Technische Universität München

Department of Informatics

Master's Thesis in Informatics

Privacy-preserving and transparent access control for data queries in sensor networks

Dominik Bitzer

Technische Universität München

Department of Informatics

Master's Thesis in Informatics

Privacy-preserving and transparent access control for data queries in sensor networks

Privatheitsschützende und transparente Zugriffskontrolle für Datenabfragen in Sensornetzen

Author	Dominik Bitzer
Supervisor	Prof. Dr.-Ing. Georg Carle
Advisor	Marcel von Maltitz, Holger Kinkelin
Date	February 2, 2018

Informatik VIII
Chair for Network Architectures and Services

I confirm that this thesis is my own work and I have documented all sources and material used.

Garching b. München, February 2, 2018

Signature

Abstract

The current development and realization of the Internet of Things is leading to a steep incline in the number of connected devices. They promise added value through sensing their environment and using this information in order to react to it appropriately. A currently wide-spread, centralized approach is to send this data to the cloud, where to data is stored, analyzed and can be queried.

The privacy implications of this method raise concerns in potential users, since they lose control over their data. Regularly, data leaks are discovered and unauthorized analysis of data by cloud providers that allows conclusions about the personal lives of the users is disclosed. This shows that without the users consent or knowledge the data can then be used for illicit purposes. In this context, anonymization of data has been shown to not be a sufficient method of providing privacy and might make the data unusable for some desirable purposes.

At the same time, for many legitimate use cases the processed form such as aggregates and averages is sufficient and raw data is not needed. Secure multiparty computation technologies try to fill the gap between useful data analysis and privacy by providing a way of obtaining processed data without the different data sources disclosing raw values. As a disadvantage, their large variety and distributed nature incurs high complexity for data queries. Furthermore a way of access control is needed so that the data sources only provide data to authorized entities.

In this thesis, we will therefore develop a gateway solution that provides a central point of contact for data sinks and data source. Data sinks can make simple queries to a cloud-like interface while the complex SMC protocols in the background are hidden. Data sources on the other hand are provided some assistant functions by the gateway, especially privacy preserving access control.

The solution is based on an analysis of the requirements and a proposed set of concepts and technologies used to fulfill them. A protocol and an exemplary implementation are provided and subsequently their applicability in a real-world scenario is evaluated. The performance evaluation shows the feasibility of the solution and that it provides only a small overhead.

Zusammenfassung

Die aktuelle Entwicklung und Umsetzung des Internet der Dinge führt zu einem starken Anstieg an vernetzten Geräten. Diese Zusatzfunktionen kommen mit dem Versprechen, einen Zusatznutzen zu generieren, indem sie Messungen ihrer Umgebung durchführen und entsprechend auf sie reagieren. Ein momentan weit verbreiteter zentralisierter Ansatz ist es, die Daten in die sogenannte Cloud zu senden. Dort werden die Daten gespeichert, analysiert und können abgerufen werden.

Die Folgen dieser Methode für die Privatsphäre erwecken bei potentiellen Nutzern Bedenken, da sie die Kontrolle über ihre Daten verlieren. Regelmäßig werden bei Cloud Anbietern Datenlecks gefunden oder Methoden aufgedeckt, mit denen sie ohne Erlaubnis Daten auf eine Weise analysieren, die Rückschlüsse auf das Privatleben von Nutzern ermöglichen. Dies zeigt, dass ohne Zustimmung oder Wissen von Nutzern die Daten für unzulässige Zwecke genutzt werden können. In diesem Kontext hat sich gezeigt, dass Anonymisierung von Daten keine ausreichende Maßnahme ist, um Privatsphäre zu garantieren und möglicherweise die Daten für manche wünschenswerte Zwecke unbrauchbar machen kann.

Hierbei ist für viele legitime Anwendungsfälle die verarbeitete Form von Daten (z.B. die Summe oder der Durchschnitt von Werten) ausreichend und Rohdaten sind nicht notwendig. "Secure multiparty computation" Technologien versuchen, diese Lücke zwischen nützlicher Datenanalyse und dem Bedürfnis nach Privatsphäre zu schließen, indem sie den Zugriff auf verarbeitete Daten ermöglichen, ohne dass die verschiedenen Datenquellen Rohdaten offenlegen müssen. Ein Nachteil besteht darin, dass die hohe Vielfalt und die verteile Natur hohe Komplexität für Datenabfragen verursacht. Ferner ist Zugriffskontrolle notwendig, damit die Datenquellen nur autorisierten Parteien Zugriff auf Daten gewähren.

In dieser Abschlussarbeit wird daher eine Gateway-Lösung entwickelt, die eine zentrale Kontaktstelle für Datensenken und Datenquellen anbietet. Datensenken können an eine Cloud-artige Schnittstelle einfache Anfragen richten, während die komplexen SMC-Protokolle im Hintergrund verborgen sind. Für Datenquellen werden hingegen einige unterstützende Funktionen von dem Gateway angeboten, vor allem Privatsphäre-schützende Zugriffskontrolle.

Die Lösung basiert auf einer Analyse der Anforderungen und Vorschlägen für Konzepte und Technologien, die diese Anforderungen erfüllen können. Es werden ein Protokoll und eine Beispielimplementierung vorgeschlagen, woraufhin ihre Anwendbarkeit in einem realitätsnahen Szenario überprüft wird. Die Auswertung der Leistungsfähigkeit zeigt die Anwendbarkeit der Lösung, und dass diese nur einen geringen Mehraufwand verursacht.

Contents

List of Figures

Chapter 1

Introduction

One of the biggest recent development in IT is the steady movement towards an Internet of Things, which means the introduction of interconnected devices to various areas of life in homes and offices. A less obvious trend for customers is the inclusion of such devices in industrial machines like gas turbines or vehicles such as trucks. This current trend is fueled by falling prices of embedded systems hardware and the manufacturers' proposition of additional services provided through these "smart" products.

A central feature of these services often is some form of analytics function, for which embedded devices can include various types of sensors. Temperature, lighting conditions or also noise levels in a factory may be measured. In the context of machines or vehicles, different indicators of wear and tear or safety indicators can be evaluated. In such contexts, the validity of information might be critical for the life of users and anomalies should be possible to detect.

Figure 1.1 shows the cloud architecture, that is typically used for data processing as of today. Information from distributed embedded networks (green entities) is sent to some central system (cloud symbol) in it's raw form (red circles). The central system then stores it and executes analyses. Upon request by data sinks, the gateway then provides the information (blue circles).

This architecture has several drawbacks, among these are privacy implications, lack of transparency (e.g. regarding purpose and data usage) and lack of accountability how data has been used. In the context of security-critical data, person-related data or data that could be related to persons, this is a serious drawback. Due to the growing amount of information that is collected about each person, these effects of these shortcomings will become even more serious and might hinder customer acceptance of such devices.

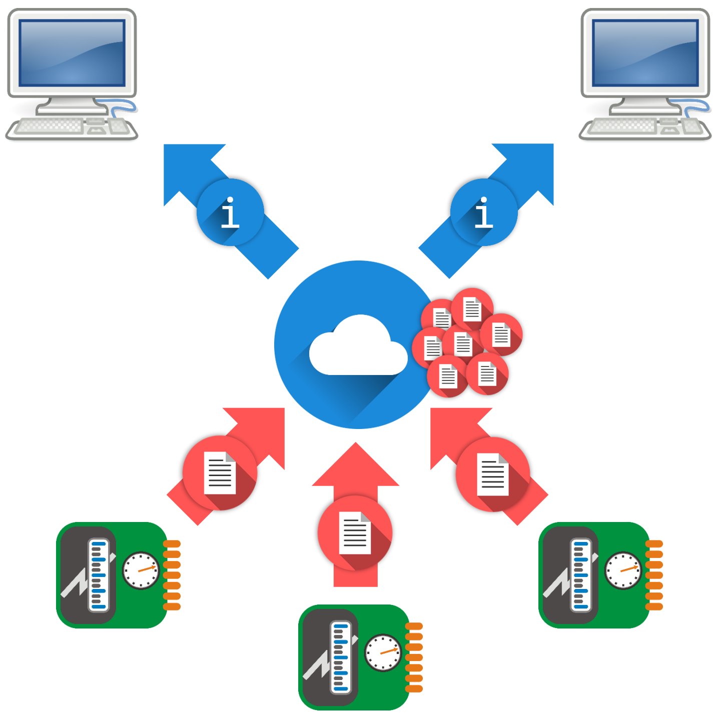

Figure 1.1: Architecture: cloud

On the other side, several methods of privacy-enhanced analysis methods have been proposed, which involve the distributed processing of data without disclosure of raw values. While this approach has many advantages regarding security and privacy, it increases the architectural complexity and the management overhead. This complexity stays visible for the data receiver, as it has to cope with querying data from a distributed system instead of a single point of contact.

1.1 Goals and research questions of the thesis

We aim for a solution which has the advantageous privacy properties of a distributed system combined with the management and organizational benefits of a centralized solution. The goal of this thesis therefore is to design and subsequently implement a gateway solution which enables data sinks to query data available at data sources distributed in a network in a secure and privacy-preserving way.

Data sources should stay in control of their data and be able to account when and how their data is used by requesting entities. Building on the mechanisms of Secure multiparty computation, the gateway should therefore provide sophisticated mechanisms of access control. These should allow data sources to deliberately make authorization decisions by giving them means of specifically describing their access control demands and enforcing them at a fine granular level on a per request basis.

The questions and the scope of this thesis are defined as follows:

1. What are the information security protection goals for privacy-preserving access control in decentralized systems in contrast to older definitions and understandings of privacy?
2. How can these goals, such as accountability, be fulfilled in communication between not fully trusted participants and interest groups?
3. What are the properties of a technical solution to these problems?
4. How can this complexity and the technology of this technical solution based on distributed systems be hidden from data sinks in a general way in order to offer simple access to services as in classical centralized systems?

While this idea intuitively applies to the central asset of the system, the raw sensor data, this understanding of privacy can comprise all aspects of an information system. These might be metadata generated by the system such as frequency of requests, derived data such as uptime of data sources or logs. Since the privacy of the various participants might be contradictory to other requirements, the thesis focuses exclusively on the privacy of data sources.

As the *practical result* of this thesis, a protocol will be designed and implemented in a prototype solution. This comprise a complete system that can be used for secure and transparent querying of sensor data. In order to show the usability of the developed solution, an example setup will be deployed on a small computer network.

The *conceptual result* of the thesis is the present documentation of the chosen approach. The answers to the research questions will be investigated as follows: In a first step, the problem will be analyzed, the requirements for a possible solution will be deducted and using these the state of the art and related work will be evaluated. The developed protocol and the prototype implementation are then checked against the requirements in a conceptual review and an performance evaluation. Lastly, starting points for future research are described.

