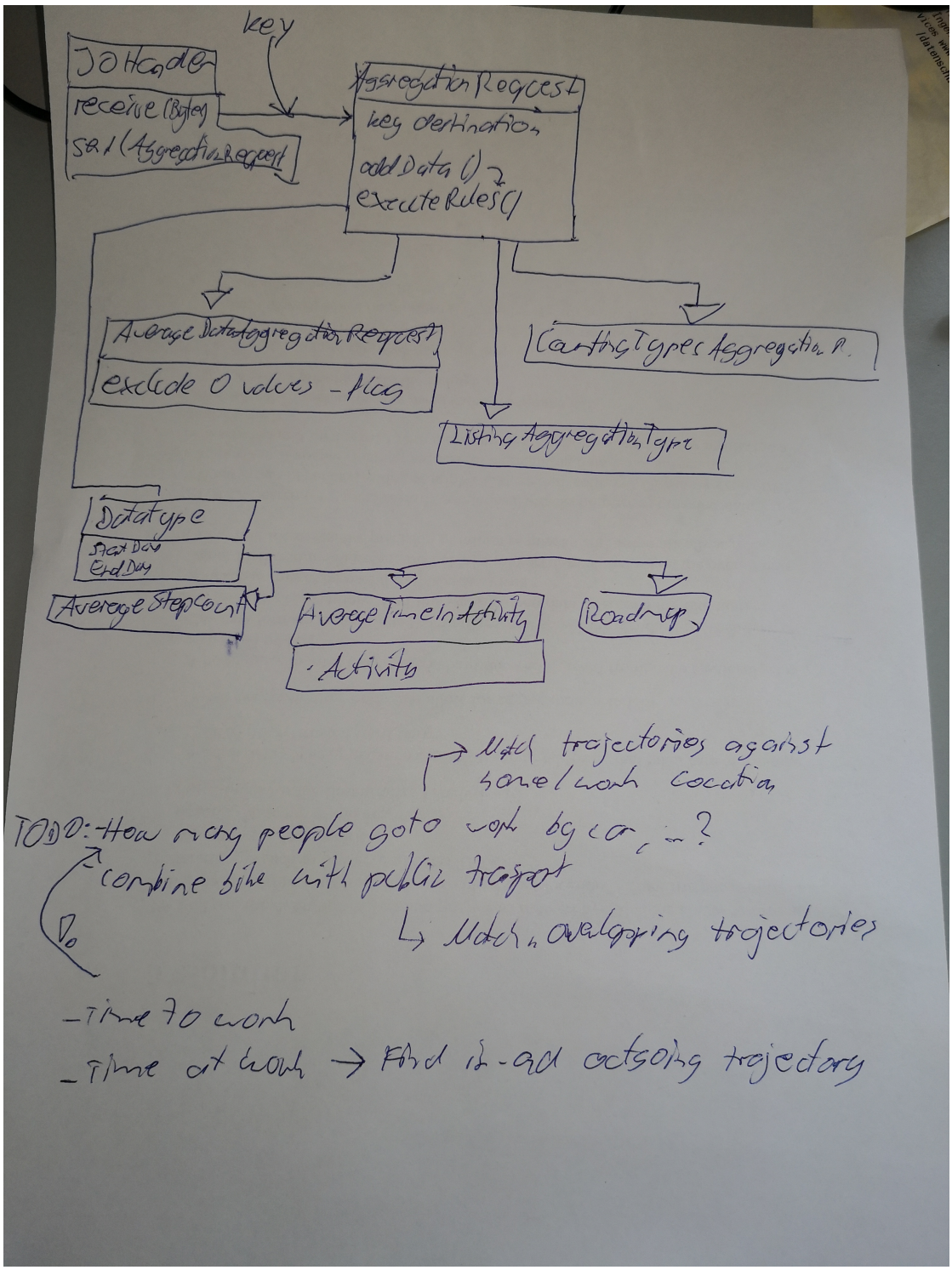
TODO: From overlapping, ... it will be possible, to compute "There is somebody, generally not walking, but biking, in this area, ... ". Check, whether part of this can somehow be used as a quasi identifier.

Resulting aggregators

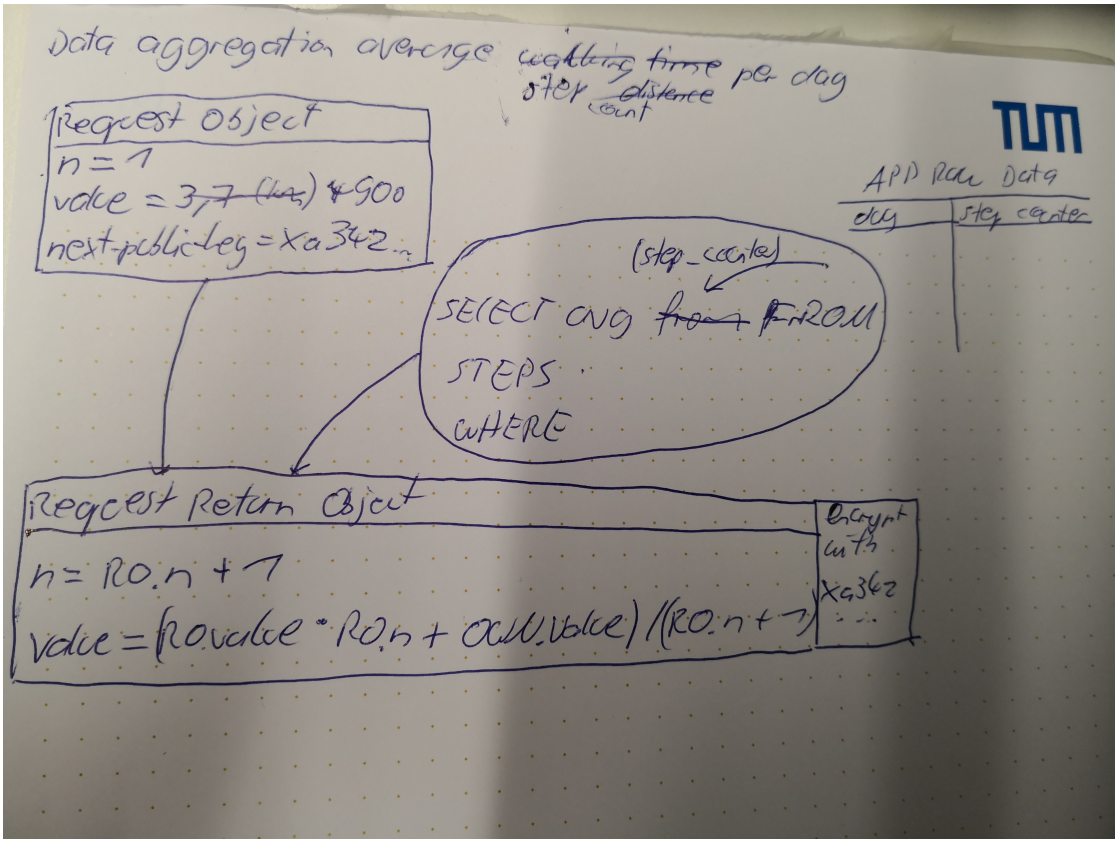
Pending question: Is the aggregation of the whole aggregational dataset unproblematic? At least the last node must randomise the database entry order. Important is that aggregational datasets do not share information so that linking could be done. How can a device validate that the public key obtained is actually a public key of another device and not created by the server itself???

