Benjamin Weggenmann—Ed. (SAP)
Daniel Bernau (SAP)

Interim Report

Deliverable D6.1

December 15, 2017
PANORAMIX Project, # 653497, Horizon 2020
http://www.panoramix-project.eu

Horizon 2020
European Union funding
for Research & Innovation

Revision History

Revision	Date	Author(s)	Description
0.1	2017-06-20	BW (SAP)	Initial draft
0.2	2017-07-07	BW (SAP)	Updated use case, requirements
0.3	2017-07-09	DB (SAP)	Updated use case
0.4	2017-07-14	DB (SAP)	Initial design
0.5	2017-07-20	DB (SAP)	Requirements coverage and conclusion
0.6	2017-08-03	BW (SAP)	Internal review
0.7	2017-08-19	VM (UCL)	Peer review
0.8	2017-08-21	BW (SAP)	Revision after peer review
0.9	2017-08-28	MW (UEDIN)	Final review
1.0	2017-08-31	MW (UEDIN)	Final version submitted to the EC
1.1	2017-11-23	BW (SAP)	Server and visualization class diagrams
1.2	2017-11-29	DB (SAP)	Use case and client class diagrams
1.3	2017-12-06	BW (SAP)	Prepare for external review
1.4	2017-12-14	MZ (UT)	External review
1.5	2017-12-14	BW (SAP)	Revision after review
1.6	2017-12-15	MW (UEDIN)	Final review
1.7	2017-12-15	MW (UEDIN)	Submission to the EC

Executive Summary

This interim report documents the work performed in the first and second year in work package 6. The goal of WP6 is to produce a prototype that demonstrates the use and benefits of privacy-enhancing techniques, particularly mix networks and differential privacy, in a representative big data scenario. The report gives a detailed description of the illustrated use case, and encompasses the requirements analysis (task 6.1) as well as the initial design (task 6.2) of the prototype.

The specific use case demonstrated in WP6 will focus on taxi trip data which is sent from simulated taxis over the mix network to a central aggregation server. The data will be anonymized by the taxis, i.e. directly at the data source, before its transmission. This specific scenario provides a highly illustrative example of a generic data collection scenario where sensitive data should be collected in a privacy-preserving manner, requiring privacy-protection for both the transmission *metadata* and the actual *data values* themselves. In the demonstrator, we protect the metadata such as source network address and timestamps using the Panoramix mix network. On the other hand, we protect the taxi trip data (pickup and drop-off locations) using geo-indistinguishability, a variant of differential privacy. Finally, the demonstrator provides a web interface and web service to visualize the thusly aggregated data to a data analyst.

For the requirements analysis, we extract requirements based on the use case outlined above. We derive functional requirements from the involved actors and their actions, and specify further non-functional requirements, particularly regarding performance and privacy. Based on the formulated requirements, we devise the initial design of our demonstrator. We divided the demonstrator into three main components: the client that simulates the individual taxis that send their location data over the mix-net, the server that collects the location data and stores it in a database, and a visualization component that provides the aggregated data to the analyst. To further concretize the design of these components, we provide class diagrams and descriptions of the corresponding classes and their methods.

Contents

1. Introduction

The central objective of PANORAMIX is to design and develop a multipurpose infrastructure for privacy-preserving communications based on *mix networks* (mix-nets) and its integration into high-value applications that can be exploited by European businesses. Mix-nets protect not only the content of communications from third parties, but also obscure communication meta-data such as the exact identity of the senders or receivers of messages, through the use of cryptographic relays. Consequently, mix-nets are absolutely necessary for implementing strong privacy-preserving systems and protocols.

The goal of work package 6 is to demonstrate the use of mix-nets in one of these high-value applications: We want to demonstrate the use of mix-nets and differential privacy for privacy-aware data processing in the cloud. In this scenario, protecting the identity of data owners that submit data to the cloud is key to elicit truthful data (e.g. in surveys) and mitigate concerns against participation. As we will outline in higher detail, the selected use-case is involving data on taxi cruises. Thus, our objective is to support private gathering of data from the taxi sensors to the privacy-preserving compilation of real-time traffic maps or other smart city big data with about 1M-5M updates daily.

1.1 Scope and Purpose of Document

This deliverable covers the advancements during the first and second year of work package 6 (WP6). It describes the results of the requirements analysis and outlines the initial design utilized to start the implementation of our use case demonstrator. More specifically, it encompasses the following tasks from WP6:

Task 6.1 (Requirements analysis): The scenario is going to be based on a large cloud provider's infrastructure. The cloud provider hosts data and application for a large variety of customers with mission-critical business applications. The customers are obviously highly reluctant to reveal data. Hence, we apply mix-nets and differential privacy in order to enforce data protection. The aggregate data can be used as an additional business case for the cloud provider who can provide benchmarking and related business services to his customers. We will analyse the requirements with use case-specific industry experts.

Task 6.2 (System design and implementation): We will select the appropriate mechanism from WP3 for differential privacy and apply them either to the database client or database management system. We will use a highly scalable in-memory system in order to address the requirements of today's cloud infrastructure.

According to these tasks, we have performed the following activities:

1. We have selected the specific use case to be demonstrated by our WP6 prototype: The goal is to privately collect and analyze vehicle location data in the cloud. The data owners (i.e. customers) are represented by several vehicles, who send their position data over the Panoramix mix-net to a cloud database for aggregation. After enough data is collected, a data analyst can perform a privacy-preserving visual analysis of the thus acquired data.

While the mix-net provides anonymity for the data owners in terms of network routing, we will apply differential privacy locally at the data source (i.e. customer) to protect the data even against a malicious (honest-but-curious) database operator or data curator.

2. From the selected use case, we have inferred and formulated further concrete requirements for the demonstrator. Based on the use case and requirements, we have devised the initial design and architecture for the demonstrator.

3. We have started the implementation of the prototype of the WP6 demonstrator, based on the taxi data use case as detailed in this deliverable. It currently provides offline functionality (i.e. with static data), supports visualization for the data analyst, and includes a differential privacy mechanism for location data perturbation at the data source.