1.2 Background and context of this thesis

The described goals are motivated by two different research projects, which both exposed the need for such a gateway solution:

SMC as a Service In a former thesis [1], a solution for privacy-preserving collection and processing of sensor data in distributed environments has been developed.

Small and locally distributed data sources sense their environment and collect information about it. They store the information only locally and keep a history of it.

A central entity can gather collected data and perform computations on it (e.g. statistics) using secure multiparty computation (SMC); a technology which allows distributed computation with input data, which has to be kept private and may not be shared itself. This central entity also takes care of all connection management and computation orchestration, so that these privacy-preserving computations can be performed in an efficient, stable and robust manner.

In this context, the proposed thesis shall provide SMC as a service. This means, the whole SMC part and the data sources management shall be concealed and a "normal" client-server interface shall be presented to data sinks which want to query data from the distributed system.

Hierarchical distributed Anomaly Detection System The second use case is based on the DecADe project, that is currently running at the same chair as this thesis is written at. In a distributed embedded network (e.g. in Smart Cars and Airplanes), sensor devices and components collect specific data from their proximity. They are interconnected and data flow between them is generally possible.

A hierarchical Anomaly Detection System (ADS) will be incorporated in these networks in order to detect suspicious and/or faulty behavior. The first level of the ADS consists of the measurement peers themselves which share certain data with their neighbors. Higher levels of the ADS are realized by dedicated components, called "Forensics Centers", which use the information of a predefined set of data sources (or other Forensics Centers of lower hierarchy levels). A forensics center can only query data from the level directly below it, which then in turn collects information from another level further down the hierarchy. Data privacy should be protected by specifically controlling which ADS components may access predefined types of collected information. Furthermore, data accesses should be logged and hence accountability of data usage should be possible. Lastly, it should be possible to revoke access previously given.

Similar to the previous context, data possessing components should provide a gateway logic which enables specific querying of data currently relevant by an ADS component. Again, this gateway has to provide sophisticated methods of data access control and measures for ensuring accountability of accesses.

In both use cases, a distributed set of peers are data sources which collect and hold specific data and its history. We take as premise that this information is privacy-critical and should be protected (which happens differently in the use cases).

1.3 Outline

In order to ensure a successful implementation that fulfills the requirements of practical usability at the same time as gaining new scientific insights, the approach and according structure is chosen for the thesis.

Analysis (Chapter 2): In order to define the functional scope and the architecture of the protocol, the abstract description of the goals is refined by detailed use cases. These describe how the solution could be used in everyday application and what expectations potential users might have regarding the software.

As a specificity of this thesis, application in two different fields are considered. On the one hand this is the querying of information collected in smart buildings, such as energy consumption. The other use case is a hierarchical anomaly detection system.

The use cases describe the information system from a users viewpoint that is not interested in the design of the technological basis. In the analysis, the goal therefore is to convert the use cases in abstract components of the information system that is to be developed and how it interacts with previously given components.

Also, the research question and some basic definitions such as for the privacy definition of the thesis are made.

Requirements (Chapter 3): In order to define the required functionality and deduct a useful design, the requirements are analyzed in detail. This step is based on the use cases and analysis described in chapter 2. Additional requirements might arise from the used platform, previous architectural decisions or usability aspects.

The derived requirements can be split up into smaller sub-requirements, which then are organized in categories and sorted into a requirement catalogue. This catalogue constitutes a list of functional and non-functional requirements.

State of the art (Chapter 4): In order to to avoid redundant research or development, the state of the art and existing solutions are described. An example might be protocols for stateless authentication, that can be used in the context of this thesis.

The existing solutions are listed, summarized and evaluated in regards to their applicability. In the evaluation customizability, performance and simplicity of application are important aspects. If some applicable solutions already exist, the decision for choosing this solution is explained. This analysis will be done for the various required components like authentication and directory services.

Protocol design (Chapter 5): Using the gained insights about requirements and the state of the art, the architecture, protocol and interfaces of the gateway are designed. The single components are described and it is shown how they solve the different aspects of the described requirements.

Implementation (Chapter 6): A prototype for the described gateway and its protocol is implemented in Python 3. The chosen architecture and implementation decisions are explained and documented in order to provide guidance for future development. This gives a more practical guide on how the solution can be used in the described contexts.

Requirements evaluation (Chapter 7): The developed solution is now compared to the requirements that were identified in chapter 3. In this conceptual evaluation different aspects are discussed, such as to which degree the previously described requirements are fulfilled and if the chosen design is a feasible solution for the initial use cases.

Performance evaluation (Chapter 8): In order to ensure good usability and performance, the application of the implemented solution is evaluated using an example setup. This will include aspects such as response times, maximum throughput and performance boundaries.

Related work (Chapter 9): Some related work to the thesis will be described in this chapter. They show alternative approaches for a possible solution to the same or similar problems. Therefore they will be compared with the result of this thesis and their comparative advantages and disadvantages will be explained.

Conclusion and outlook (Chapter 10): In conclusion, the thesis and the findings are summarized, which shows the final context between the chapters.

Apart from the initial use cases, based on the developed and implemented solution new possibilities for extension arise. Such possible future developments and the role of the solution are outlined and starting-points for further research are described.

This provides starting points for future research and extension of the protocol and implementation. In order to provide a good overview about the arising possibilities, they are therefore converted to a set of new research questions.

Chapter 2

Analysis

The introduction and problem statement in Chapter 1 describes a gateway and a protocol for the centralized querying of distributed sensor data on an abstract level. This already gives a large part of the requirements, but is too ambiguous for using them within development. In order to ensure the successful development of systems that are accepted by users, they should be designed according to the needs of these users. Therefore a developer should have a description of the functioning of the desired information system, that is as precise as possible.

The functional scope and architecture are now described in more detail. This step is called requirements engineering in the context of software development and describes a system for the identification of desired properties of the developed artifacts.

As a first step, possible use cases and the desired behavior of the system from a user's point of view are described in detail in section 2.1. Subsequently, the similarities between the different use cases are worked out in section 2.2 in order to have an abstract overview of the desired functions. This is necessary in order to develop a flexible system, that serves several use cases. This is opposed to a system that just provides a large number of very specific solutions to few problems.

Since it has been shown that various groups might be interested in obtaining unrightful access to sensor-data, the security setting will be discussed in section 2.3.

From the results of this chapter follows an abstract description of the system in section 2.4. It includes a description of the purpose of the system as well as the single components that are necessary in order to fulfill this goal.

2.1 Use cases

Two different examples of use cases for the system developed in this thesis are described in this section, which stem from current research projects. Since literature is only sparse, the use cases are identified by considering about the requirements of different groups. The goal is to describe in detail what the system should do. Technical details are deliberately emitted, in order to provide a general and easy description of the problem.

2.1.1 Electricity smart meter in shared office spaces

Use cases can be described with user stories, which is especially well known from agile development methods. In this format, the work-flows of different user groups are described from their perspective. This is especially useful, if no potential users can be interviewed and requirements are collected by thinking from potential users' perspectives. Structuring and identification of the technical requirements will happen in the following chapter 2.

The chosen use case is that of an electricity smart meter in an office space, that is shared between independent parties. Here these are several IT start-ups with only small numbers of programmers and highly ambitious team leads. Since the office floor has an open space concept, the layout of the office might quickly change instead of the classical case of separation by rooms.

In order to ensure acceptance of the collection of data through the sensors, the users affected from the data collection need to be assured that their privacy will not be impacted. The device for the measurement of electricity consumption should therefore offer some advanced connectivity functions and analyses while preserving privacy and strictly enforcing access rights.

2.1.1.1 Billing of teams

An obvious requirement for an electricity meter is the billing of teams for their energy consumption. For this the landlord wants to query the meter about aggregates over a month's usage. The amount of energy usage of separate outlets is not required for this, just the sum over all of the outlets that are assigned to one team.

The landlord is suspicious about one team on the ground floor though, since they have above average electricity usage. The team claims that this is due to their high-end computing stations, which are running around the clock. He is sure that somebody is taking their e-bike inside in order to charge it, which is forbidden according to the house rules for hygienic reasons. The landlord therefore wants to know if the electricity usage is evenly distributed over the team, or if one team member is far above average with spikes during work hours.

The landlord maintains the network infrastructure and has full access to any communication. Also he runs the gateway and has full access over it. In a first variant, the landlord uses their position while adhering to the protocol and therefore does not modify or forge any requests. In a second variant, they might try to use this position in order to modify requests or send forged ones and cover up traces of spying on tenants. The landlord does not have any privileges over the peers though and wants to stay undetected so he won't scare away any of the tenants.

2.1.1.2 Identification of electricity leaks

Since the start-ups are working on a tight budget, the team leads want to cut unnecessary costs. In order to do so, they want to know if there are any energy leaks such as broken power adapters with high overnight consumption or permanently switched on lights.

For this they need an evaluation of hourly electricity consumption, which brings obvious implications for the privacy of team members. This is due to the fact, that the boss wants to find out when exactly significant drops in electricity consumption take place while he is on a business trip. In order to do so, he uses the system's capabilities in order to get information in high granularity.

First of all, this information should only be possible to request by the start-ups for their own domain. Neither the utility provider nor the other start-ups or anybody else should have access.

Secondly, privacy also needs to be preserved within teams. This means, that the boss should not be able to and does not need to see the hourly data per outlet, which can be associated with specific employees.

Lastly, an employee might choose to not take part of this analysis for various reasons. They should therefore be able to reject this request and not provide any data, while still providing data for other use cases like billing.

2.1.1.3 Overall occupation of office space

The process of cleaning the building should work as efficiently as possible while not interrupting teams while they still are in the office. It would therefore be very useful, if the cleaning company starts with the parts that are currently the least occupied. The landlord therefore wants to provide information about the current occupation of the different floors of their office building. This can be derived from the office lighting and electricity consumption.

Again this is a service that gives insights into the teams' work processes. If for example one floor is occupied entirely by one team, they might choose to opt out of this comfort function. A reason for this could be that the cleaning company seems highly suspicious and the team lead doesn't want them to sell any information about the team's workload to the competitor on the floor below.

For the same reason, the cleaning company should only be able to derive this information Monday through Friday at specific hours between 19:00 - 21:00 when they are usually working. Apart from these times, they should not have any interest in this information and therefore don't need to know.

Peers in especially privacy-critical areas, such as bathrooms, should not participate in these requests. Since queries may frequently be rejected or peers may fail, the system should still provide usable results based on the rest of the information.

The team lead actually was right about his suspicions of the cleaning company and now tries to prove to the landlord, that they should hire a new cleaning company. In order to improve their salary, one cleaning clerk offered the competitor start-up from the floor below to sell information about the start-up. Now that the team lead blocked this access, the clerk is worried that they will lose this additional income. They therefore try to request information from the gateway using forged requests.