- Average data collection
- Average data collection excluding 0 values
- Counting types (e.g. driving to work, walking to work, biking to work)
- Listing pairs of each (longitude, latitude) and the activity (driving, biking, walking) and speed
- Median (in combination with average request. Fake values must be consistent): Collect the samples themselves and at the last stage compute median , .25 and .75 percentile, skewness, standard deviation. ATTENTION: This makes the collective dataset visible! Is this a problem?

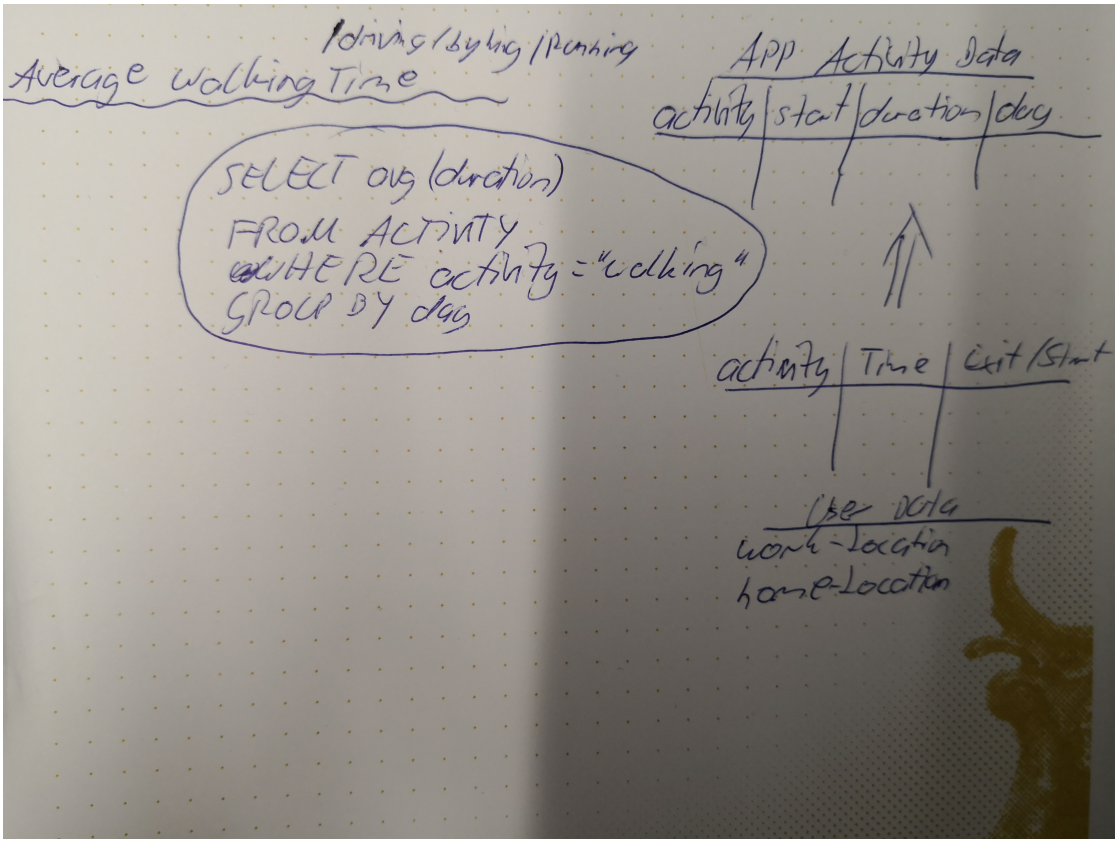
Overall structure of classes:


key
IOHandler
receive(Bytes)
serialize(AggregationRequest)
AggregationRequest
key destination
addData()
executeRules()
AverageDataAggregationRequest
exclude 0 values - flag
CountingTypesAggregation R.
ListingAggregationType
DataType
StartDay
EndDay
AverageStepCount
AverageTimeInActivity
Activity
Roadmap
→ Match trajectories against home/work location
TODO: How many people goto work by car, ...?
combine bike with public transport
↳ Match overlapping trajectories
- time to work
- time at work → Find if - and outgoing trajectory