1.2 Relation to other Tasks and Deliverables

In the third year of PANORAMIX, we will complete our demonstrator and integrate it with the Panoramix mix-net (task 6.3), which is the main outcome of WP4. Furthermore, we will evaluate the performance, usability, anonymity, data confidentiality, and accuracy of the final prototype (task 6.4). This will help future commercially ready systems to tune their parameters. We will summarize the results of the validation and testing, including lessons learned, in deliverable 6.2, which will provide guidance to future adopters of the system.

2. Use Case Description

As part of the PANORAMIX project, SAP is primarily working on the definition, implementation and validation of a use case which relates to a company transitioning its data and business operations into the cloud. An important driver for the cloud business is big data, where large amounts of information are aggregated and analyzed in order to extract value and provide new insights from the processed data.

The demand for data, however, is often faced with the data owners' reluctance to give out their data due to privacy reasons. Depending on the legislation, privacy regulations (such as the GDPR) might further prevent sensitive data from being shared or analyzed. A PANORAMIX goal is to provide European companies with tools to improve their business while protecting their own and their customers' privacy. To this end, we want to apply technical anonymization measures in order to convince data owners to share their data and to fulfil the necessary legal requirements. Anonymization could allow our customers to leverage client data that would previously have been unavailable for further analysis due to privacy concerns. This could give our customers better insights into their business or other activities.

2.1 Overall Goals

In work package 6, we want to demonstrate the use and advantages of the Panoramix mix network in a collaborative (SaaS) application. Such applications typically collect data (e.g. survey answers, sensor data from IoT devices) from a set of predefined (simulated) clients and aggregate those in a database. Due to the sensitivity of the data (e.g. health, religion, business secrets, etc.) it needs to be strongly protected. Still, we want to perform the typical big data type of aggregate analysis on them with reasonable accuracy. The overall objective of WP6 is to equip the database with the necessary mechanisms and connect it to the mix network.

We have identified several stakeholders at SAP whose use cases match this big data scenario where data from multiple sources is aggregated in a database in order to be analyzed. Among others, these include

- anomaly detection for enterprise systems (such as SAP Enterprise Threat Detection),
- evaluation of vehicle telematics data such as position for finding frequent routes, and
- evaluation of customer feedback in surveys or on social media.

In these applications, it is often required to incentivize data owners (i.e. customers) to share sensitive data with data analysts (i.e. service providers) for the sake of enabling new business models. For example, customers of a cloud service provider might be asked to provide feedback on the cloud service provider and may be reluctant to provide negative feedback, since they are dependent on the long-term business relationship.

Another example is benchmarking between multiple data owners, which would potentially provide beneficial performance insights, but could also be abused by competitors. Anonymity removes the link between contributed data and the corresponding data owner. Hence it encourages reporting, free from fear of retaliation. Still, in certain situations like an outstanding small

or large company the data values themselves may reveal the data owner. Hence, we also want data confidentiality in order to protect them as well. Last but not least, we need performance to ensure scaling to the large volumes in a big data cloud scenario.

We have already previously identified the above overall goals anonymity, data confidentiality and performance for our generic big data scenario. In the following section, we present a specific scenario that is illustrated in our WP6 demonstrator. We will then analyse its requirements in chapter 3 and describe the initial design of the prototype in chapter 4.

2.2 Identification of Frequent Taxi Pickup/Drop-Off Locations

In PANORAMIX, we will focus on the evaluation of position data from vehicles. The decision to focus on this use case was made as the underlying business logic is already mature and thus ensures a stable basis for research. Furthermore, stake holder interviews convinced us that this scenario best integrates scale, privacy and business aspects. The scenario is based on data collected from a fleet of taxis in real time. The data contains taxi trip information (e.g. pick-up locations and drop-off locations). The location data is then collected in a database for further analysis. While the data has the potential to realize efficiency gains through real time fleet management and planning, as well as insights through periodic data mining, it is obvious that passengers face a high re-identification risk. Consequently, we want to protect both the identity of the taxis by hiding the source of the collected location (privacy on the network level), and the identity of the passengers by anonymizing the exact locations (privacy on the data level). Finally, a data analyst can analyze the collected and anonymized data, for example, to find the most frequent pickup/drop-off locations (hot spots). The insights on the identified hotspots could be used by the taxi company to improve their service by providing more taxis in those highly frequented areas in (almost) real time without the previous inherent risk of violating customers' privacy and re-identification.

While the illustrated use case is built around taxi location data, it should serve as a generic example for a very general scenario that can provide value for many business cases: SAP previously offered a tool called "SAP Digital Consumer Insight", which basically provided data about pedestrian traffic around businesses (or any other location). It allows businesses to benefit from knowing about consumer demographics and behavior at a given location or point of interest. The insights provided allow users to run better marketing campaigns, improve advertising and services, scout locations and more. Moreover, from a technical point of view, the illustrated use case easily generalizes to many IoT scenarios where sensitive data from multiple clients is collected and aggregated in a central database.

Actors We have identified three actors in the illustrated use case:

- Data Owner ("Client"): The taxis that participate in the scenario. Each taxi submits its current location on each passenger pickup or drop-off.
- Data Analyst ("Analyst"): The analyst whose task it is to identify the most frequent pickup or drop-off locations. He/she is, for instance, acting on behalf of the taxi company that wants to improve their service, or advertising agency that wants to determine where to best place their ads.
- Data Curator ("Server"): The data curator receives and stores the locations from the taxis in a central database. We currently only assume a subordinate role for the data curator in our scenario.

Flow of Events The flow of events is as follows:

1. Collection: For each passenger pickup or drop-off, the taxi sends its current location to the server.
2. Aggregation: When the server receives a location from a taxi, it stores the data in a central database.
3. Analysis: Once sufficient location data has been collected, the analyst can trigger the analysis to identify the most frequent pickup/drop-off locations.

3. Requirements Analysis

In this chapter, we will analyze the requirements for our demonstrator based on the use case described in section 2.2. Before we start with the actual analysis, we give a description of the requirements analysis approach we have pursued.

Requirements Analysis Approach

1. We first analyze the underlying use case outlined in section 2.2. Our main focus is on deriving functional requirements from the actors and the actions they need to perform. If there are additional constraints or prerequisites, we will record them accordingly.