2.1.1.4 Management of office and outlet setup

One of the challenges in most of the previously described case is the open space concept and teams with high fluctuation and growth. The landlord therefore wants to flexibly assign electricity outlets to teams. For efficiency reason this should work with the highest possible degree of automation and without any physical modifications.

The system is therefore highly dynamic with frequent changes in its devices and setup. In order for users to be able to make requests, they first of all need to know what nodes and what kind of data is available though. If there are any changes in the office, this information should be quickly available and easy to gain knowledge of. Therefore information about all of the currently connected devices and their capabilities in the office should be easy to request.

Such management tasks are especially privacy critical, because the configuration could be modified in a way that private information could be leaked. At the same time, they are not needed on a daily basis and do not need to be available through any device, as e.g. when somebody is working from home.

Access should therefore be granted only from trusted devices in the management office of the utility provider. This means that the authorization decision is not only dependent on the identity of the person making it, but also of the device that it is made through. Such authorization decision processes are currently developing in information security. [2]

2.1.2 Use case: Hierarchical distributed Anomaly Detection System

A primary example of a use cases for the second system described in section 1.2 is transportation. While the context is different, the use case is structurally very similar to the previous one. This section will therefore only briefly describe the use case and discuss the aspects that make up the difference.

In cars and airplanes essentially separate components collect data about their functioning. Other components in the same vehicle may want to request insights that can be won from this information. For example it is used to track the state of the system in its entirety and should provide information about anomalies in order to detect e.g. failure of components. At a higher level a stationary back end may exist, that in turn requests data from the vehicle-side forensics center.

For central collection only the results of computation within such modules is necessary rather than the raw collected information. Additionally, the different physical groups are mostly independent from each other from a functional point of view. For example, the collected information about the landing gear don't have to be set into relation with the steering mechanism in an airplane.

Therefore local processing of data within such groups is sufficient. A gateway such as the one that is proposed within this thesis can serve as a forensic center in this context. It can take care of collection of data and process it close to the source. A central system then directly queries results from the forensic center as a single point of contact.

The difference to the previous use case is, that each forensic center receives raw data from a lower level and then processes it itself. Therefore no SMC-protocol has to be used for actual data queries when collecting data that is aggregated by component.

2.2 Generalization

Even though the described use cases are very different in the type of application field and kind of clients, they have a similar underlying problem. In order to serve both use-cases, a flexible solution should solve this abstract problem instead of solving only the very specific separate problems. The similarity between the use cases are the following:

Data sources, sets of peers that collect and store critical data, need to protect information. At the same time, different data sinks are interested in analyses that are based on this data. The raw data is actually not of interest for the data sinks, apart from serving as a basis for these analyses. According to the need-to-know principle, as few as possible entities should have access to this raw information.

Therefore technical solutions have been developed, that try to solve this problem. These solutions add complexity and make it very difficult for data sinks to request information. In order to make the usage of collected data as easy as possible, the complexity of the underlying technologies should therefore be hidden.

A central point of contact for data sinks is desirable, which serve as a gateway to the data sources as seen in figure 2.1. While offering a simple interface to the outside world, they offer functionality that assist clients in obtaining desired data and take care of management tasks. Most importantly, they translate received requests into technology-specific protocols and then forward them to data sources. The results should then be provided to consumers in a form that is as simple as possible.

As a central difference to the architecture as seen in figure 1.1, the raw data (red circles) stays at the data sources (green entities). Information is only provided in aggregated form (blue circles) after computations (yellow circles) have been executed directly in the data source network.

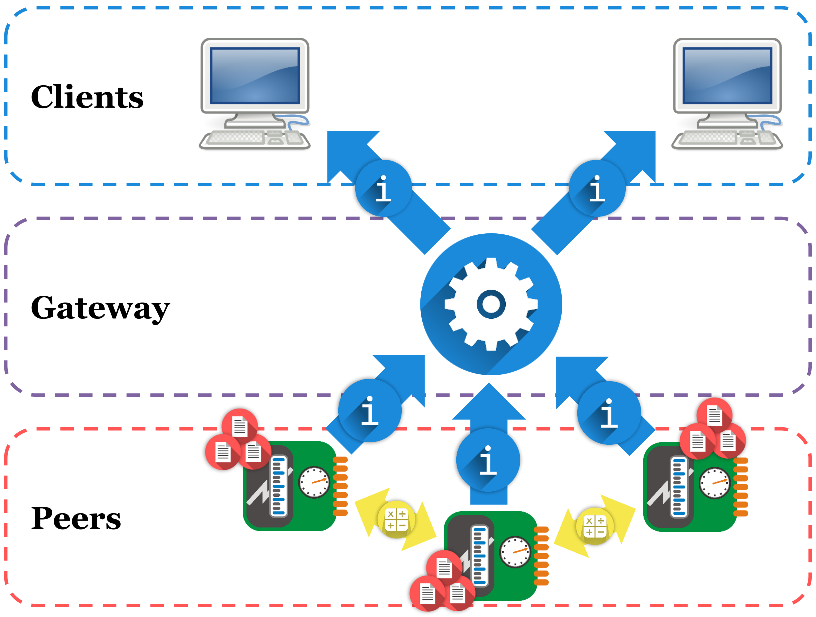

Figure 2.1: Architecture: virtual centrality

Since the collected information is critical, these gateways should obviously not aid unauthorized parties in their goal of collecting information. Therefore they should not only take care of executing analysis and making any desired requests. In order to help protecting the data and ensuring access for rightful consumers, a layer of access control has to be provided.

The discussed primary use-cases are increasingly large and complex, dynamic networks. This is due to the fact, that with the move towards more smart devices new sensors and data sources constantly are added while others are being removed. In order to keep the complexity of the system landscape low, all of these devices should be possible to integrate with the developed system.

2.3 Security setting of the thesis

The use cases have shown real-world applications of a privacy understanding that includes control over data. It now has to be defined, what exactly this understanding of privacy is and how it differs from more established definitions. It will then be refined by specific requirements that it brings with itself.

2.3.1 Adversary models

In section 2.1.1 it could be seen, that different people and organizations were interested in gaining more information than they are supposed to have. There were big differences in the kind of information they want, how far they would go in order to obtain this information and what are their abilities and prerequisites. Since they also come from within different organizational boundaries, their position in the use cases varies from insider to outsider.

It is therefore useful to analyze these aspects for each of the entities in order to have a clear picture of the kinds of attacks that the designed system might face. This will show, that between some of the attackers there are similarities in the traits they have.

Therefore there there will once again be an abstraction of the attackers, in order to gain transferable results. This is an established approach in security research and is called "attacker model" [3].

2.3.1.1 Classification of adversaries

In the described use-cases, the different attackers can therefore be classified in two dimensions. The first is the position of the adversary in the system. The second is the adherence of the adversary to the protocol, or differently said if they are willing to modify communication in a malicious way.

For the position of the adversary, three dimensions were picked:

Global: The models AM.1) and AM.4) describe adversaries with access to any messages sent over the network. They don't have control over any participants other than the messages sent over the network.

Outside: The models AM.2) and AM.5) describe adversaries from outside the network that try to make requests through the gateway.

Inside: The models AM.3) and AM.6) describe adversaries with access to the central component of the network, which is the gateway. Due to the central position in the solution this comprises all requests and messages sent through the gateway, where they are not secured by transport layer security.

Since the thesis covers the secure querying of sensor data through a gateway, comprised peers are not considered. They therefore are assumed to be trusted, where the respective querying protocol has to ensure that malicious peers can be handled correctly.

For the adherence to the protocol, there are two possible dimensions:

Honest but curious: The adversary only work within the boundaries of the protocol and fulfill all of the functions that they are supposed to fulfill.

Malicious: The adversary uses all means they possess in order to gain additional information. They may therefore forge, modify or withhold and messages.

Combining the both dimensions gives us the following possible adversary models:

	Honest but curious	**Malicious**
Global	AM.1) Global passive observer	AM.4) Dolev-Yao attacker
Outside	AM.2) Honest but curious client	AM.5) Malicious client
Inside	AM.3) Honest but curious gateway	AM.6) Malicious gateway

Table 2.1: Adversary models

2.3.1.2 Definition of adversaries

Below are the detailed descriptions of the identified attacker models relevant to this thesis. For each of them their goals and their capabilities are shown together with an example.

AM.1) Global passive observer: As the least invasive of the presented attacker models, the global passive observer is able to record any communication between any peers that is sent over the network. They do not initiate or modify any communication though and only have access to the data as it is transmitted starting from the network interface of a device. Using this information, they want to draw conclusions about the peers and the contents of their communication.

An example for this is the first variant of the use case of the landlord who controls the network infrastructure as it was discussed in section 2.1.1.1.

AM.2) Honest but curious client: The honest but curious attacker uses only their granted privileges and communicates according to the protocol. By these means they still try to gain information that they are not intended to have. This could be possible by combining information from different requests or specifying a valid request in a way that leaks information from the system. An example for this is the curious team lead on a business trip, as they were described in use case 2.1.1.2.

AM.3) Honest but curious gateway: The gateway takes a special position due to the central position it has in the network and the trust that it receives, since it is presented all communication between services and peers. Even when adhering to the protocol, it therefore might use this information in order to make conclusions about both services and peers.

An example for this case once again is the landlord in the first variant from case 2.1.1.1.

AM.4) Dolev–Yao attacker: Similar to the global passive observer, the Dolev-Yao model constitutes some form of global man-in-the-middle attacker. [4] Such an attacker can therefore modify and forge any messages sent over the network in addition to just reading them as the attacker in model AM.1). They might therefore try to trick participants into disclosing private information or comprise the used cryptography.

An example for this is the second variant of the use case of the landlord who controls the network infrastructure as it was discussed in section 2.1.1.1.

AM.5) Malicious client: The malicious client does not adhere to the specified protocol or it's authorizations. It might therefore try to forge messages in all possible ways, communicate with devices it is not supposed to and not collaborate in any way voluntarily. Any information that the client is able to receive this way can be stored and combined.

An example for this can be found with the spying cleaning company in use case 2.1.1.3.

AM.6) Malicious gateway: Same as in the last model, the gateway itself might forge messages and communicate with any devices it possibly can. Additionally though, it can access any information that is not end-to-end encrypted and store, modify or resend these messages at any time. The gateway therefore already has access to a lot of information to begin with and might use it's privileges to enrich it's knowledge by the described methods.

An example for this case is the landlord in the second variant from case 2.1.1.1.

2.3.2 Information security and privacy protection goals

In the usecases in section 2.1.1, several parts of the system were shown to need protection from various potential adversaries. This section will therefore sum up the different kind of protection goals that were identified. The descriptions include examples from the described use cases, how attackers with the various attacker models might try to unrightfully gain access to information.

In all use cases the necessity of confidentiality of different information assets were shown. This goal would be violated if confidential raw data of peers can be accessed by an adversary (AM.1-AM.6).