Average step count



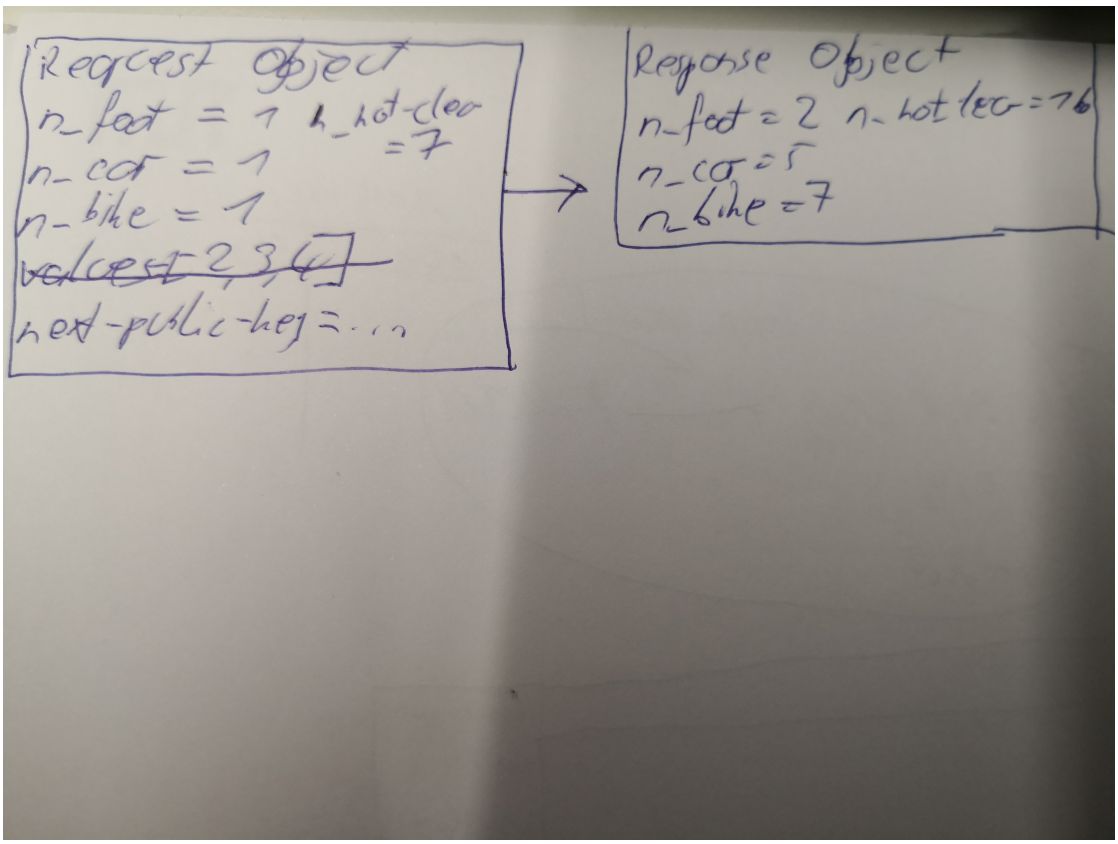
Average walking, driving, ... time



For each of the activities [walking, biking, driving a car, driving public transport] a request is emitted to compute average data. The request can further for each activity exclude or include 0-value-computation. Other solution: the 0-values are counted and the average / median of the other data (only in case).

How many people go to work by car, ...

Search trajectory database for trajectories starting at the home location and ending at the work location and infer activity from other sensor data. If there are trajectories matching home and work location, each trajectory is treated as a car trajectory, if there was only car and walk activities within the trajectory. As bike trajectory, if there was only bike and walk activities and walk if there was only walk. Each trajectory home-> work or work-> home is treated as one entry for counting. If none of the former matches, the trajectory is counted as not_clear. Need sophisticated methods to determine work time, ...



How many people combine bike with public transport

select everybody where activity biking and activity public transport are close to each other. As we cannot map public transport, we will use the driving activity in general.

Create a road map

The Request Object consists of a list of 2 location pairs associated with an activity. Only pairs with a spatial difference of less than XX meters are taken into account. Aggregate all trajectories of the users longer than xx meter, cap the ends and the start for 50 meter, put similar trajectory coordinates together and publish the whole map. A users data is only included, if in a radius of xx there are xx more –> k-anonymity. Create a car / bike / walking map or color-code the map.

Compute the average speed (taking daytime into account) for roads

Filter for gps points that happen during driving or biking. Determine the speed 1. by the speed sensor measurement and 2. by the subsequent gps points. This can e.g. be used to identify roads where zone 30 will reduce noise and co2 exhaustion a lot, if one can compute, that cars accelerate and then typically rest again, so that an average of 30 would have the same timing result. [TODO: Cite cities, ... where 30 is the limit in citycenter, ...]

Time to work

Compute, how much time on average a user needs to go to work. This requires implementing locally the calculation of a users work and home place from the data (or we could also ask for the data and state that it will be locally only) or combine the two approaches.

Average time at work

How much time do people usually spend searching for a parking spot

Select activity driving while the trajectory circles around the same spot

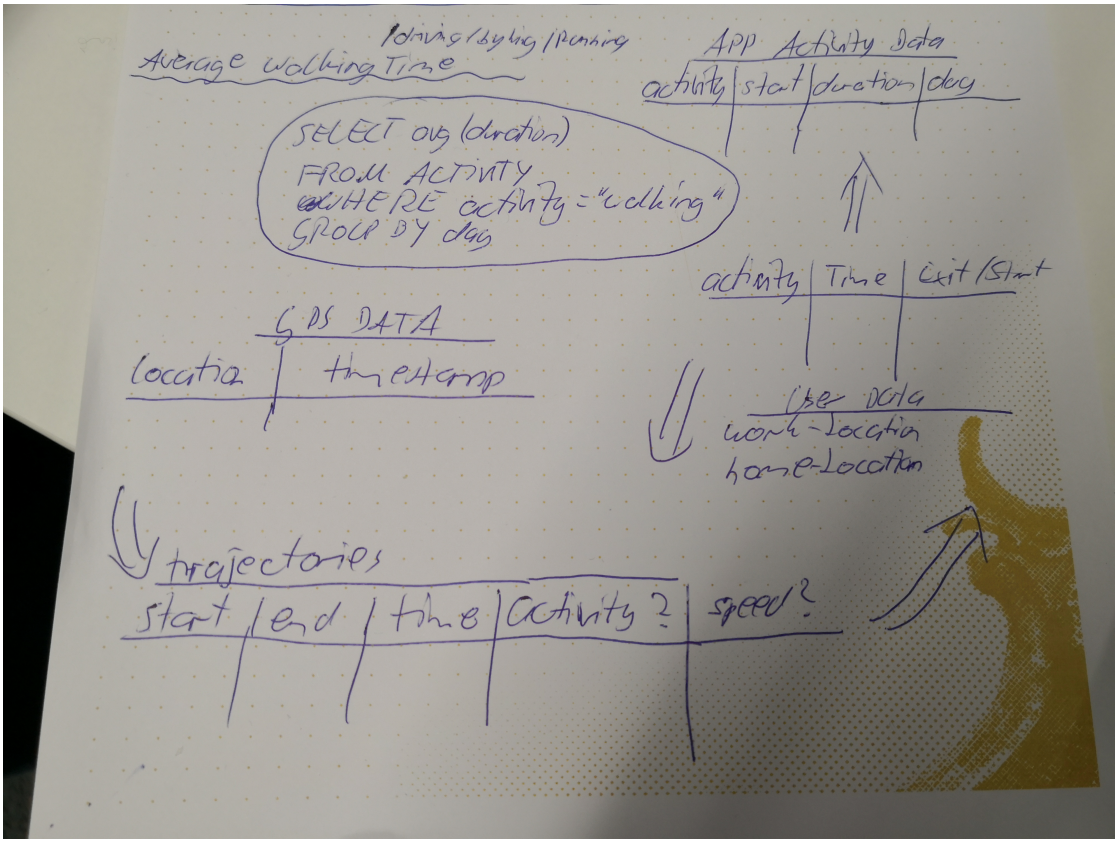
Identify roads, where (maybe even including the current traffic light sequence) one would be faster with a bike (thus average speed of 20, 22, ...)