2. We analyze and specify any other overall requirements that might be important for the demonstrator.

3.1 Requirements from Actors and Actions

The use case includes several actors with different goals and desired activities. We will describe them here and subsequently specify the resulting requirements, which are listed in table 3.1.

Client Each client (i.e. simulated taxi) needs to submit their location to the system for collection and further analysis. Therefore, the system must provide such functionality. Furthermore, the clients wish to protect their own and their passengers' privacy, so we demand corresponding measures to prevent their re-identification. Note that the latter is in line with the overall objectives of WP6.

Server The aggregation server (curator) will take the incoming messages with the taxi locations and store them in the database for further processing.

Analyst The analyst wants to query the aggregated location data from the database and visualize it to find the most frequented areas ("hotspots"), that is, where most passengers were picked up or dropped off.

Req. ID	Title	Description	Motivation
CLI-R1	Submit Location	Clients MUST be able to submit their location coordinates to the system.	Functionality
CLI-R2	Network Privacy	The system MUST provide measures to prevent the re-identification of the taxis based on their network address.	Anonymity
CLI-R3	Data Privacy	The system MUST NOT reveal exact routes in order to protect passengers' privacy and prevent their re-identification, as well as their origin and destination.	Confidentiality

SRV-R1	Aggregate Data	Upon receipt of new location data from the mix-net, the system MUST store the data on a central aggregation server.	Functionality
ANA-R1	Visualize Hotspots	The system MUST provide a graphical user interface for the analyst to visualize the aggregated data.	Functionality
ANA-R2	Real-Time Analysis	The average latency between a client sending a location and the analyst being able to include the location in the analysis SHOULD be at most 15 minutes.	Performance

Table 3.1: Requirements derived from actors and their activities.

3.2 Other Requirements

Apart from the requirements that we could directly derive from the actors and their activities, we identified several other requirements that are important for a data collection system such as required in our use case. We group them in different categories as follows.

Anonymity and Data Confidentiality In addition to the privacy needs of the individual actors, we want to specify some overall privacy and security requirements. They are listed in table 3.2.

Req. ID	Title	Description
PRI-R1	Network Privacy	The server (data curator) MUST NOT learn the network source of the collected data. To this end, the client data WILL be routed through a mix-net to the aggregation server.
PRI-R2	Collusion Resistance	The mix-net SHOULD tolerate at least two colluding peers (mix-net servers).
PRI-R3	End-to-End Encryption	The data sent by the clients MUST BE encrypted end-to-end and between clients and the aggregation server.
PRI-R4	Data Privacy	The system SHOULD provide data confidentiality to ensure anonymity at the data-level, in order to prevent re-identification attacks on the taxis as well as the passengers. To this end, the system WILL use suitable differential privacy mechanisms.
PRI-R5	Untrusted Server	The system SHOULD provide anonymity and data confidentiality even in the case of an untrusted server (data curator).

Table 3.2: Additional privacy requirements.

Performance For any system involved in processing data at scale (i.e. “big data“), performance is a critical measure. This applies in particular in IoT scenarios such as our use case, where data is collected from a huge number of devices. Due to the large amount of arising data, it needs to be processed efficiently, which is why we formulate the following additional

performance requirements in table 3.3.

Req. ID	Title	Description
PERF-R1	Mix-Net Through-put	The system SHOULD support a message throughput of at least one million messages per day.
PERF-R2	Mix-Net Latency	The average latency between the client sending a location and the aggregation server receiving the location SHOULD be at most 10 minutes.
PERF-R3	Database Efficiency	Queries on the collected location data SHOULD be performed efficiently, that is, within less than one minute.
PERF-R4	Performance Evaluation	In anticipation of task 6.4, we need to evaluate the performance of the system. Therefore, the system MUST support the measurement of throughput and latency of messages.

Table 3.3: Additional performance requirements.

4. Initial System Design and Implementation

The prototype focuses on simulating the combination of local differential privacy and mix-nets for productive scenario. The simulation has the goal to resemble an end-to-end scenario, where both the original data as well as the sender remain hidden. It is built using an internal Java class implementing the geo-indistinguishability mechanism for a coordinate input and the Panoramix mix network software. The simulation should also be end-to-end encrypted to ensure data cannot be leaked (e.g. through a man-in-the-middle attack).

While the data analyst (i.e. SAP) wants to collect data for statistical analysis of user behavior, the user (i.e. data owner) does not want to reveal any information that could potentially be traced back. Both these needs must be respected according to requirements CLI-R2, CLI-R3 and ANA-R1 (cf. section 3.1).

The use-case has specific requirements that the anonymization infrastructure needs to fulfill to suit productive use. Because the scenario is about many independent IoT-devices (the telematics units in the taxis) whose data should be collected, efficiency is an important factor. The infrastructure should at least be able to simulate the New York city taxi data in real time, equaling approximately one million messages per day (PERF-R1 in section 3.2).[1] However, to be finally considered efficient the end-to-end throughput needs to be quick as well (PERF-R2). Also, a high level of both data and origin privacy (i.e. privacy of the data source) should be granted. These metrics will not yet be evaluated in this deliverable. The simulation should offer an easy way to test the existing components for geo-indistinguishability and mixing end-to-end, meaning that it needs to cover everything from the client sending data to the analysis of the received data on the server. Both the client and the server component are written in Java.

4.1 System Architecture

The WP6 prototype is depicted in figure 4.1 and covers a simple end-to-end scenario with multiple clients.

Figure 4.1: Simulation set-up and dataflow

Here, only two clients are used to allocate more resources to the mix network. Each client reads data from a data source, adds noise to and sends the anonymized data to the mix network. The mix network then processes the message by decrypting it layer by layer. Afterwards, the

[1] https://ny.curbed.com/2017/1/17/14296892/yellow-taxi-nyc-uber-lyft-via-numbers reports about 650 thousand trips per day, which amounts to 1.3 million messages per day.

server collects the message from the mix network and stores it in a local database. The collected data can then be queried to be used for analytic purposes.