In order to ensure the correct and secure functioning of the protocol, information and systems need to be secure from modification by attackers. This goal would be violated, if a request that was made by a client is modified before reaching a peer such as in use case 2.1.1.1, therefore requesting additional information under the identity of a trusted client (AM.5).

If an attacker can make fake requests in the name of legitimate clients, they would be able to abuse other entities' authorization rights. Such attacks should be detected and the access to resources should be denied. An example for this would be an attacker with AM.3 or AM.5 such it was described in use case 2.1.1.1, which tries to make requests on behalf of another client.

Since not only intruders but also malicious insiders (AM.2, AM.3, AM.5, AM.6) are possible, the clear attribution of communication to a communication partner is important in the case of abuse. It should therefore be possible to identify the entity making this request, proof that they have made it so that then further steps can be taken in order to penalize such adverse behavior. An example for this would be a highly specific query by selecting various groups in a way that only one peer could possibly fall into the requested category such as in use case 2.1.1.2. Another important aspect is the identification of compromised peers that suddenly change to unusual request patterns.

Continuing the previous example, it has to be proven that the identified entity (AM.2, AM.3, AM.5, AM.6) actually was responsible for the identified request as it was described in use case 2.1.1.2. Otherwise there could for example be no legal means of prosecuting the privacy breach.

As a very central requirement, transparency is necessary in order to make sure that only rightful requests are made, since otherwise the provided data analysis operations might be abused in order to gain additional knowledge (AM.2-AM.6). This requirement was described in section 2.1.1.3, where the tenants want to know the purpose and origin of requests that were made to them.

Another threat is the combination of different sets of data, where each instance of them doesn't bring privacy problems but the combination of them does. It is therefore closely related to the idea of purpose binding. An example for this can be found in section 2.1.1.2, where the identification of electricity leaks should not give hints about the working times of team members.

Lastly, data sources need to be able to reject a requests, if it is identified as a try of breaching privacy and should not be processed further (AM.2-AM.6). An counterexample to this would be storing data centrally, which would therefore not be in the locus of control of a data source any more. An example for this was described in section 2.1.1.2, where an employee should be able to reject the gathering of information about their electricity usage time profile. They should be able to selectively reject such requests, while allowing others such as requests for billing.

2.4 Anticipation of solution components

In this chapter so far use cases and possible usage settings for a system for the querying of sensor data have been described. Technological details or possible architectures were so far left out though and therefore at this point a precise question about the kind of system that should be developed has not been defined yet. In order to structure the loosely coupled use-cases from section 2.1 into an abstract information system, this section will therefore specify the components that are necessary for a gateway in order to serve this purpose.

In the abstraction of the use cases in section 2.2, the need for a central point of contact for the querying of data has been shown. Therefore from now on we assume that a dedicated gateway is installed which separates data sources and data sinks (=clients). The purpose of the gateway is to take care of all management tasks (also protocol-specific tasks such as SMC) and perform the orchestration of connected data sources.

In order to be able to offer the services that were described in chapter 2.1, the following components need to be offered:

2.4.1 Access control (AC)

In the described use cases it had to be made sure, that only authorized entities get access to data. The gateway therefore should carry out a first layer of authorization so that only valid and legitimate requests are further processed (2.1.1.3). This process needs to securely identify the authenticating entity. In other words, they should not be able to access information that is restricted to other clients (2.1.1.3).

To make this decision, it needs to hold authorization information about the clients. In the described use cases, the authorization decisions were not only based on the identity of the requester but actually on various factors, such as e.g.:

Attributes of the requester: An example might be the device used to make the request, such as for example in section 2.1.1.4.

Attributes of the requests: In some cases, the authorization decision might depend on the requests themselves. High frequency requests should for example be treated differently than only sparse requests as in the example in section 2.1.1.2.

Environmental conditions: This is the case of the the time of the request is used in order to make an authorization decision such as in section 2.1.1.3.

Attributes of the requested resource: The authorization decision might depend on the requested resource, such as the criticality of the sensor data that it provides (section 2.1.1.3).

The authorization framework should therefore be able to reflect such information and base it's authorization decisions on various factors at the same time.

Since the system should not rely on trust for the gateway 2.1.1.1, the peers should be able to provide an own second layer of access control and make their decision about requests. This is a concrete measure of ensuring the protection goal of accountability and intervenability. In order to do so, they need to receive all relevant information of a request and be able to selectively reject or accept them without interfering with the correct functioning of the system.

The system was described as highly dynamic with possibly frequent changes in the network topology and devices that are participating (2.1.1.4). Since many requests might fail and have to be retried, this incurs a large management overhead. Due to scalability considerations and in order to rely on a single point of failure, no state information should have to be held on the side of the gateway for the authorization and authentication process.

Clients should therefore receive authorization documents, which must be presented to the gateway when performing a query. The gateway should then be able to decide legitimacy of the request by the presented certificate, which afterwards allows clients to perform a set of predefined queries.

2.4.2 Request processing (RP)

The described gateway doesn't store any raw data and has to obtain it by other means in order to offer analysis functions. Hence, it must be able to interpret the client's request and communicate with the the peers in their protocol in order to provide meaningful results. In order to offer this service in a way that fulfills the boundary conditions, some steps need to be taken in this process in addition to the request translation.

2.4.2.1 Central point of contact

Most importantly, a central point of contact has to be provided to all the systems involved in the protocol. More specifically, this is an interface that accepts requests from clients or peers and then handles coordination between the various gateway components in order to provide simple access to all services and functionality.

Most importantly, this comprises the following steps:

- accept requests
- pre-processing of request, such as authorization checks
- translate the given request into a target protocol
- query data from peers
- post-processing of results
- provide results to clients

2.4.2.2 Request translation

Due to the privacy understanding as it will be described in section 4.1, the request processing should actively support accountability by providing authentication and authorization information to peers and clients. Therefore, as a second layer of authorization on the side of data sources, they should receive information about requests in a way that they can verify and log themselves (2.1.1.3).

They should therefore be able to obtain the original request, understand and track the purpose of it and validate its legitimacy (2.1.1.3). This means, that the gateway has to provide proof that the request actually stems from a client who requested it at the given timestamp (AM.3+AM.5) (2.1.1.1).

The actions of the gateway and the client are then validated by the peers. If some of these checks fail, data sources must be able to reject the request and be allowed to veto against it (2.1.1.2).

Regardless of the acceptance or rejection of requests, peers (e.g. the tenants in the smart building) want to retrieve detailed information about the further processing of queries. This includes information about the purpose of the request, the person or system requesting the information and the time that it was made. This information can be used to identify abuse of privileges.

While therefore removing itself as a single point of failure, the gateway should still validate and certify the legitimacy of requests itself and store all relevant information in appropriate logs.

2.4.2.3 Handling of protocol exceptions

In dynamic environments, peers and communication might frequently fail and require multiple tries until they are successful. The outcome of requests is therefore largely dependent on dynamic groups, peer failures or partially to fully rejected requests. While such cases are also are possible in traditional systems, the probability and frequency is much higher in the discussed use case of sensors. While this might be partially handled by the connected protocol, the gateway should transparently reflect these circumstances in the communication with clients.

So that missing information can be interpreted in a useful way, the processing of requests should therefore provide a way of handling such exceptional cases. If a data gathering action fails, interrupted queries should be addressed by some form of recovery mechanism and a retry should be efficient since not all checks are necessary to be made again. If requests are rejected, there should be a predefined manner of constraining queries while providing at least partial results and transparently communicating information about this outcome.

2.4.3 Directory Service (DS)

Users need to know about currently available nodes and the kind of information that they offer in order to make specific requests 2.1.1.4. In a dynamic environment, data sources and types of data available through the gateway are constantly changing 2.1.1.4.

The gateway should therefore maintain a directory and provide an interface, through which clients can obtain information about currently available data for query through this gateway. In the DecADe context, this functionality is vital, as it allows clients (which can be higher order Forensics Centers) to automatically find out which gateway offers a currently desired type of information.

Most importantly, the gateway needs to provide different kinds of information about the peers in the sensor network, such as the following:

- Generally and currently available peers
- Peers' capabilities
- Peers' metadata such as location
- Groups of peers, e.g. functional such as all kitchen spaces, or spatial such as 1st floor

Since peers might be added or removed at any time, the directory should serve as a point of contact for them. The first contact of a peer with the gateway is called pairing in this context. Afterwards, the gateway should keep track of peers' availability since they might be permanently removed or temporarily unavailable and returning later.

Chapter 3

Requirements

The analysis in section 2 has structured the use cases from chapter 2.1 into an abstract gateway system. For application in the described use cases, it was shown that there are requirements regarding topics such as privacy or scalability. These specific and verifiable requirements will now be described, summarized and classified.

The result of this is a list of functional as well as non-functional criteria of different groups like users, operators or developers that may want to maintain or enhance the software. Each requirement is assigned an unique ID in order to easily describe and identify it in later parts of the thesis. These groups give a useful structure of the mentioned catalogue. An additional criterion for the requirements is their prioritization in comparison to each other.

The analysis and definition of research questions in section 1.1 has shown the focus of this thesis on requirements such as security and privacy. At the same time, the desired functionality of the gateway largely depends on these global requirements. Therefore such non-functional requirements are discussed first, while functional requirements will be listed afterwards.

3.1 Non-functional requirements

In contrast to functional requirements, which describe the directly observable behavior that users expect of the system during regular operation, non-functional requirements define some general boundary conditions, that the designed system has to fulfill. Examples for this are privacy considerations and reliability.

The non-functional criteria for this thesis are described below.

3.1.1 Information security protection goals (ISP)

In section 2.3.2, it has been shown that the system is supposed to protect the data in different ways. In order to systematically examine the necessary protection of information and systems, information security requirements are usually approached using abstract desired goals, such as keeping a certain piece of information secret. As a minimal set, usually the CIA triad of confidentiality, integrity and availability are used, but different additional requirements have been defined in literature. Below are the seven goals defined in the ISO 27000 standard [5].

Since they are of varying importance for the purpose of this thesis, only those that are of particular interest are assigned a requirement ID. Two requirements are not within the focus of this thesis, which is privacy. For example an unavailable system has no privacy implications, which is why the according requirement will not be considered.

The specific implications of these abstract goals on the developed system and its components will be further specified in section 3.2.

ISP.1) Confidentiality: Only authorized entities may gain access to information that they were intended to have.

ISP.2) Integrity: Information and devices are secure from unwanted modification.

Availability: Information or devices are accessible when needed. While an important aspect in production systems, this aspect is not concerned with privacy and will not be in a special focus in this thesis.

ISP.3) Authenticity: Entities can assure through some form of authentication, that the sender of information or commands is actually the communication partner that they are claiming to be.