Identify routes that are faster by bike than by car

Stage 2: When moving on a specific road, report bad traffic

We will fake this in a way that when somebody is driving (recognized activity), we will randomly trigger the application to send an alert. This way we do not have to implement downloading the standard speeds for a road map, recognizing a user being on this road and then checking for less than usual speed.

7.2.3 Data Models



- `CREATE TABLE steps (`
  `day TEXT,`
  `steps INT`
  `);`
- `CREATE TABLE activities (`
  `day TEXT,`
  `activity INT,`
  `start TEXT,`
  `duration INT,`
  `FOREIGN KEY (activity) REFERENCES activity_type (id)`
  `);`
- `CREATE TABLE location (`
  `longitude REAL,`

```
lattitude REAL,
speed REAL
);
```

- ```
  CREATE TABLE gps_data (
  location INT
  timestamp TEXT,
  FOREIGN KEY (location) REFERENCES location (id)
  );
  ```

- ```
  CREATE TABLE trajectories (
  start INT,
  end INT,
  time_start TEXT,
  time_end TEXT,
  activity INT,
  FOREIGN KEY (activity) REFERENCES activity_type(id)
  FOREIGN KEY (start) REFERENCES location (id)
  FOREIGN KEY (end) REFERENCES location (id)
  );
  ```

7.3 Implementation

7.3.1 Android Application

We will base our implementation on [35]. [TODO: quit battery efficient] They provide the open-source code of an application logging a devices location at regular intervals to a file. We will adapt the logging to use json format and adhere to the standards of schema.org. We will furhter use reverse-geo-coding to determine the postal-code, city, region and country of GPS coordinates. The format used will adhere to the standard of [41]. For reverse-geo-coding we will use the google-api: [19]. The rsa public private key encryption will follow the example of [43]. The raw data on the mobile device will either be stored using sqLite or using googles room:

8 Conclusion

TODO: Hint, that our approach can be easily integrated in other applicaitons like open maps projects.

Relate to open street maps project introduced in [31]

Performance: How many data is needed for each mobile? Limit to send over WIFI only!

8.1 Limitations

It has to be calculated, until which extend the method of pre-filling the requests with fake data will still enable a precise calculation of the actual data.

9 Paper Summaries

[31] is one of the first investigating privacy issue in location data. For inferring the home location of a set of car travelling traces of their research subjects, they identify taking the last destination of the car before 3 pm as the most successful among 4 algorithms / heuristics to determine a persons home location. They where able to identify 12.8% of the users home coordinates. Furthermore by looking up those home coordinates on a free online tool, they are able to retrieve the correct name for about 5% of the subjects. Nevertheless, those results could be improved by far, as they show that the used data source / white pages are outdated. In order to protect anonymity, they mainly identify the following different countermeasures:

- Pseudonymity: Stripping original IDs from the dataset (by many shown not to be sufficient)
- Spatial Cloaking: Application of k-anonymity by hiding all data points in a circle with the center placed randomly around the actual home address
- Noise: Adding Gaussian noise to each data point
- Rounding: Placing a grid on the location data and mapping each data point to the closest intersection
- Dropped Samples: Completely dropping samples in order to reduce the frequency of the data points.

Of the application of these countermeasures, only spatial cloaking can preserve data quality, while a Noise with a standard deviation of 5km and a grid for rounding with 5km distances is needed, which render the data useless for many applications. On the contrary, they showed how easy it is, to make the final step from home location to actual identity. Furthermore, there analyzis is only based on data covering two weeks.

[15] finds that even when personal data is anonymized thus that names and addresses, etc. are removed, sensitive information can be inferred from the data. In this study it was shown that from call-records in the US the home address and also often the work address of a person could be inferred. They highlight that while adhering to the k-anomymity model proposed by [47] it is practically not possible to publish

datasets that are still of any significant use.

Also [18] highlights the thread that home and work locations can be inferred from anonymized datasets and can in combination with other sources yield even more information about a user. To reduce this risk, they propose "to collect the minimum amount of information needed". In contrary, we want to investigate another approach, so that rich data can still be used and be published in an aggregated manner to let people profit from the data but still preserve privacy.

Another problem that arises is that anonymization algorithms applied to datasets prior to publishing them might yield good results if the location data is in a densily populated area but might perform poorly if the population is only sparse [25].

[25] identify that while privacy algorithms might successfully provide privacy for location data samples in highly frequented areas, but perform poorly and disclose sensitive information for samples in areas with lower traffic frequency. They discuss the problem commonly accepted in research that either the quality of the data becomes poor or useless when applying techniques like k-anonymity [40, 47, 46] or that privacy cannot be guaranteed. They propose a novel algorithm based on time-to-confusion. Thus basically whenever it is possible to attribute two different samples of a dataset with a high probability to the same user, the corresponding sample gets removed from the data-set to be published. This is necessary, as "the degree of privacy risk strongly depends on how long an adversary can follow a vehicle" [25]. In more detail, time-to-confusion also takes into account the entropy information provided by the whole dataset, thus that even when two samples cannot be connected with high probability due to to many possible consecutive samples, analyzing the whole dataset can provide information that actually the possible consecutive samples have different probabilities due to common route choices. E.g. a vehicle on a highway is much more likely to follow on the highway for some more time than leaving the highway. While this information is taken into account, they point out the limitations of their work that when the dataset is matched with street maps, even more samples would have to be remoed to ensure privacy because it will render some former possible consecutive samples impossible due to missing streets connecting them.

[3] introduces the concept of mix-nodes already known from privacy research on a network level (TODO: "copy" related work part of paper "time-to-confusion"). They propose a framework in which privacy is protected through frequently changing pseudonyms. Furthermore they find that similarly to the problem of identifying consecutive samples in [25], the change of pseudonyms has also to be obfuscated in order to provide complete privacy. In contrast, this paper focuses mostly on solving the problem that location aware services that e.g. notify you when you are close to a venue of interest, do not need to have access to your location data at anytime but

can register to events with a mix-node. Thus they register for the venues of interested and only get notified when the mix-node, which is trusted and has complete access to location data, detects a match. One sees straight away, that this again depends on trust of the users on the mix-node. Nevertheless, the proposed solution of mix-nodes and mix-zones analyzed on a sample shows that even using this framework, privacy cannot be provided, especially as here again the entropy provided by the history of the released or somehow collected data-set makes it too hard to obfuscate the consecutiveness of different pseudonyms.