Panoramix Mix-net To ensure sufficient origin privacy, three peers own an endpoint in the basic configuration (see requirement PRI-R2 in section 3.2). This provides protection against collusion attacks with two peers. Both clients and the server run on the host machine, while the Panoramix mix network controller and the peers run in separate virtual machines.

Since data is sent constantly to the mix network in the given scenario, an open endpoint is needed at all time. To ensure this, every virtual machine runs multiple peers, with each participating in a different endpoint.

Panoramix offers simple REST APIs, so both the simulated client and the server only need to call those interfaces via HTTP. Thus, there are no specific restrictions for the client adapter and the server adapter. However, as the mix network only encrypts the route from the client to the endpoint outbox which can be retrieved by everyone, an additional encryption layer is needed. For this simulation, the asymmetric RSA encryption scheme was chosen. As Panoramix is currently unidirectional, encryption systems which require a connection between client and server (e.g. TLS) cannot be used.

Data Objects and Representation The collected data consists of the taxi pickup and drop-off locations. In general, we represent location information as pairs of longitude and latitude. Therefore, the database contains a table with longitude and latitude columns where the server stores the locations it receives from the mix network.

For the actual transmission of the location data from the clients over the mix-net to the server, the clients encode the longitude and latitude tuples as comma-separated strings. Each location string is then encrypted with the public key of the server as to provide end-to-end confidentiality, and the ciphertexts are then transmitted over the mix-net to the server for storage in the database.

In addition to the location data itself, the clients optionally include a timestamp of the current time in each message when it is sent. This will allow us to perform performance measurements such as mix-net latency in the final evaluation of the demonstrator (Task 6.4).

4.2 Components

In this section, we describe our prototype components in more detail. Figure 4.2 gives an overview of the components and how they interact. Apart from the Panoramix mix-net and its provided local agent (yellow), our prototype consists of three main components:

1. a *client adapter* that anonymizes and encrypts the location data, then forwards it to the local agent,
2. a *server adapter* that collects and decrypts the data from the mix-net and stores it in a database, and
3. a *visualization component* with a web interface and service that is responsible for visualizing the aggregated data for analysis.

4.2.1 Client Adapter

The client adapter has the task to read data from a specific source, anonymize it, and pass it to the local agent as depicted by figure 4.3.

As the prototype uses the New York City taxi data set as test data, the records must be read from a data source. With the public data being offered as .csv files, the easiest way

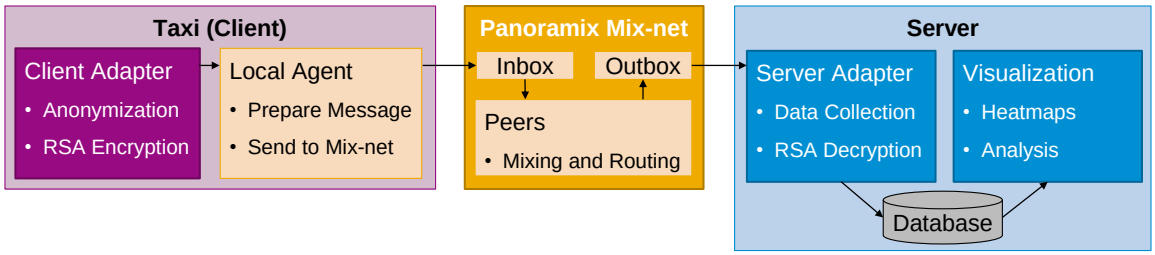

Figure 4.2: Overview of components in our prototype.

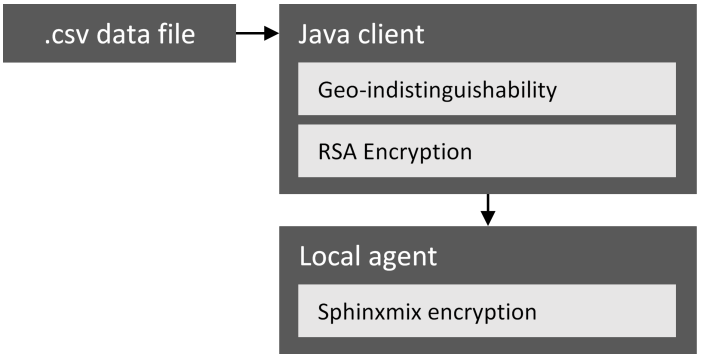

Figure 4.3: Client architecture

to get the data is reading the file row by row. Also, this format is quite lean and does not require any additional software. A database connection can be integrated in later if this is needed for further simulations. The columns representing the latitude and longitude from the read data are anonymized while all other attributes are pruned. Anonymization is realized by instantiating a coordinate object from the coordinates and passing this object to an instantiated geo-indistinguishability class. This class executes the mechanism according to the privacy strength and radius set in the configuration. Afterwards, the data can either be written into another csv file, which can be useful to quickly visualize the mechanism or to check whether the parameters are set correctly, or sent to a Panoramix mix network by using the HTTP API of the local agent. If sent to the mix network, the message content is encrypted before using the public RSA server key. In the prototype, only the mix network output is used. As the procedure to start the agent depends on the Python environment, it currently has to be started manually before the client sends messages. To allow measurements of the latency of the mix network, a column containing the current time can optionally be added before sending the message to the local agent. This can be used by the server to measure the delay of each message.

There is also a class that starts multiple clients as threads to simulate multiple taxis with each running independently.

4.2.2 Server Adapter

The server adapter as depicted in figure 4.4 collects the anonymized data and enables analysis on this data.