ISP.4) Accountability: Entities need to be able to securely identify all direct and indirect communication partners that sent information or commands. While ensuring accountability at the gateway itself, also peers should have all necessary information that is needed in order to ensure accountability.

ISP.5) Non-repudiation: An entity can not deny having sent a message

Reliability: The used procedures work reliably and whenever needed. This protection goal is also not a main focus of this thesis, since it is similar to "availability" and does not state anything about the privacy of information.

3.1.2 Privacy protection goals (PP)

While the previously mentioned goals partially address privacy concerns, they are not sufficient for serving as requirements for privacy protection in information systems.

Some, such as availability and reliability, have a focus other than privacy that is not covered by the scope of this thesis and are therefore not applicable. Additionally, a different understanding of privacy has recently been establishing itself, as it will be described in section 4.1.

Therefore additional and more precise goals that come with this understanding have been defined in order to specify the desired behavior of systems that follow the privacy-by-design approach. [6] Below are the three main goals, that will serve as a basis in the design of the protocol and the system that are developed in this thesis. These directly relate to the use-cases that have been discussed in section 2.1.

PP.1) Transparency: Ability to review the collection and processing of data before, during or after such operations take place.

PP.2) Unlinkability: Provided data should not be possible to be combined or processed in a way that makes drawing conclusions about other information possible.

PP.3) Intervenability: Data sources should be able to reject a data analysis request.

3.1.3 Performance in dynamic environments (DE)

Some additional requirements with regard to performance and functionality in such environments have to be fulfilled in order to provide an extensible system as it was described in section 2.2. Below are some central aspects that are necessary for the desired functioning or such a system. They should be fulfilled by all components of the gateway, as far as applicable in their context.

DE.1) Scalability: The system should be capable of sustaining acceptable performance levels even when large numbers of requests are processed by the sensor network.

DE.2) Extensibility: Adding new kinds of sensors and smart devices into the network can mean the demand for additional functionality, which was not planned in the initial design of the system. Therefore there should be efficient ways of extending the protocol while remaining compatibility with the rest of the system.

3.2 Functional requirements

As opposed to the previous requirements, parts of the expected behavior can simply be described by what the system does. These requirements therefore follow the description of the components from section 2.4.

Other requirements are derived from the previously discussed non-functional requirements in section 3.1. These are solved by the different components of the gateway, while each partial solution is referenced to the requirement it fulfills.

3.2.1 Access control (AC)

A central purpose of the gateway is to serve as a first layer of access control, therefore checking if requests only are further processed if sufficient authorization can be presented (2.4.1). The gateway must therefore offer a service, that makes the decisions based on a process as follows:

AC.1) Privacy-preserving access control: The authorization decision process should fulfill the information security (3.1.1) and privacy protection goals (3.1.2). In order to do so, it fulfills some of the goals itself while for the rest it cooperates with the other gateway components. For requests that the gateway receives from clients, it must therefore ensure the following:

- **ISP.1:** granting and rejecting authorization is solved in a way, that doesn't leak any information about the system and network
- **ISP.2:** any authorization information as described in section 2.4.1 is safe from modification
- **ISP.3:** clients (data users) must authenticate before querying
- **ISP.4+ISP.5:** requests and authorization information are bound to an entity in a manner that can be traced back to them in a a way that is possible to prove securely
- **PP.1:** relevant request and authorization data can be presented to the peers, so they can understand the authorization process (also see section 3.1.1)
- **PP.2:** no information about peers can be derived from the access request, since it contains no information that allows a deeper understanding of the state of the peers, beyond what was already provided by the client in the authorization request. Therefore it only states abstract groups for defining the scope of an authorization decision
- **PP.3:** peers are able to make their own authorization decision based on this information and able to reject requests

AC.2) Dynamic authorization decisions: The authorization decision can be based on various factors such as the attributes of the requester, requested resource, the desired operations or environment conditions.

AC.3) Stateless authorization grants: In order to allow efficient retry, clients should be able to proof authorization without having to rely on the gateway to store this information.

3.2.2 Request processing (RP)

The most important reason for developing a gateway is the goal to offer services to clients for gathering information about and from peers at a central point of contact. In order to offer a successful final product, the request processing component therefore needs to fulfill the customers' needs. At the same time, it needs to assist in order to fulfill some of the information security and privacy protection goals.

The process of processing requests should therefore fulfill the following requirements:

RP.1) Efficient failure recovery and retry: Since the peers of the system are distributed in very heterogeneous networks with varying quality of connection or peer failure rates, frequent loss of communication is possible. The system should therefore offer means of efficient failure recovery, in order to provide ways of completing pending requests without major implications for performance.

RP.2) Dynamic job building and execution: Since the members and topology of the networks might be constantly changing, the system should as little as possible rely on the availability or static state of entities. Even if changes in the state of involved systems happen before or during execution of the request, the execution of jobs should still at least be partially possible and the response should reflect the outcome of this job execution process.

3.2.3 Directory Service (DS)

The gateway should efficiently provide information about the current state of the sensor network and it's peers. It therefore needs to offer some directory service, that assists the clients and other components of the gateway. The kind and mode of providing information therefore needs to fulfill the following requirements:

DS.1) Extensible peer information storage: In order to successfully execute requests, the gateway needs to store and provide various data about the current state and metadata about peers and clients. New types of information should be be possible to add (DE.2).

DS.2) Pairing and tracking of peers: The gateway needs information about the current state of the network, in order to adapt it's communication and other behavior to the dynamic system. It should therefore offer an initial point of contact and then keep track of peers' availability in an efficient manner (DE.1).

Chapter 4

Background and State of the Art

The largest part of the goals of this thesis are solutions to well-known and common problems. For example, many authentication and authorization solutions are already well established.

The challenge of this thesis therefore rather is finding solutions that are feasible in a new context: providing privacy in distributed and dynamic systems according to a new understanding. These systems might have completely different constraints and requirements than traditional ones. This understanding, the motivation for it and the reason of its necessity will first be explained in section 4.1.

In order to base this thesis on established approaches, after a short introduction the advantages and drawbacks of the application of different existing solutions in this particular case will be evaluated. The functionality is compared to the requirements that were identified in chapter 3, which provides the mode of evaluation.

Even if the described requirements are not completely fulfilled by any of the protocols, partial aspects already are solved. The identification of such approaches allows to re-use them in the own design and base the system on known standards. This allows to avoid common sources of mistakes and provide a familiar and intuitive design for users, operators and developers.

4.1 Privacy beyond anonymity

The introduction to this thesis in chapter 1 has shown the privacy implications of the traditional setup for data analyses. Once peers commit their data to a central entity, they do not have any control over it and how it is used any more. Therefore there is a requirement for trust between several of the entities.

Most importantly, all of the information is stored and analyzed at a central entity, which therefore has to make privacy guarantees that can not be reviewed let alone be enforced. Furthermore it has to protect the data from attackers, which try to gain access to the data by breaching the system. Also, data sinks may use the data they are authorized to access in any way they want since they do not have to state a purpose per request. Data sources can not review those requests and need to trust the data sinks not to abuse the data.

Traditionally, many systems follow the approach of anonymization or pseudonymization of data in order to provide privacy. An example for this is the TOR network, which aims to hide the origin of requests. [7] In research, many different anonymization techniques and measures have been suggested for providing and measuring anonymity, such as l-diversity and k-anonymity [8]. While this is supposed to prevent a direct connection of information to the data source, the raw data can still be analyzed without any limitations and conclusions about the data source can be drawn.

A current development is the collection of more and more data with increasing accuracy while data mining capabilities are improving [9]. This has raised the question about with methods are suitable for providing privacy in this context. Research has shown, that even from seemingly harmless, anonymized sensor data such as from gyroscopes and accelerometers significant conclusions can be made about users [10]. The release of anonymized AOL search engine queries and movie ratings on Netflix lead to privacy concerns as several cases were documented where the de-anonymization of users was possible [11]. Furthermore, the actual information as such itself may be critical. An example of this would be if somebody has searched their own credit card or Social Security Number on AOL.

The use cases in section 2.1 have shown the necessity for a privacy understanding beyond the discussed anonymization concepts. For example, for billing of electricity a landlord only needs to have access to data as a monthly total rather than raw data with high temporal and spatial resolution (2.1.1.1). In other examples, users of smart devices may want to selectively opt out of data collection (2.1.1.2 and 2.1.1.3).

Research has since presented alternative models of providing privacy for data sources [12]. Together with these new models and methods comes a new understanding of privacy, that involves the users' control over data [6]. The solution presented in this thesis therefore constitutes a proposed model of providing privacy that goes beyond the classical concept of providing privacy through anonymization.

This approach of ensuring privacy through the mechanisms used by a system is called privacy by design [13]. Different strategies may be employed in order to provide privacy by design, such as minimizing the amount of private information processed by a system and using private information only at the highest feasible level of aggregation [14].

4.2 Access control

A central mechanism needed for ensuring privacy is restricting access to resources. It should then only be granted in legitimate cases, where a requester can prove it has been sufficiently authorized for the intended usage of a resource beforehand. This section will explain the different mechanisms that are used to provide such functionality while considering the requirements that were explained in section 3.2.1.

4.2.1 Authorization policies

Access control requires the definition of access rights for users. These definitions are called authorization policies. The authorization decision can be made based on various attributes of the various subjects and objects in such a process. Therefore a system has to be found that defines how the policies are expressed and formalized.

In literature, several possible approaches have been discussed and the state of the art has changed several times due to the different requirements that came with changing information systems and security considerations. A selection of those and their applicability for the gateway solution is discussed in the following.

4.2.1.1 Traditional approaches for Access Control

Discretionary Access Control (DAC) means managing access rights per user. Mandatory Access Control (MAC) already provided some abstraction at the cost of granularity, since access is granted based on confidentiality levels of the resource and the clearance of the subject requesting access. In both cases, the management of access rights is user-centric [15].

As an improvement, Identity Based Access Control (IBAC) is based on constructs such as Access Control Lists (ACLs). In this resource-centric approach, users provide a credential that is checked against a white-list that is managed per object [15].

These solutions quickly showed scalability problems in big and quickly changing networks. Getting information about the current state of access rights is complex, which made withdrawing access rights from users or limiting access to resources a big administration effort. The discussed solutions had their strength in either one of the cases but not both at the same time.

The information handled by the system developed in this thesis is privacy critical. Therefore, access rights should be easily apparent and due to the flexible and possibly quickly changing topology, a reliable and easy way of changing access rights has to be present.

Early solutions for access control considered only a limited number of users, that rarely changed. Therefore, most traditional access control solutions are not feasible for use in this case.

4.2.1.2 Role Based Access Control (RBAC)

In contemporary IT systems, Role Based Access Control [16] is the most established model of granting access to resources. Compared to Discretionary Access Control and Mandatory Access Control, it provides easier management of permissions.