[47] is the current state of the art of minimum data protection. They define a dataset as the commonly understood tables in SQL. Besides the unique identifier used in the table, a quasi-identifier is the combination of several attributes with which a set of entries can be identified. a dataset adheres to the rules of k-anonymity, if querying every possible such identifier returns at lest a set of k different entries. Thus 1-anonymity identifies an entry exactly and provides no anonymity at all. The anonymity problem arises not from the dataset itself, but from a combination of datasets, that have the attributes of the quasi-identifier in common. This way anonymous knowledge from both datasets can be linked in order to infer information not intended to be made public. They also highlight, that also publishing the same dataset with different privacy-rules, i.e. different anonymization techniques applied, can result in inferences that reveal the original dataset.

[47] clearly highlights that there are two approaches to hiding sensitive information. One is to restrict queries to a database that might reveal sensitive information. In contrast to this approach, they focus on anonymizing the data already before any access to it. Nevertheless, this is based on the assummption that the data owner knows about any possible quasi-identifier in order to obfuscate the dataset sufficiently to provide k-anonymity for all quasi-identifiers. If one quasi-identifier is not thought of, the dataset might expose 1-anonymity for this identifier and result in possible exposures of data not intended to be public.

[47] also discusses further problems that are easy to tackle but nevertheless necessary to protect users' privacy. The order of the published table must be random. Otherwise there is more information (hidden) available that can be used to break k-anonymity. Another problem is when the same table is released and obfuscated differently for the same quasi-identifier, other attributes in the releases can be used to link entries and thus de-anonymize the data.

[18] further investigates the fact that from a dataset containing GPS data of trajectories or e.g. twitter-posts as in [54] the home location can be inferred with high probability. They show that also the work location can be identified with pretty high accuracy and probability. Furthermore they find that people who live and work in different regions or more generally, the further work and home diverge, the smaller the anonymity set

of the specific user in the dataset and thus the lower also the anonymity. This is similar to the findings of [3] that users in less populated areas are exposed to more privacy risk than in denser areas.

[4] extends the analyzis of [3].

TODO: Cite approach of disclosure algorithms by [21] TODO: Cite confusion approach similar to [25] by [23] TODO: Read [48]

9.0.1 Approaches to avoid central datasets

[28] addresses the possible solution of p2p communication instead of using a central instance. They state, that mobile p2p communication is mainly based on WIFI and bluetooth. They propose a middleware embedded on top of the android operating system to facilitate widespread use of p2p. However, those p2p networks are so far limited to devices close to each other locally, as it works over WIFI or Bluetooth and so far there is no established approach to connect smaller local p2p networks over the internet to completely stop relying on central server instances.

[29] investigates the problem that in crowdsourcing (with real humans) the reported results, thus the work of the workers allows for inferences about personal information of the (anonymous) worker. To achieve this, they use decentralized computation while "guarantee[ing] security of the proposed protocol". They use as an example the case where a map of publicly available automated external defibrillators is generated through crowdsourcing. Through the reported location data, the privacy of the "worker can be invaded". The state, that so far, only a few have investigated the privacy problems in crowdsourcing. The use a "sum protocol" where sums can be added in an enccrypted way and then be decrypted finally. This is based on the work of [13]. [34] also find, that this sum protocol is the only way (in crowdsourcing) to guarantee privacy, as otherwise messages between workers would have to be exchanged (our telegram, etc. approach.) In this paper, two other papers are mentioned, that deal with aggregating data while preserving privacy: [7], [42].

[44] motivates decentralized data storage and anyalysis as well, though stemming from another motivation. In IoT, bandwith limits require devices to perform decentral analysis and only forward aggregated data to a central database or to aggregate data in a completely decentralized manner. Others, that deal with decentralized analysis are [5] and [50]. Furthermore, similar to our motivation, they find that "Especially sectors that deal with highly personalized information, such as healthcare, require according means for the secure and privacy-preserving processing of data". Apparently also [49] and [45] find, that inferring information from data (not location data in this case) is possible. (TODO! read the papers if its correct). Another related source for distributed analysis is [14]. They define the term "fully decentralized" as where information is

only exchanged with local peer nodes. No coordinator is needed thus. Also this paper highlights, that the centralization of data at one single point poses a high risk in case of data theft and furthermore is a single point of failure. They quote from [8] that the most power intensive thing in smartphones is receiving and sending data. For vertically distributed data, as it is also the case for our mobility data, they find five different modi of data analysis:

- Central analysis
- Local preprocessing, central analysis
- Model consensus
- Fusion of local models: Each local node builds its own model. Those models are transmitted to a coordinator and fused to a global model
- Fusion of local predictions: When a prediction is made, predictions of the single nodes are collected and combined into a consensus

[24] also tackle the problem that through to "inference attacks" users can be reidentified even if the location data are anonymized. Especially home locations are easy to infer in suburban scenarios where one person / family belongs to one address. Such sensitive information as home, medical visits, etc. might be inferred. The are also aware that using geo-coded address databases it is easy to infer a person's identity from it's home location. The further quote [12] (TODO!!) who state that for obtaining "traffic flow information" it is sufficient, if 5% of all traffic participants send data. To ensure anonymity of data owners, they propose a system in which a trusted third party interacts as mediator between the data owners and the traffic server. The location data is encrypted with the traffic servers public key (private, public key pair) and sent to the third party, using a symmetric encryption algorithm. This way the third party server can confirm the identity of the sender and thus prevent fraud, while it has no access to the data itself, which it forwards to the traffic server. This is a huge advance, but still it is based on a trusted third party server. Also it depends on savely storing (and putting) the symmetric keys into the car / mobile device. They further find that it is necessary to sanitize the incoming data on the traffic server because it might be simply wrong or malicious. This is also necessary in our approach, but in general only, if data is sent as a push variant by devices. In a pull scenario, the data could only be compromised, if somebody manages to attack the application itself. And this can still be handled by sending the aggregation request to different groups in parallel and see whether one request delivers outliers. In addition, they are aware, that the data stored at the traffic server still bears a privacy risk if access to this data is obtained. Thus our approach to

totally decentralize the collection and analysis is a huge and necessary step forward. This is especially even more compromising, if one takes road maps into account in order to identify the home locations they find as well. Furthermore, the mapping of a location to an identity is facilitated by white pages like telephone books or real estate records they state. They also report manual inspection in order to map locations to homes which is easy as for most areas there are precise satellite images which help to outrule possibilities. They manage to identify about 27% of plausible home suggestions from a 239 record data set. (Not validated as exact home location is not revealed in the data set used). Even when data samples are dropped so that there is one GPS point every 10 minutes instead of every minute, still more than 16% of plausible home locations can be detected. So data suppresion algorithms do reduce the risk of home identification but only to limited extend.