As the Panoramix controller currently does not feature any push-functionality, the server must poll regularly to collect messages. In the prototype, the server requests a list of all processed endpoints. If there are new endpoints in the reported list, the server will get those endpoints messages. The messages are then decrypted using the private RSA server key and the location data is inserted into a database table specified in a properties file. In order to evaluate performance, the server can evaluate the latency and throughput by referring to the timestamps that are optionally included by the client in each message, and by counting the total number

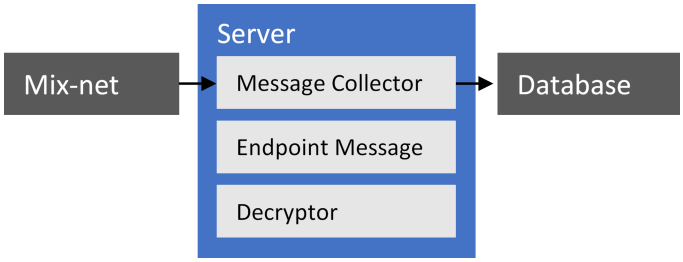

Figure 4.4: Server architecture

of incoming messages per second. As the data is sent as comma separated values, the database table meta data is used to cast the values to the correct type before insertion, meaning that the order the data is sent by the client has necessarily to be the same order the database table columns have. Since we want to be able to analyze the data efficiently (cf. ANA-R2, PERF-R3), we use an in-memory database that can handle large volumes of data efficiently and provides fast query execution.

4.2.3 Visualization

To visualize the data and simulate the analysis stage, we provide a web interface (frontend) and a corresponding web service (backend, cf. figure 4.5) to access the aggregated data. The web interface provides a map centered around the city of New York, where the taxi trip data has been collected. By dragging or editing the bounds of a rectangle displayed on the map, the analyst can select the area being analyzed (cf. figure 4.6).

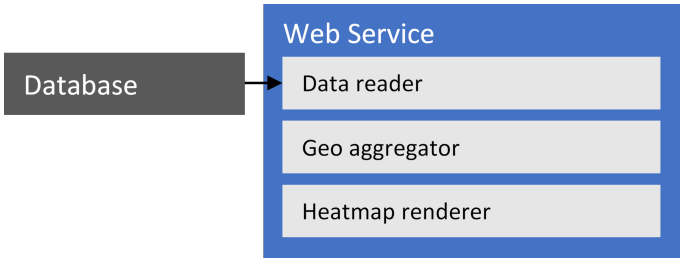

Figure 4.5: Visualization backend

For this area, the web service then generates heatmaps by getting the coordinates from the database and clustering them in a two-dimensional array where each field equals 1 pixel in the output image. When the aggregation is finished, the interface requests the selected cell size of the clusters. This option is included because especially with a relatively low density of points, bigger clusters are needed to improve visibility as shown in figure 4.7.

Depending on the selected cell size, the aggregation is re-calculated on bigger clusters before the final heatmap image is created, rendered and responded to the client. The color of a certain cell is based on the relative position of its value between two deciles. Figure 4.8 illustrates exemplary results.

4.3 Designated Classes and Methods

This section presents the classes that make up each component of the demonstrator. For each class, we list and describe its methods to explain its functionality.

Note that the current design reflects design decisions that have been made to work with the Panoramix minimum viable product (MVP) version that was available in Y2. Since we intend

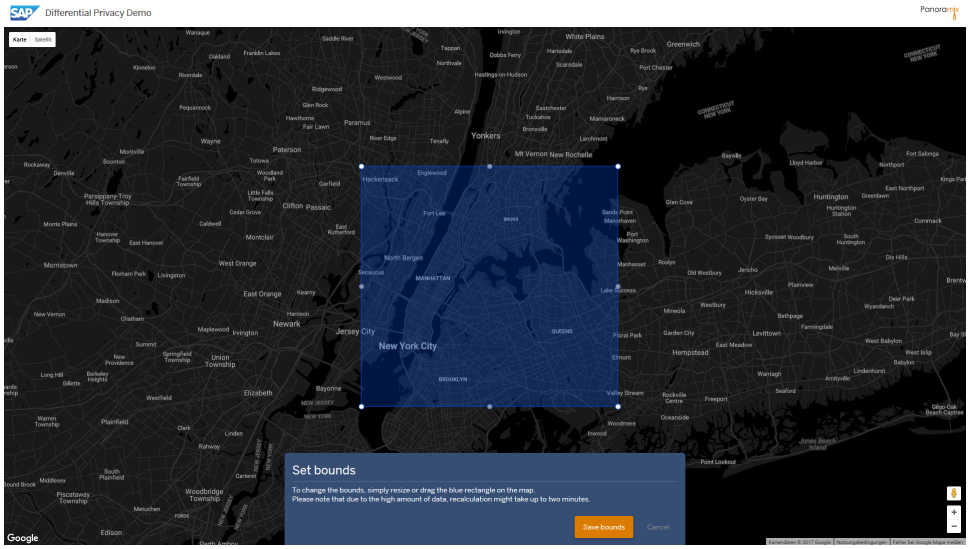

Figure 4.6: Selecting area-of-interest on the map.

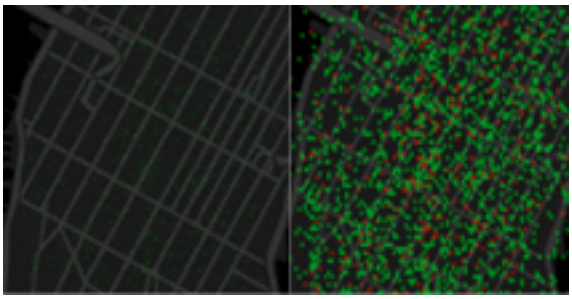

Figure 4.7: Visibility of data points for different cluster sizes

to use a recent version of the Panoramix mix-net in the final version of our demonstrator, we will adapt our design to match new releases of the mix-net framework as needed.

4.3.1 Client Adapter

The client adapter consists of classes that provide encryption and connectivity to the mix-net. Furthermore, the client provides functionality for simulating the taxis with historical taxi trip data.

Encryptor In order to provide end-to-end confidentiality between the taxis and the server, the clients encrypt their locations before sending them through the mix-net. On the client-side, the message is encrypted by the `Encryptor` class. For encryption of outgoing messages the server's public key is used.

getInstance `Encryptor` is designed as a singleton class and consequently returns the reference to the global class instance through this method.

encrypt Encrypts an outgoing message with the server's public key.

MixnetAdapter To actually send messages to a recipient over to the mixnet clients need to instantiate a connection to the mixnet. The `MixnetAdapter` accepts latitude and longitude data from a client, encrypts the data and hands over to the mixnet.