This is made possible by using abstract roles, to which privileges for the objects are assigned. When access is requested, the privileges of the assigned group are evaluated. A subject is granted access by including them into one of the applicable groups. Changing access rights is therefore possible by changing group membership or updating group privileges.

Due to the flexible nature of sensor-networks, this approach provides a considerable reduction of complexity in the management of access rights. Since the different types of subjects requiring access to various types of information can be managed in groups, at the same time a sufficient granularity of access rights is given.

On the other hand, this approach doesn't offer inclusion of any other types of information in the access request process though. In case the evaluation reveals such higher flexibility is useful in the described cases, this solution would not suffice.

4.2.1.3 Attribute-Based Access Control (ABAC)

The flexibility of the previously discussed access control solution "RBAC" is limited, since additional aspects other than group membership are not included in the decision of granting access. An example for this are environmental conditions, such as the current time, or attributes of the resources, such as their classification (public, secret etc.).

These aspects might be very useful in a setting of heterogeneous sensor networks though. This is due to the different parties that might need access to the sensor data, as well as due to the large number of types of sensors. Privacy and security considerations should be taken into account, depending on the location of peers. For example, a light sensor in the living room handles more sensitive data than one in a hotel lobby.

In order to provide such flexible solutions, the approach of **Attribute-Based Access Control** (ABAC) [15] has been suggested. Such an IAM may grant access based on attributes of the requester, requested resource, the desired operations or environment conditions. While allowing for far more flexible solutions, it therefore is by far more complex than the three previously described principles.

Decision: Sensor networks in real-world scenarios are far more complex then resources such as files or printers traditionally handled in information systems. At the same time, sensors can provide significant insights about the environment that they are deployed in and therefore the users that live in this environment as seen in the various of examples provided in section 2.1.1.1. Due to this increased complexity and privacy requirement, a flexible model is needed.

Therefore Attribute-Based Access Control will be used as an access control model in this thesis.

4.2.2 Proving authorization

Since PP.1) demands transparency for peers and AC.3) requires independence from the gateway, it is not enough for the gateway to simply store authorization information and only communicate the results of the authorization decision process to peers. The protocol needs a possibility of directly proofing authorization to peers in a way that allows them to verify the information. This solves part of the problem that is presented by AM.6), where the gateway could otherwise grant itself access to data of the peers.

4.2.2.1 X.509 authorization certificates

The X.509 [17] standard is one of the most widespread solutions for certificates and describes a format of public key certificates that can be used to define a hierarchy of keys that are signed by certificate authorities. It is best known for being used in TLS connections, which again is used in many application layer protocols such as HTTP or SIP.

In X.509 infrastructures, usually a CA or their delegates issue certificates. These might not be the most apt entity for deciding authorization questions, since their scope usually is just the identity. In order to solve these shortcomings, an Attribute Certificates Profile is defined for Authorization and the authorization process transferred to an attribute authority. [17] This means, that certificates are specified that can contain information about the privileges of a subject. In one of the described use-cases this could be the authorization to access information about the amount of electricity used.

The biggest problem of revocation of access rights will be discussed in section 4.2.3.3.

4.2.2.2 SAML and XACML

Security Assertion Markup Language (SAML) [18] and eXtensible Access Control Markup Language (XACML) [19] are two complementary standards for authentication and authorization. They were both developed by the Organization for the Advancement of Structured Information Standards (OASIS) while considering their interoperability. The standards describe XML-based markup languages and the communication process for their respective scope.

SAML can be used for communication of authentication information but also about entitlement and attributes of authenticating entities. Sharing this kind of information allows to provide single sign-on, since the identity of the requester can be described.

XACML on the other hand is a standard for the description of access rights and a request/response language for queries about these. It allows for the implementation of an ABAC solution and therefore also of most other authorization schemes.

In the context of this thesis, the authentication process for an information query might happen centrally at the gateway. Subsequently, peers can request authorization information about the requester from the gateway. The protocols are especially interesting since they have their focus on easy extensibility. Also, environmental attributes such as clearance for confidential sensor data could be included.

The XACML standard describes a JSON profile, which can be used in order to provide a format that is consistent with the other messages in JSON-based services.

Decision: The concept of cryptographically secured authorization certificates as very useful in the use case described in this thesis. This is due to the fact, that authorization can be proved directly between two parties and no third party is required after issuance of the grant. XACML on the other hand provides a very flexible way of representing authorization information.

Therefore this thesis will develop a solution that is based on the concept of authorization certificates while using the XACML format for representing authorization information.

4.2.3 Authentication

The attacker models of malicious entities AM.5) and AM.6) constitute a threat to nearly all of the protection goals, since they might try to assume a false identity in order to execute illegal requests. By breaching protection goal ISP.3), it would therefore by proxy also have implications for the protection goals ISP.1), ISP.4), ISP.5) and PP.1) since any entities could not be sure about the identity of the communication partner any more.

The necessity of a secure authentication solution is therefore obvious.

4.2.3.1 Kerberos

In a dynamic setting such as described above, using stateful authentication solutions requires additional communication of resources with the gateway and has additional drawbacks such as redundancy and limited scalability.

Kerberos is a widespread, stateless authentication protocol that offers a solution to this problem. Instead of using centrally stored and managed sessions, clients can request a ticket from a "Key Distribution Center". This ticket then can be used in order to request further tickets or access to a resource, each by authenticating directly to the counterparty.

The advantage of this is, that a client only has to authenticate once at the beginning at the KDC and can then use the proof of authentication for further requests. Additionally, multiple KDCs may be used, which provides redundancy and easier scalability.

4.2.3.2 Public key authentication

Since authentication through secret credentials, such as passwords or tokens, comes with security flaws, an alternative approach is the usage of public key authentication. Here a challenge is sent to the authenticating party, which subsequently signs it with their private key and sends it in their response. The response can then be verified with the public key.

Such as it is a challenge in most public/private cryptography applications, this brings the need for public key distribution since those must be stored on all peers that need to provide authentication. Remembering the large-scale and flexible networks considered as a primary use-case of this thesis, scalability and regular re-keying of large numbers of system consumers need to be covered.

4.2.3.3 X.509 certificate authentication

The public keys contained in X.509 certificates can be used as a basis for solutions of security requirements such as authentication and confidentiality. Since the solution has proofed itself as highly scalable due to its use in many internet applications, consideration for the described use cases is an obvious option.

A major flaw is the problem of revocation. Additions to the standard such as certificate revocation lists exist, which are accompanied by approaches such as the Online Certificate Status Protocol or Certificate Stapling. These mean additional traffic and complexity though, which also has implications for the scalability in large-scale networks. At the same time, strict enforcement might come with usability restrictions. The different certificate revocation schemes all have their own drawbacks, which mostly consists of incurring an overhead [20]. As an alternative, short validity periods can be defined and certificates then are periodically renewed and re-distributed.

4.2.3.4 OAuth

When devices are authenticating on behalf of another authorized entity, using their password means putting it to risk since it is revealed to possibly untrusted parties. This would mean full access to all their privileges, even if just a part of these is required for an application to function.

OAuth [21] provides an authorization framework for access delegation. If a user grants access to applications, a token is issued that the application then can use to access the resource. Even though OAuth is an authorization protocol, the token can be used to authenticate to a server, if it provides the service and the token at the same time or the tokens are distributed to the resources.

A use case for this would be granting access to certain information of a sensor network to a system, so that it can access the resources for automated analyses.

4.2.3.5 OpenID

In the OAuth section it already became clear, that direct authentication through secrets requires knowledge of this secret by all accessible resources. This brings obvious implications for security in case of compromised peers or connections.

In order to avoid local storage of such information, OpenID [22] provides an open standard for centralized authentication. If a subject requests access, it states it OpenID and proves ownership of this ID which then is verified by a central server.

This process transfers well to the described cases of sensor networks. If some party requests access to information from a sensor, it could proof ownership of an ID that is associated with an account that is stored on the gateway. The gateway would therefore become the central server, that verifies the login information.

Decision: Some of the solutions presented here directly or indirectly involve a third party in the actual authentication phase, since it needs to verify a secret or grant a temporary document proofing authentication. Public key and certificate authentication is a wide-spread standard, that allows to proof authentication directly between two entities. The problem of revocation is solved by setting short validity periods in order to keep the complexity of the system low.

Therefore this thesis will use certificate authentication for all communication between the different entities.

4.3 Request processing

In order to provide the data of peers to clients, the gateway needs a component that processes requests and translates protocols. The solution has to fulfill the requirements that were discussed in section 3.2.2.

4.3.1 Unique addressing of artifacts

In information systems, automated processing makes handling of large numbers of artifacts such as messages, records or physical assets possible. For later reference or usage, they may need to be referenced from other artifacts. This raises the question, how an automated method can be used to specifically identify such an artifact without the risk of mixup with others.

Such an identifier can be generated in various ways and many different standards have been proposed. For example the Universally Unique Identifier (UUID) standard proposes a 128 bit identifier [23]. Choosing a value from this extremely large range can be based on various factors such as time, simple hashes or random values.

The disadvantage of using such values is, that they provide no way of validating the contents of the artifact that they address. A malicious entity, that is asked to present the artifact that belongs to an identifier, could present any contents. In cases such as such as authorization grants, this is security critical if forged information is presented.

4.3.2 Using digital signatures as identifiers

Modern cryptographic hashes or digital signatures can be used as an identifier for data objects. Due to their size, collisions are extremely improbable and infeasible if the underlying signature scheme is computationally secure. A plain hash provides the advantage of making it possible to verify the data. Using a signature adds the advantage, that the source of a message can be attributed to an entity.

If the signatures are calculated anyway, no additional overhead is incurred by this method. Such addressing of objects by a cryptographic digest is used by protocols such as IPFS [24].

Decision: Since signatures are impossible to be tampered with and at the same time provide integrity for the data object that they belong to, they will be used for addressing content in this thesis.

4.3.3 Blockchain storage of information

The blockchain is an append only medium that ensures correctness of newly inserted information using a consensus protocol [25]. Using cryptographic primitives it is ensured under certain circumstances, that no entity can modify the contents of previously included records without this being detected. No party can deny the validity of the contents of the blockchain as long as the boundary conditions are met. It therefore provides a way of guaranteeing accountability and non-repudiation for information once it has been added.

These properties provide advantages for many applications apart from the original purpose, which was the transfer of digital assets. For example, it can be used to permanently store any kind of messages or requests to resources. Later on, it can then be proved that the respective message has been sent by an entity. Therefore it provides accountability by making the information accessible, while non-repudiation is ensured by the contained message.

Decision: The properties of the blockchain, especially it being an append-only medium, are very useful for logging data. Therefore it is used as a supporting feature and will be used as a medium of storing information for later forensic purposes.