[36] tackles the common problem through introduction of a "location anonymiszer" and a "privacy-aware query processor" which enables a user to set its privacy to match k-anonymity (k is variable) and also to choose the degree of spatial-cloaking applied to its data. This approach also requires a trusted third-party (the location anonymizer). Mainly the concept works as following: The third party server receives the exact location data of every user and for each data point creates a new data point through spatial-cloaking that satisfies its privacy setting which then is stored on the server. Whenever a location service (note this approach is mostly for the location-services requiring instant or medium delay data) needs a users location e.g. to notify him about near gas stations (private query), the server receives only the obfuscated location data (a region) and answers with a list of possible matches within this region which is than on the end users device matched with the exact location to retrieve e.g. the nearest gas station. The second benefit is that applications can still reach out to users in a specific area (e.g. to send a promotion to all users in a certain area) through sending this request to the third party which then also replies with a list / area of end users that respects all privacy settings. this is called public query. (On page 764 there might be additional sources - TODO!!). They say that closest to their work is [17] (TODO!!!) and [20] (already read and summarized). Though this still poses the problem of a centralized data set at the query processor. Even the data is pseudonymous and locations are only blurred, other papers like [TODO: cite the others] revealed that even when blurring location data, inference of home locations is still possible. Especially as XXX states, when the data considers rather sparsely trafficed areas. Furthermore identification attacks become more and more precise, the more data points are available (bigger time-span). They further prove, that the overhead of sending a list of possible matches as response to a private query is acceptable (if the data privacy setting is not to strict i.e. $k < 50$ and cloaked region size < 64 in one region).

[20] investigates privacy preservation through spatial and temporal cloaking also

using a (third party) middleware and a generated location data set. They furthermore talk about the topic that the participant itself should be anonymous (network identifier). This can be achieved in our setting through forwarding a request instead of adding data or sending it to the database in a percentage of cases. They state that this has been investigated by [9] and also many papers refer to onion routing regarding this topic. Also the message size is always padded in order to hinder inferences. They differentiate location services according to three measurements:

- Frequency of Access
- Time-accuracy / Delay sensitivity
- Position accuracy

There motivation stems from using sensors already installed in cars instead of separately equipping roads with sensors. This highlights also that our approach is not only limited to location data itself but can be generalized (though usually all information is related to location and time). They state that that for traffic / road information usually high accuracy of the information is not necessary and also a few minutes of delay are tolerable. This highlights, that our approach is also feasible for traffic data and not only pure historical data. They make the example that e.g. harsh breaking data might be used to infer bad road conditions and warn other traffic participants. Regarding this data, huge delays are acceptable as usually a dangerous crossing stays dangerous for a long time (if not forever). They also state that for retrieving nearby points of interest, the time accuracy is high, thus results are needed instantly, but location accuracy does not need to be high. This is in line with the findings of [36]. Additionally, they classify threats to privacy into two categories:

- Sender anonymity (which can be tackled by us by forwarding with a certain probability)
- Inferences of repeated samples (which we tackle by not publishing raw data)

The solve this problem by using a (third-party) anonymity server which on the one hand acts as a mix-node and thus solves problem one (also by pseudonyming the data) and also applies cloaking in order to solve the second problem (partially as shown in other papers). They use the definition of anonymity according to k-anonymity (page 35) [Good definition]. They follow an approach that takes time and space into account. If not enough users for k-anonymity where in a certain area, they either apply spatial cloaking (making the data less accurate) or temporal cloaking (gathering in a certain area for a longer time). Also a mixture is possible we think, though then one has to take into account that the same data is not included in different publications. This is

also thematized by them using an example and stated that tuples must not overlap in time and space. They find that in their experminental setting, a temporal cloaking of 70 seconds or a spatial one of 250 meters is sufficient (in their setting!) to provide some feeling about the extends. Definition of security as the successfull attempt of an adversary to retrieve data not intended to be public / "violate anonymity constraints". They also talk about the possibility that a user spoofs the service by fake users. We also will address this problem to limit fake users.

[48] also stresses, that beyond pure GPS based location awareness, nowadays the location can also be determined by analyzing WIFI access points or phone towers. They also refer to [32] and who investigate this topic. Furthermore they state that [11] have investigated when users want to share their location information. [Use this and paper which says it becomes more and more difficult for the user to decide in order to motivate our approach]. They also refer to [27] and [26]. In detail, [48] propose *hitchhiking*, a framework to provide location-privacy while collecting information about the location. Their approach also somehow follows the concept of only collecting aggregated data. Hitchhiking works without middleware. They also highlight, that they apply privacy-masking of the data before the data becomes available to the central servers. They furhter on page 95 cite 8 more sources that investigate location determination through WIFI and cell-tower signals. They emphasise, that they treat locations instead of peple as the main entity of interest. They also cite [38] who apparently states, that chosing the correct privacy setting becomes more and more difficult. They further highlight, that it is important to need trust only for the client device and no middleware, as this is also a single point of failure. They also talk about the authentication problem in case of malicious attacks. They further state that e.g. bus tracking through installed devices in the bus is affordable for big cities but does not work well for smaller less rich communities. Also they highlight that in case of detecting the availability of a conference room, this can also be handled by having a system in the conference room, so not everything has to be handled automatically and by using GPS coordinates. They further find that many applications do not need instant data access but it is ok with some delay.

[23] also tackle the problem of re-identification attacks. Their solution is two manipulate the original data so that the concept of path confusion evaluated by XXX becomes more likely and thus privacy increases, while the intentional data manipulation does not have a significant effect on the analysis results of the data. If two trajectories are close, but e.g. parallel so identifiable, after applying the path confusion algorithm, the paths cross and make confusion possible. They also (implicitly) classify applications into three set, where in the first set data can be processed offline, thus a huge delay is tolerable, in the second for online applications, a slight delay is tolerable and (completing the categorization) in the third set almost no tolerance is acceptable. They

highlight furthermore, that the data about e.g. traffic would be highly beneficial for traffic optimization (usually carried out by the municipalities). Related works are by [1, 2, 37] investigating privacy preservation in general. [10] is an article about privacy in public databases. [30] investigates privacy in data mining.