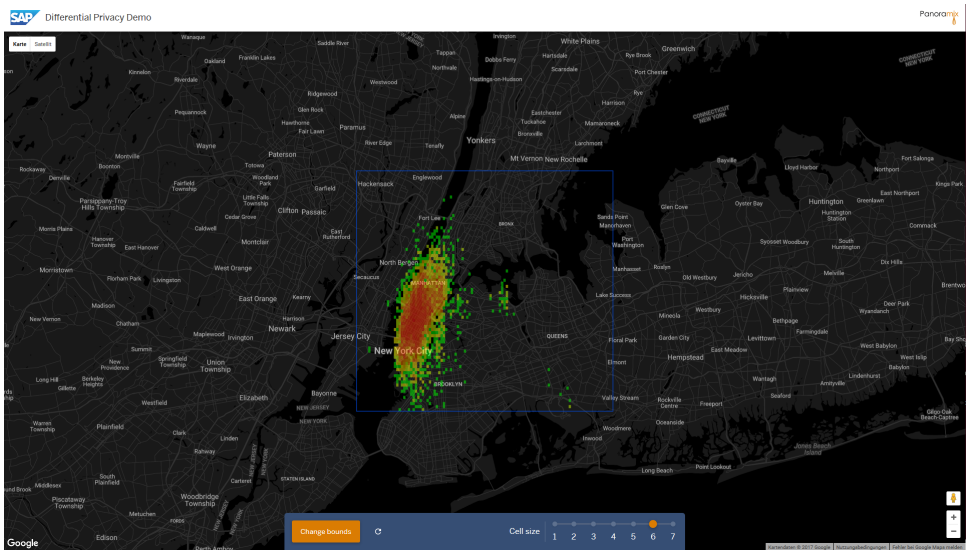

Figure 4.8: Exemplary results.

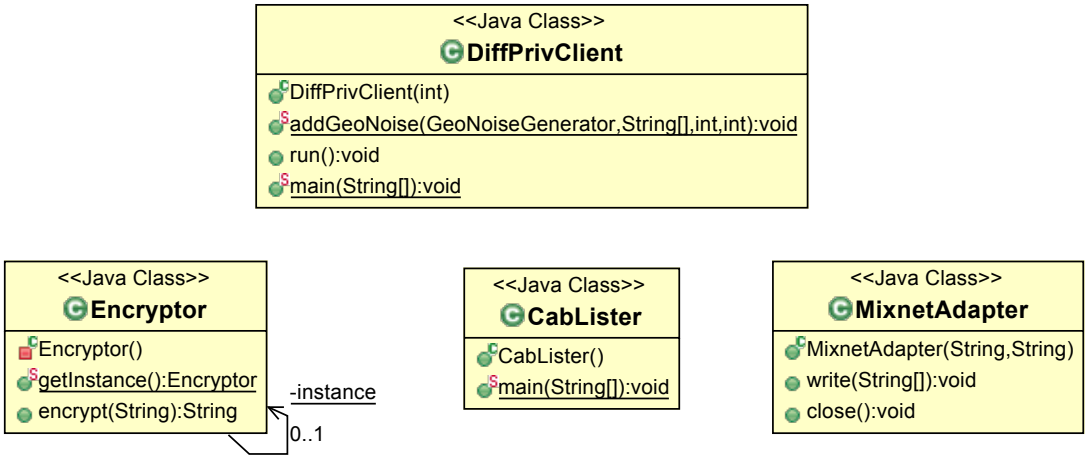

Figure 4.9: Classes providing the client component.

write Longitude and latitude data is transformed into a message and the message is encrypted by `Encryptor`. The encrypted message is posted to the mixnet.

close In case an adapter becomes obsolete due to a change in recipients the local mixnet agent is killed.

DiffPrivClient As we do have historical data from actual taxis we need to simulate the data flow, anonymization and hand-over to the mixnet. For this the `DiffPrivClient` class is simulating a taxi by reading data from a prepared csv file and applying anonymization based on differential privacy to the data. Anonymized data it is handed over to the `MixnetAdapter`.

addGeoNoise Anonymizes original longitude and latitude with a differentially private noise generator.

CabLister Do simulate scenarios with varying configurations of clients and data per client the `CabLister` class allows to split and distribute data to n `DiffPrivClient` instances. Consequently this class represents a critical component for performance evaluation.

4.3.2 Server Adapter

The server adapter consists of classes that collect and decrypt the incoming taxi data from the mix-net, and store the locations in a connected database.

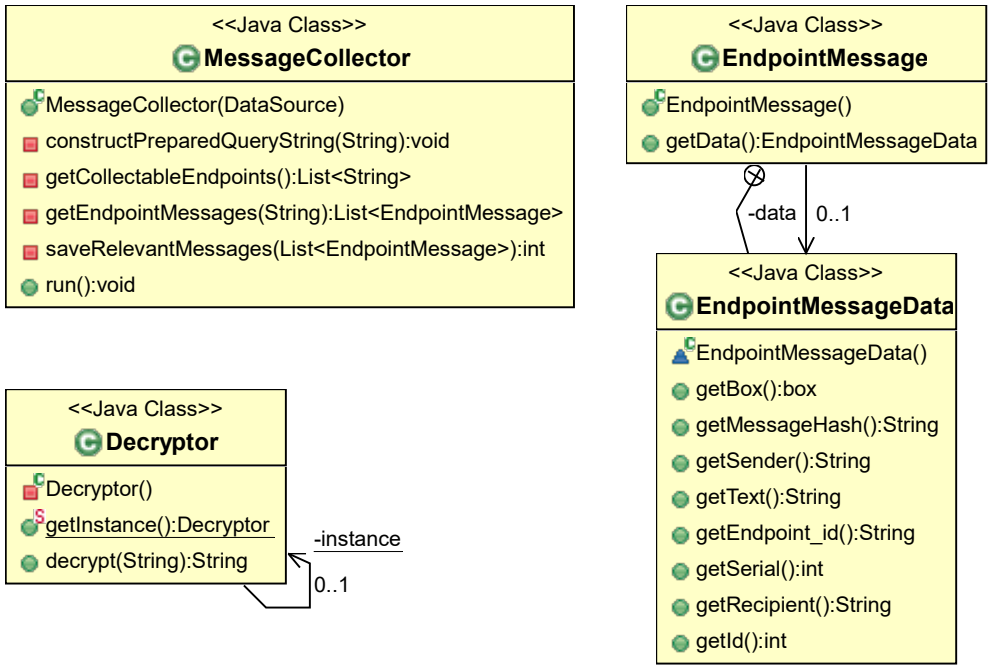

Figure 4.10: Classes in the server adapter component.