4.3.4 Web service interface protocols

All involved entities should be able to communicate with the gateway easily, reliably and using standard protocols. This helps to integrate the data and functionality offered by the gateway into new applications without a significant amount of knowledge about the SMC protocols used in the background.

The gateway should therefore expose an interface that is easy to understand and implement. This raises the question, how to define the endpoints of the interface and the message formats that it uses for communication

4.3.4.1 Simple Object Access Protocol (SOAP)

The Simple Object Access Protocol (SOAP) [26] is an example for a general protocol for interface design. Based on XML, it defines how web service interfaces can be designed in a standard way and how participants can communicate over those interfaces.

Due to the underlying format, the standard is very flexible and extensible. All messages can be validated using schema definitions as they are typical for any XML protocols.

On the other hand, the extensive standard makes fully compliant implementations more complicated. Furthermore, the XML is less easy to read or write. Using SOAP therefore is only feasible on the client- or server-side when using special tools and libraries, that help with the implementation.

4.3.4.2 JSON communication over HTTPS following the REST paradigm

Representational State Transfer (REST) defines a set of principles, that should be used when designing web service interfaces [27]. Examples for this are using a stable format for messages or using the URI for addressing an object while the method used in the request is transmitted in a different manner.

It is not defined however, which specific protocols or technologies should be used for an actual implementation. Furthermore, the REST paradigm is not standardized but rather constitutes a widespread convention on how to design and use interfaces.

A common combination of technologies used in this context are JSON messages transferred over HTTP or HTTPS. Messages that are serialized in the JavaScript Object Notation (JSON) format [28] are by comparison with e.g. XML by far easier to read or write. Furthermore, they can be easily represented by native map-datatypes of modern programming languages, such as dictionaries. Using HTTPS provides the advantage of using a widespread standard and compatibility with most platforms that are connected to the internet.

4.3.4.3 gRPC and protocol buffers

The textual representation of the messages in the previous two communication protocols causes performance losses. On the one hand, message sizes are increased since the values of the different attributes are not optimized for size but rather for easy readability. Additionally, messages have to be serialized and then again de-serialized before and after transmission.

Simply transferring information in it's binary format as it is stored in memory is no solution, since the representation would then be platform dependent and complicate interoperability between programming languages. gRPC [1] provides a protocol that uses the binary format Protocol Buffers [2] to transfer information between communication partners. It can be used for communication over the network between different platforms and can be used with many different programming languages.

Decision: Since the thesis describes networks of embedded devices and is supposed to provide easy integration of the provided data in other systems, a compromise needs to be made.

[1]https://grpc.io/
[2]https://developers.google.com/protocol-buffers/

gRPC and protocol buffers are interesting in cases where very high performance is needed. Since SMC protocols are expensive and a lot of information is transmitted during communication with the gateway anyway, the relative impact of performance improvements through using protocol buffers would be minimal though. At the same time, the integration of the gateway interface in other software would be made considerably more complicated.

XML-based SOAP messages are not very compact since they contain a lot of information that doesn't relate to the actual information that needs to transferred and at the same time difficult to read by humans.

JSON messages will therefore be used, since they are compact and simple to use. Using HTTPS and the REST paradigms, it follows a contemporary and widespread way of providing APIs.

4.3.5 Fulfilling information security goals in communication

Since the request processing component is the central component for handling and forwarding all communication, it has a central position in fulfilling information security goals. The following technologies are options for providing this security.

4.3.5.1 Javascript Object Signing and Encryption (JOSE) formats

The standards of the JOSE series [29] of the IETF describe a set of formats for representation of different cryptographic information in the JSON format. The container formats allows to include all information necessary for application of the correct method such as method identifiers, versions and keys and describes the representation of information resulting from the cryptographic methods. A set of algorithms that can be used with those formats for cryptographic processing of data are described in the JSON Web Algorithms standard [30].

JSON Web Signatures (JWS) provide a standardized way of deriving and representing signatures for arbitrary objects that can be represented as text, without impairments in the verification process that would e.g. result from the flexible order of fields in JSON documents [31].

JSON Web Encryption defines the representation and description of encrypted content in the JSON format [32].

JSON Web Keys are a format for representing and describing key and certificate information [33]. A method to obtain a distinct identifier a key is described in the JWK thumbprint standard [34].

Decision: The formats of the JOSE series provide simple integration when using the JSON formats such as in this thesis. Therefore they will be used for providing a solution for different security requirements.

4.3.5.2 Transport layer security

The Transport Layer Security protocol [35] defines a standard approach for providing a secure channel for communication of application layer protocols. Due to it's usage in many protocols such as HTTP, SMTP and SIP, it constitutes a widely spread standard. It therefore is highly likely that clients will be able to communicate over this protocol. Apart from confidentiality it also provides integrity through message authentication codes.

The protocol is initiated with a handshake, where protocol details are negotiated and information such as certificates are exchanged for authentication as described in section 4.2.3.3. Based on the result of the handshake, data can then be exchanged using TLS records as a container format.

Decision: TLS is a widespread standard and supports the X.509 certificates for authentication as described earlier, therefore it will be used to secure communication between the entities.

4.4 Directory Service

The following technologies exist as possible components for a solution that needs to fulfill the requirements defined in section 3.2.3.

4.4.1 Lightweight Directory Access Protocol (LDAP)

The gateway needs to store and provide information, both as a primary function such as in directory queries and as a supporting function for the other components. LDAP provides a full directory service, which can provide distributed directory services in IP networks [36]. E.g. active directory provides a common implementation of LDAP, that is widespread for use in corporate networks.

Through the so-called subordinate and superior knowledge information, one LDAP directory service can refer to other services for further information. [37] An example for this could be running some central directory service, that holds information about all currently available devices. For further information about the current state of the sensors, the requester would have to contact a local directory service though.

The possibility to set up such a hierarchical, distributed network might prove useful for the described use case of storing information about sensor networks and attributes of the single peers that are connected to it.

The extensive standard and covered functionality make the protocol very complex though.

4.4.2 Document-oriented databases

Contrary to relational databases, document-oriented databases store data in key-value pairs. Unique identifiers can be used as the key, while the value may be arbitrary content that does not require any special structure.

An example for such a database is MongoDB [38]. The representation of objects in the database uses the same structure as JSON-serialized documents. Queries can be made involving the subfields of such objects.

Decision: LDAP is highly complex, while the organization of entities in this thesis is flat and they can be described by a few simple attributes. Therefore the directory will be based on a simple document-oriented database, where each document represents one peer.

Chapter 5

Design

After analyzing the requirements of the gateway solution in chapters 2 and 3, they were compared to currently existing solutions in chapter 4. A solution that covers all of the requirements was not identified, therefore a system will be proposed in this chapter. The goal is to provide a design that makes the retrieval of information from distributed and dynamic system environments easy and feasible in practice. At the same time, the system should support and actively enable data sources to protect their privacy.

Based on the previous findings, first an architecture of the system including all of the entities will be derived in section 5.1. The components and the architecture of the gateway are then discussed in more detail in sections 5.2 to 5.4. The description consists of the used methods and their application in the protocol, and the format of the messages that are used for transmission.

Then the specific mode of functioning of the protocol has to be defined. As a combination of the architecture and the interaction, an exemplary protocol is laid out in section 5.5. This protocol shows how the system can be used to cover the functional requirements described in section 3.2

5.1 System architecture

In section 2.2, a system with a central point of contact for querying data and coordinating communication with peers was described. The system shall therefore provide virtual centrality. This system design paradigm means hiding the complexity of decentralized systems to clients while maintaining the advantages it provides in other fields such as privacy [1]. This functionality is provided by the gateway.

An abstract view of the different parties involved in communication in the designed system can be seen in figure 5.1. On the top, the clients can be seen. They request data from the gateway solution, which can be seen in the middle. The gateway consists of different components, which are coordinated with the goal of processing and then forwarding the requests. The peers on the bottom, which constitute the data sources in this system, then accept and process this forwarded information.

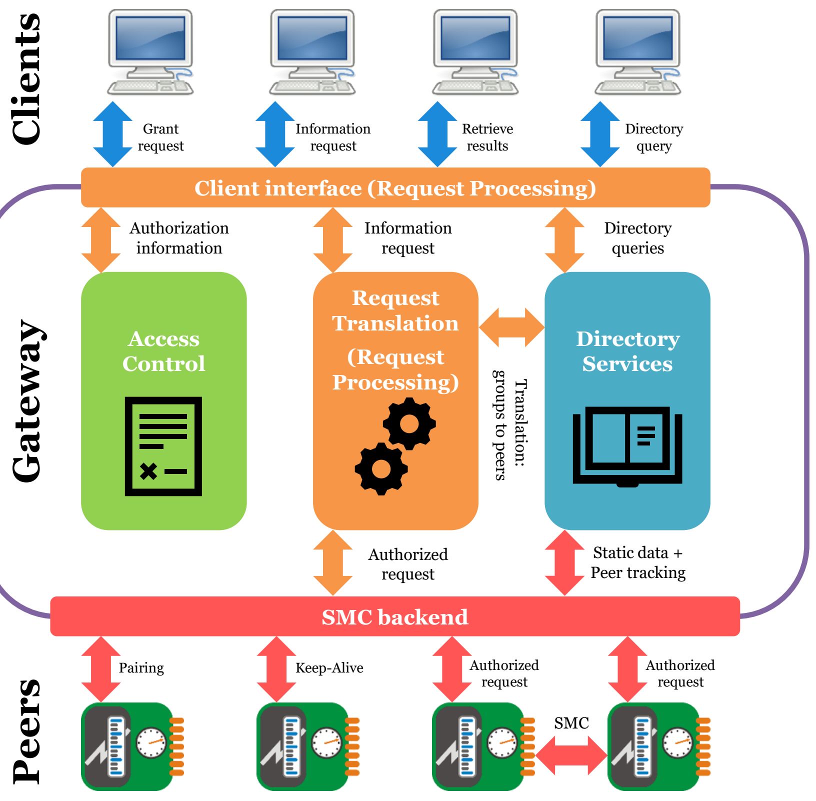

Figure 5.1: Gateway providing central point of contact for all communication while coordinating components

The specific scope, functionality and abstract architecture of each of these entities is described in the following subsections.

5.1.1 Clients

In the context of this thesis, the data sinks that make information requests are called clients. Since they are not trusted, it has to be assumed that their goal is to gain access to as much information as they are able to obtain. All of their communication happens directly with the gateway, which processes their requests and handles the subsequent steps of the protocol.

As a first step they can query information about available peer groups. Using this information they can then request an authorization decision and document from the gateway, which is necessary for further steps. Using this document, they can send their information requests to the gateway. Lastly, they can then request the results of earlier information requests.

The data they demand in legitimate requests is supposed to be used in various use cases. Without receiving this requested information, they may not be able to function correctly or provide the added value that they promise. Therefore a deliberate trade-off has to be made between granting the authorization that they need for correct functionality and the privacy requirements of the peers.