9.0.2 Infer activities from location data (and publicly available data)

[33] investigates the possibility to label certain times as specific activities (at home, at work, shopping, dining out, visiting, all other) through the use of location data (reduced to the binary variables near restaurant and near store), supervised machine learning and publicly available information of places, etc. They use machine learning and relational markov networkds for it. Also they take transitions into account, e.g. that one does not go from dining out to dining out or also one does usually only dine out a maximum number of times a day. They find that when using data from different subjects, the algorithms can be trained to have an error rate below 20% when labelling activities, in one experimental setting they even achieve an error rate of 7%. This shows, that using open source software to infer activities from location data should be pretty accurate right now (the paper is 14 years old right now) so when we infer activities like walking, driving, on a bus, etc. We can assume a pretty low error rate which is in favor of the correctness of our analysis's.

9.0.3 Crowdsourcing

[6] clarifies the definition of crowdsourcing, especially in contrast to open source (which makes all the components of a product available and free to use - which is especially easy with software and trickier with hardware), presents some of the first big crowdsourcing approaches and draws conclusions. Crowdsourcing is defined as the distributed outsourcing of work to single individuals. Nevertheless, in contrast to open source, the collected and aggregated result will be property of the company, the individuals only get a compensation. The state that following another definition, crowdsourcing applies only, if the result is later on used in mass production. They also talk about the downside - namely that the companies make huge profit through crowdsourced ideas and the bounties distributed to the workers are a very low compensation regarding the success of the company and far less than full-time employees would usually be paid. [This would be different in our approach as it combines crowdsourcing and open sourcing]. Crowdsourcing is indeed not only used for simple but also for complex problem-finding tasks. They highlight, that it is hard to make a living from open source, as it does not pay a share to the contributers. They further highlight the limits of crowdsourcing (especially through the internet) as the typical user is around 30 years

old, English speaking, western world. This highlights that crowdsourcing should be considered very carefully when information other than pure facts should be generated. Also for example when considering a road-map of where people walk, it might show some paths as not used at all, while many older people prefer those paths - but do not carry a reporting device / smartphone.

[51] investigates privacy and reliability in crowdsourcing. The problem identified by them is that the initiator of some crowdsourced classification request might not want to disclose the item of interest. They suggest an approach where the item of interest is shown to the crowdsourcing workers with added noise. This way there is a protection while on the other hand with sufficient workers, the algorithms can still find the actual solution to the posted problem. Crowdsourcing anyway already relies on redundancy to cope with unreliability and errors of the workers. They cite [52] who also investigate the problem that arises when the same data set is made available with different privacy levels and thus allows inferences.

9.0.4 P2P

[16] investigates P2P streaming on Android devices. This paper is mainly of interest because they seem to have managed to estabilsh a p2p connection between Android devices.

As a final learning one can take away that similar as in Linux, a small trustworthy core like our proposed application would solve many problems due to its verifiability.

[53] investigates inference attacks and how they can be protected through to a cloaking set (thus not giving away the exact location). The focus on the semantics in order to make the cloaking set contain at least one of each semantic. Paper is not thaaaat interesting.

[39] deals with the topic of evaluating user's data in mobile crowdsensing. Mobile crowddsensing by the way is a key term regarding my project.The specify that one has to decide between wrong data due to physical malfunctions of sensors (hardware errors) and wrong data as an input by malicious users. They state that mobile crowdsensing can be used for various purposes - to detect air pollution, water quality, road conditions, etc. But it is all related to location data. Furthermore they target the topic of trustworthiness of data. Two approaches are analysed (with one additional hybrid) - either central server rating or decentralized voted rating. They find that introducing some anchors - some nodes assigned 100% trust - improves the rating system. They find that a central repository has an advantage for eliminating wrong users. This brings us to the same conclusion as we find: Without authentication, it is impossible to provide total anonymization / trust.

List of Figures

List of Tables

Bibliography

[1] D. Agrawal and C. C. Aggarwal. "On the design and quantification of privacy preserving data mining algorithms." In: *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM. 2001, pp. 247–255.

[2] R. Agrawal and R. Srikant. "Privacy-preserving data mining." In: *ACM Sigmod Record*. Vol. 29. 2. ACM. 2000, pp. 439–450.

[3] A. R. Beresford and F. Stajano. "Location privacy in pervasive computing." In: *IEEE Pervasive computing* 1 (2003), pp. 46–55.

[4] A. R. Beresford and F. Stajano. "Mix zones: User privacy in location-aware services." In: *IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second*. IEEE. 2004, pp. 127–131.

[5] S. Bin, L. Yuan, and W. Xiaoyi. "Research on data mining models for the internet of things." In: *2010 International Conference on Image Analysis and Signal Processing*. IEEE. 2010, pp. 127–132.

[6] D. C. Brabham. "Crowdsourcing as a model for problem solving: An introduction and cases." In: *Convergence* 14.1 (2008), pp. 75–90.

[7] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos. "SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics." In: *Network* 1.101101 (2010).

[8] A. Carroll, G. Heiser, et al. "An Analysis of Power Consumption in a Smartphone." In: *USENIX annual technical conference*. Vol. 14. Boston, MA. 2010, pp. 21–21.

[9] D. L. Chaum. "Untraceable electronic mail, return addresses, and digital pseudonyms." In: *Communications of the ACM* 24.2 (1981), pp. 84–90.

[10] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. "Toward privacy in public databases." In: *Theory of Cryptography Conference*. Springer. 2005, pp. 363–385.

[11] S. Consolvo, I. E. Smith, T. Matthews, A. LaMarca, J. Tabert, and P. Powledge. "Location disclosure to social relations: why, when, & what people want to share." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2005, pp. 81–90.

[12] X. Dai, M. A. Ferman, and R. P. Roesser. "A simulation evaluation of a real-time traffic information system using probe vehicles." In: *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*. Vol. 1. IEEE. 2003, pp. 475–480.

[13] I. Damgård and M. Jurik. "A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System." In: *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*. PKC '01. London, UK, UK: Springer-Verlag, 2001, pp. 119–136. ISBN: 3-540-41658-7.

[14] K. Das, K. Bhaduri, and H. Kargupta. "A local asynchronous distributed privacy preserving feature selection algorithm for large peer-to-peer networks." In: *Knowledge and information systems* 24.3 (2010), pp. 341–367.

[15] K. Drakonakis, P. Ilia, S. Ioannidis, and J. Polakis. "Please Forget Where I Was Last Summer: The Privacy Risks of Public Location (Meta)Data." In: *CoRR* abs/1901.00897 (2019). arXiv: 1901.00897.

[16] P. M. Eittenberger, M. Herbst, and U. R. Krieger. "Rapidstream: P2p streaming on android." In: *2012 19th International Packet Video Workshop (PV)*. IEEE. 2012, pp. 125–130.