MessageCollector The `MessageCollector` polls the mix-net for incoming messages. If new messages are received, they will be decrypted and the location data will be stored in the database.

constructPreparedQueryString Helper method to construct database insertion queries.

getCollectableEndpoints Poll mix-net for yet unprocessed endpoints where new messages can be collected.

getEndpointMessages Query and fetch new messages from a collectable endpoint.

saveRelevantMessages Decrypt newly collected messages and insert the location information into the database. Optionally also compute latency from timestamps when performance measurements should be carried out.

run Create and start a thread that performs the message collection in the background.

Decryptor In order to provide end-to-end confidentiality between the taxis and the server, the clients encrypt their locations before sending them through the mix-net. On the server-side, the messages need to be decrypted; this is implemented in the `Decryptor` class which uses the server's private key to decrypt incoming messages.

getInstance Since `Decryptor` has been designed as a singleton class, this method returns the one global instance of the class.

decrypt Decrypt a given message with the server's private key.

EndpointMessage Wrapper class for mix-net messages.

getData Returns the `EndpointMessageData` object which contains the actual message.

EndpointMessageData Data class representing a mix-net message. Allows easy retrieval of the message contents and its meta-data.

getBox Return the type of box (INBOX, OUTBOX, or PROCESSBOX) where this message is from.

getMessageHash Return the hash of the message.

getSender Return the endpoint of the sender.

getText Return the message contents.

getEndpoint Return the endpoint this message was sent to.

getSerial Return the serial number of the message.

getRecipient Return the recipient name.

getId Return the message ID.

4.3.3 Visualization

The visualization component consists of a web service that provides visualizations of the aggregated locations in the form of heatmaps. These heatmaps are displayed to the analyst in a complementing web interface that connects to the web service to request the heatmap images.

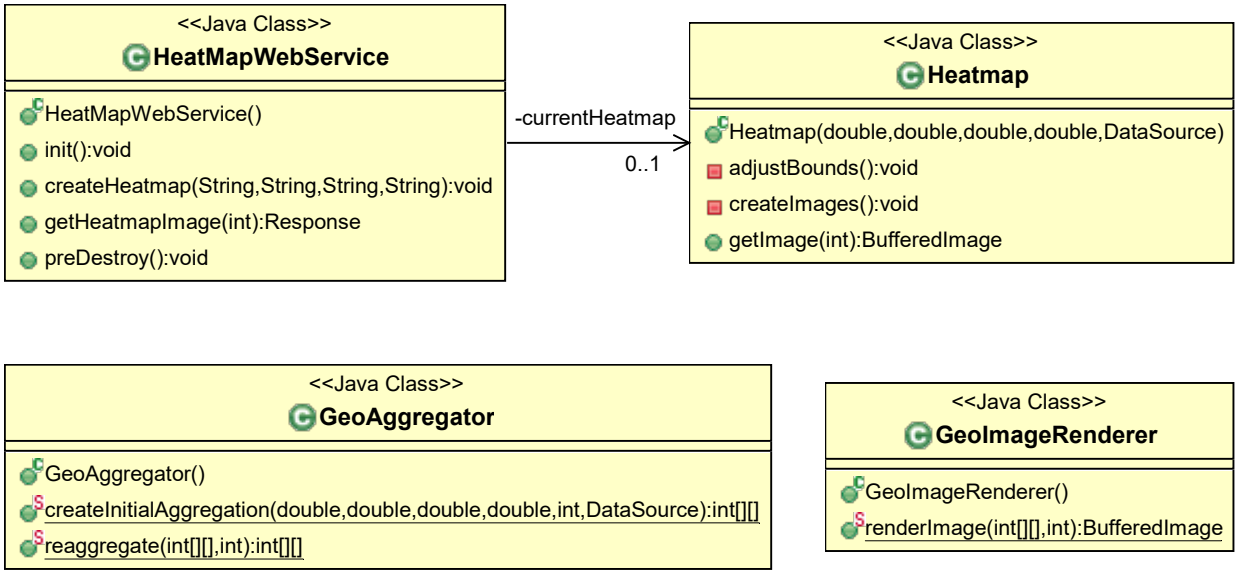

Figure 4.11: Classes providing the user interface.

HeatmapWebService The `HeatmapWebService` is the backend for the visualization component. The service connects to the database to access the collected location data and produces heatmaps for the region as requested by the analyst through the web interface.

init Setup connection to the database where the locations can be retrieved.

createHeatmap Create a (numerical) heatmap of the collected locations within the specified longitude and latitude boundaries.

getHeatmapImage Return an image with the specified cell size that visualizes the created heatmap for display in the browser.

Heatmap The `Heatmap` class provides a numerical representation for the aggregated location data in a certain area that is specified by longitude and latitude bounds. Furthermore, the class provides methods to produce actual heatmap images for this covered area.

Heatmap Uses `GeoAggregator` to construct a new (numerical) heatmap with location data from the specified datasource for the specified longitude and latitude boundaries.

adjustBounds Adjust longitude and latitude bounds to match image resolution and heatmap cell size.

createImages Uses `GeoImageRenderer` to render the heatmap to images with different cell sizes.

getImage Return the pre-rendered image of the specified resolution.

GeoAggregator The `GeoAggregator` retrieves and aggregates locations from the database that fall within a certain area specified by longitude and latitude bounds.

createInitialAggregation Creates a new (numerical) heatmap by counting the number of locations from the datasource that fall into each of the cells between the specified longitude and latitude boundaries.

reaggregate Recalculate heatmap with a different cell size based on previous heatmap.

GeoImageRenderer The `GeoImageRenderer` computes heatmap images from the numerical heatmaps. The images are served by the `HeatmapWebService` to the web interface.

renderImage Render a heatmap with the specified cell size (in pixels).

4.4 Use Case Interaction

To illustrate the connection of actors and components in the identified scenario we created a use case diagram. The diagram depicted in figure 4.12 points out how Data Owner, Data Analyst and Data Curator interact with Client Adapter, Server Adapter, Panoramix mix-net and Visualization. As of the current requirements planning the components are mostly used exclusively by one of the actors, with the Panoramix mix-net sitting in between Data Owner and Data Curator.

4.5 Coverage of Requirements

Table 4.1 illustrates coverage of requirements from chapter 3 by the initial design depicted in section 4.2.

Req. ID	Title	Component	Realization
CLI-R1	Submit Location	Client Adapter	HTTP push to Local Agent
CLI-R2	Network Privacy	Client Adapter, Panoramix mix-net	Routing data through mix-net

CLI-R3	Data Privacy	Client Adapter	Perturbing data using differential privacy
SRV-R1	Aggregate Data	Server Adapter	Message polling and insertion into database
ANA-R1	Visualize Hotspots	Visualization	Generation of heatmaps
ANA-R2	Real-Time Analysis	Server Adapter, Panoramix mix-net	Storing data in an in-memory database, Efficient mix-net implementation (deferred to mix-net developers)
PRI-R1	Network Privacy	Panoramix mix-net	Routing data through mix-net
PRI-R2	Collusion Resistance	Panoramix mix-net	Using three or more mix-net servers
PRI-R3	End-to-End Encryption	Client Adapter, Server Adapter	Public-key encryption (RSA) between client and server
PRI-R4	Data Privacy	Client Adapter	Perturbing data using differential privacy
PRI-R5	Untrusted Server	Client Adapter, Panoramix mix-net	Perturbation at the data source (local differential privacy), Routing data through mix-net
PERF-R1	Mix-Net Throughput	Panoramix mix-net	Efficient mix-net implementation (deferred to mix-net developers)
PERF-R2	Mix-Net Latency	Panoramix mix-net	Efficient mix-net implementation (deferred to mix-net developers)
PERF-R3	Database Efficiency	Server Adapter	Storing data in an in-memory database
PERF-R4	Performance Evaluation	Client Adapter, Server Adapter	Inclusion of timestamps in sent messages (only for evaluation)

Table 4.1: Mapping of Requirements to Components.

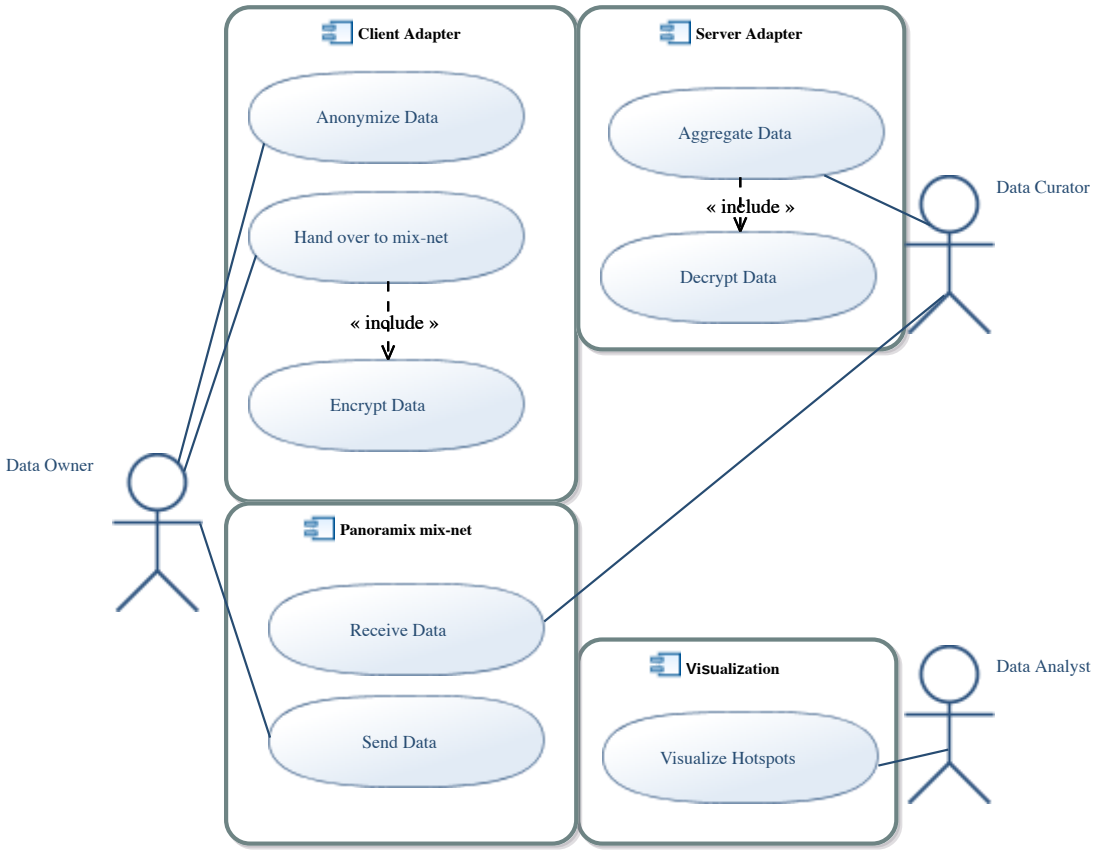

Figure 4.12: Use Case Diagram

5. Conclusion

This report presented the identified PANORAMIX use case for end-to-end privacy, as well as the requirements and initial design for the prototype of our use case demonstrator. The use case addresses privacy-preserving data analytics based on IoT sensors deployed to taxis and a cloud analytics provider, illustrating the requirement to protect both the transmission metadata and the transmitted location data itself.

The scenario combines differentially private anonymization at the data source with mix network protection for data transmission from the data source to the data processor. Based on this scenario, an initial design was formulated. We concretized the design by providing class diagrams and descriptions of the classes that constitute the components of the demonstrator. We illustrated that this design addresses the identified requirements.

Upcoming activities will put this design into live and empirically evaluate certain requirements (e.g. performance) along the identified use case.