[17] B. Gedik and L. Liu. *A customizable k-anonymity model for protecting location privacy*. Tech. rep. Georgia Institute of Technology, 2004.

[18] P. Golle and K. Partridge. "On the Anonymity of Home/Work Location Pairs." In: *Proceedings of the 7th International Conference on Pervasive Computing*. Pervasive '09. Nara, Japan: Springer-Verlag, 2009, pp. 390–397. ISBN: 978-3-642-01515-1. DOI: 10.1007/978-3-642-01516-8_26.

[19] Google. Accessed: 2019-04-10.

[20] M. Gruteser and D. Grunwald. "Anonymous usage of location-based services through spatial and temporal cloaking." In: *Proceedings of the 1st international conference on Mobile systems, applications and services*. ACM. 2003, pp. 31–42.

[21] M. Gruteser and B. Hoh. "On the anonymity of periodic location samples." In: *International Conference on Security in Pervasive Computing*. Springer. 2005, pp. 179–192.

[22] T. Guardian. Accessed: 2019-04-10.

[23] B. Hoh and M. Gruteser. "Protecting location privacy through path confusion." In: *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*. IEEE. 2005, pp. 194–205.

[24] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. "Enhancing security and privacy in traffic-monitoring systems." In: *IEEE Pervasive Computing* 5.4 (2006), pp. 38–46.

[25] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. "Preserving Privacy in Gps Traces via Uncertainty-aware Path Cloaking." In: *Proceedings of the 14th ACM Conference on Computer and Communications Security*. CCS '07. Alexandria, Virginia, USA: ACM, 2007, pp. 161–171. ISBN: 978-1-59593-703-2. DOI: 10.1145/1315245.1315266.

[26] J. I. Hong and J. A. Landay. "An architecture for privacy-sensitive ubiquitous computing." In: *Proceedings of the 2nd international conference on Mobile systems, applications, and services*. ACM. 2004, pp. 177–189.

[27] J. I. Hong, J. D. Ng, S. Lederer, and J. A. Landay. "Privacy risk models for designing privacy-sensitive ubiquitous computing systems." In: *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques*. ACM. 2004, pp. 91–100.

[28] W. A. Jabbar, M. Ismail, and R. Nordin. "Peer-to-peer communication on android-based mobile devices: Middleware and protocols." In: *2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*. IEEE. 2013, pp. 1–6.

[29] H. Kajino, H. Arai, and H. Kashima. "Preserving worker privacy in crowdsourcing." In: *Data Mining and Knowledge Discovery* 28.5-6 (2014), pp. 1314–1335.

[30] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. "Random-data perturbation techniques and privacy-preserving data mining." In: *Knowledge and Information Systems* 7.4 (2005), pp. 387–414.

[31] J. Krumm. "Inference Attacks on Location Tracks." In: *Pervasive Computing*. Ed. by A. LaMarca, M. Langheinrich, and K. N. Truong. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 127–143. ISBN: 978-3-540-72037-9.

[32] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, et al. "Place lab: Device positioning using radio beacons in the wild." In: *International Conference on Pervasive Computing*. Springer. 2005, pp. 116–133.

[33] L. Liao, D. Fox, and H. A. Kautz. "Location-Based Activity Recognition using Relational Markov Networks." In: *IJCAI*. Vol. 5. 2005, pp. 773–778.

[34] X. Lin, C. Clifton, and M. Zhu. "Privacy-preserving clustering with distributed EM mixture modeling." In: *Knowledge and information systems* 8.1 (2005), pp. 68–81.

[35] G. Logger. Accessed: 2019-04-10.

[36] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. "The new casper: Query processing for location services without compromising privacy." In: *Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment. 2006, pp. 763–774.

[37] G. Myles, A. Friday, and N. Davies. "Preserving privacy in environments with location-based applications." In: *IEEE Pervasive Computing* 2.1 (2003), pp. 56–64.

[38] L. Palen and P. Dourish. "Unpacking privacy for a networked world." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2003, pp. 129–136.

[39] M. Pouryazdan, B. Kantarci, T. Soyata, L. Foschini, and H. Song. "Quantifying user reputation scores, data trustworthiness, and user incentives in mobile crowd-sensing." In: *IEEE Access* 5 (2017), pp. 1382–1397.

[40] P. Samarati and L. Sweeney. "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and specialization." In: *Proceedings of the IEEE Symposium on*. 1998.

[41] Schema.org. Accessed: 2019-04-10.

[42] A. Shamir. "How to share a secret." In: *Communications of the ACM* 22.11 (1979), pp. 612–613.

[43] Stackoverflow. Accessed: 2019-04-10.

[44] M. Stolpe. "The internet of things: Opportunities and challenges for distributed data analysis." In: *ACM SIGKDD Explorations Newsletter* 18.1 (2016), pp. 15–34.

[45] M. Stolpe and K. Morik. "Learning from label proportions by optimizing cluster model selection." In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2011, pp. 349–364.

[46] L. Sweeney. "Achieving k-anonymity privacy protection using generalization and suppression." In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (2002), pp. 571–588.

[47] L. Sweeney. "k-anonymity: A model for protecting privacy." In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (2002), pp. 557–570.

[48] K. P. Tang, P. Keyani, J. Fogarty, and J. I. Hong. "Putting people in their place: an anonymous and privacy-sensitive approach to collecting sensed data in location-based applications." In: *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM. 2006, pp. 93–102.

[49] S. Thrun, L. K. Saul, and B. Schölkopf. *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*. Vol. 16. MIT press, 2004.

[50] C.-W. Tsai, C.-F. Lai, M.-C. Chiang, and L. T. Yang. "Data mining for internet of things: A survey." In: *IEEE Communications Surveys & Tutorials* 16.1 (2014), pp. 77–97.

[51] L. R. Varshney. "Privacy and reliability in crowdsourcing service delivery." In: *2012 Annual SRII Global Conference*. IEEE. 2012, pp. 55–60.

[52] X. Xiao, Y. Tao, and M. Chen. "Optimal random perturbation at multiple privacy levels." In: *Proceedings of the VLDB Endowment* 2.1 (2009), pp. 814–825.

[53] H. Xu, Y. Zheng, J. Zeng, and C. Xu. "Location-Semantic Aware Privacy Protection Algorithms for Location-Based Services." In: *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE. 2018, pp. 1219–1224.

[54] H. Zang and J. Bolot. "Anonymization of location data does not work: A large-scale measurement study." In: *Proceedings of the 17th annual international conference on Mobile computing and networking*. ACM. 2011, pp. 145–156.