5.1.2 Peers

The peers constitute the data sources in the setting of this thesis. They don't directly communicate with clients in order to provide information. Much rather, after registering to the gateway and periodically sending updates with their attributes and metadata, they are available for receiving and processing information requests. Depending on the SMC protocol used, they then collaborate in order to provide results. For their deployment they therefore only need a peer certificate and the address of an gateway.

Due to their physical location, ownership or operator they are likely to be under control of somebody other than the gateway or the client provider. Therefore, as little trust as possible should be necessary between the peers and the gateway or clients.

Since they are in possession of the actual raw data that is queried, their goal is to enforce the privacy protection goals discussed in 3.1.2 by using the mechanisms that ensure information security goals discussed in 3.1.1. Therefore they provide the second layer of privacy-preserving access control.

They should only allow queries from authorized peers that they trust and evaluate the legitimacy of the request itself. In case any of these circumstances are not given, they should make use of their veto right that is given to them due to the demand of requirement PP.3.

In the context of this thesis, the peers are trusted to act in a trustworthy and non-malicious manner, as long as they can provide a certificate that has been signed by the certificate authority. Therefore there is no process for checking the plausibility of any information that they provide.

5.1.3 Gateway

Clients and peers directly communicate with interfaces exposed by the gateway for all the different phases of communication, such as directory queries, grant requests and data queries themselves. The gateway then processes these requests and manages the interaction between the different components within the gateway that were defined in section 2.4. After taking care of the communication with peers, it provides processed results.

Based on the different components that were identifier in section 2.4, the gateway design follows a modular approach, which has several advantages. First of all, some privacy-critical parts like the access control component can be provided by a trusted party. This is especially important in the adversary models AM.3 and AM.6. Second, the scalability is improved since the different components can run on a distributed system rather than one single system which is important for requirement DE.1. Third, if in such a system one of the components fails, it can easily be exchanged while the other components can continue their work which helps to fulfill requirement RP.1.

These are the different components and their functionality on a high-level view:

5.1.3.1 Directory services

Section 3.2.3 described the scope and requirements of the directory component. Most importantly, it keeps track of the current state of the dynamic network. This information can then either be offered to communication partners or be used for processing of requests that the gateway receives.

Details of the different functions of the directory services component can be found in section 5.2. The definitions of the messages that implement this functionality can be found in section 5.5.1.

5.1.3.2 Access control

A central purpose of the gateway is to offer a first layer of access-control as it was described with the scope and requirements for the access control component in section 3.2.1. Looking at it from the client-side, the gateway needs to provide some form of authorization policy management and authorization verification process for requests.

These authorization decisions are independent of actual information queries. The decisions are then provided in a format that can be presented in later information queries. These two processes, which usually happen in conjunction with other systems, are therefore decoupled. The authorization grant itself will be discussed in further detail in section 5.3.1.

Details of the different functions of the access control component can be found in section 5.3. The definitions of the messages that implement this functionality can be found in section 5.5.2.

5.1.3.3 Request processing

Clients with sufficient authorization should then be able to obtain information from peers. The boundary conditions and mode of functioning for this request translation process was described in section 3.2.2.

After an initial information request from the client, the gateway verifies it and then forwards an "authorized request" to the peers and to the original requester. The peers process this request and subsequently contact the gateway with their response, that either holds results or a veto or failure message. The client can query the gateway about the current level of completion of the information request and intermediate results.

Details of the different functions of the request processing component can be found in section 5.4. The definitions of the messages that implement this functionality can be found in section 5.5.3.

5.2 Directory services component

The directory maintains all information needed for the correct functioning of the gateway, apart from the authorization information which is handled by the access control component as described in section 5.1.3.2. The directory component therefore mainly serves an assistant function when storing data. All of the information specified here is stored persistently, e.g. in a database, in order to ensure correct functioning of the gateway in case of a temporary system outage.

5.2.1 Stored peer attributes and state information

As stated in requirement DS.1, the gateway needs to store information about the state and capabilities of peers.

The current address and state of availability are need for information requests. Information about the certificate of a peer has to be stored in order so that it can authenticate later.

The information that a peer offers can be defined by dynamic attribute key:value-list mappings. If two or more peers define the same set of attributes, they form a group. These groups don't necessarily have to be reflected in a distinct data structure. These attributes and the derived groups can be used to describe the peer in different ways, such as it's location, required clearance and offered data.

Ambiguous semantic meaning of groups is possible and no specific groups have to be defined from the code-side. As a simple example though, the following list 5.1 can describe a temperature-measuring peer that is located in the 3rd-floor meeting room of an office, that has high confidentiality requirements:

```
{
  "peer_attributes": {
    "clearance": [9],
    "rooms": ["all", "meeting_room"],
    "floor_number": ["all", "3rd_floor"],
    "sensor_data": ["temperature", "light"]
  }
}
```

Listing 5.1: Example for peer attributes

While at least one specific value was given for each attribute, multiple or no values per attribute are also possible. The value *all* in the attribute-fields *rooms* and *floor_number* shows, how this can be used to assign a peer to multiple groups. In this manner, groups of sensors in either all rooms on a certain floor or e.g. the meeting rooms on all floor can be defined.

The retrieval of this information is discussed in section 5.2.2. In summary, the following information needs to be stored about each peer:

- Peer identifier (serves as certificate identifier)
- Last seen
- Next planned keepalive message
- Current state of availability
- Current contact address
- Peer attributes and capabilities

5.2.2 Tracking of peers

As stated in requirement DS.2, the gateway should provide a central point of contact for peers, where they can announce their state and capabilities or any changes to it. This functionality describes the mechanism of obtaining and curating the information that was described in section 5.2.1.

The initiation of such a tracking process between a peer and the gateway is a pairing mechanism. If at a later stage the state changes due to re-location of the peer, it should be able to provide the gateway this updated information.

Since the setting of this thesis is a dynamic environment, a peer may not be able to properly temporarily de-register itself before becoming unavailable. Therefore the gateway keeps track of the last contact it has had with a peer. This value is for example reset, when an update is received by the gateway.

Sending an entire update message for this purpose would constitute a large overhead due to re-transmitting information about attributes and state information, that has not changed. Therefore as a more efficient mechanism, peers can periodically renew their availability information by sending a keepalive message to the gateway, as it was described in requirement DS.2). If too much time has elapsed since the timestamp of this last communication, the gateway can assume the peer to be unreachable.

The connection quality between the peers and the gateway strongly depends on the purpose that they are used for. Different usage scenarios therefore require for different frequency of keepalive messages sent. Therefore the time that needs to elapsed until a peer is set to inactive can be defined by the peer in a separate field.

5.2.3 Providing processed metadata

As stated requirement DS.1, the clients need to be able to query the contents of the directory about peer information in order to be able to make relevant requests. The data that is described in section 5.2.1 therefore needs to be made accessible in a simple and privacy preserving manner.

The attribute described in section 5.2.1 can be used in order to describe the peers that provide the desired kind of information. When clients make requests, they therefore need to specify which kind of data they want (temperature, electricity consumption etc.) from which peer groups.

However, before a request they want to know which groups are available and how big they are. The gateway should therefore provide aggregated information. This abstract view makes the information easier to understand by clients, that don't need to know the exact state of all peers. Furthermore this removes privacy implications, since providing raw instead of aggregated metadata would be privacy critical.

If the peer doesn't define any attributes that it wants to query by, all attributes and their number of occurrence are presented. Adding more attributes to the request increases the specificity of the query and refines the granularity of the groups. An example for the aggregated contents in a response to a directory query can be seen in listing 5.7.

5.2.4 Storing information for later retrieval

The previously described peer information is dynamic. Apart from this, the gateway also needs to store some static data that is requested from other components or other entities as described in the section 5.4.2.

The following information is stored for quick retrieval (e.g. <100ms) when requested by other components as described in section 5.5.1.4: All of the records are stored in their JOSE container formats as they were described in section 4.3.5.1. If not specified differently, this is the JWS container, while the signature is used as a unique key for retrieving the object.

- Client and peer certificates in JWK format, using the key thumbprint as an identifier
- Authorization grants
- Authorization policies
- The original request of an authorized request

Information is added to this storage by the respective components such as access control or request processing.

5.2.5 Logging in a private blockchain

Other than that, storing static data also serves for logging. Requirement ISP.4 stated the need for accountability, which needs to be ensured in gateway communication. Section 5.2.4 stated, that all received and sent messages are being logged in order to provide traceability of requests. Generally, the purpose of logging is to provide accountability since the course of communication can be reconstructed later. It allows to proof illicit behavior of entities, in this case malicious clients. Since the logging is controlled by the gateway, the deletion of data would be possible in normal storage solutions.

In contrast to the data stored for retrieval during queries, the data that is being logged is only used for forensic purposes. This means, that the later retrieval of data that has been logged is not time critical. Therefore large quantities of data can be stored without special performance requirements.

The proposed solution uses a blockchain for this purpose, as it was discussed in section 4.3.3. Using the blockchain in the background therefore adds non-repudiation in case of a malicious gateway. Since it provides an append-only medium, once data has been added later deletion could be immediately noticed. By using a private blockchain, performance is higher than when using a public blockchain and it is guaranteed that only authorized entities can gain access to information.

The records that are stored are the raw JWS containers of the messages, which provides an additional layer of verification of the source and integrity of data. Integrity of data is ensured by the signature of this container that all messages are stored in.

The private blockchain runs at a minimum of one node, which is part of the gateway directory component. Additionally, other entities may connect to the blockchain in order to have a copy for later proofs or increase the certainty level by providing an additional node in the consensus process. Entities may only connect if they are authorized to do so. The authorization process requires that the gateway operator actively adds a node to the blockchain network together with their public key that is used for authentication and sets the according permissions of the node.

If a node connects without adding further access rights, they may read all of the contents of the blockchain. If they are granted mining permissions, they participate in the consensus mechanism of approving data additions to the blockchain. Then a proof of work mechanism is used between entities, since they potentially don't trust every node. In that case, a single party would only be able to withhold adding of information if they offer more than 50 percent of the computing power in the network. There is no risk when false information is included, since the signatures stored with the records allow to verify any information added to the blockchain logs.

The storage of data is by default triggered by the directory component of the gateway, if the request processing component calls the according function. Other nodes may also store information, if they are granted the appropriate access rights.

5.3 Access control component

Making information requests requires proof of authorization. The access control component keeps track of all authorization information and makes authorization decisions based on this (AC.2). In order to do so, it implements the XACML standard as it was described in section 4.2.2.2. Authorization information is represented as policies as they are defined by XACML and it's JSON profile [19].

The big advantage of using this standard is, that as demanded in requirement AC.2 it is very flexible and extensible regarding the attributes that can be used for access decisions. The gateway solution and request protocol therefore do not need to make decisions about the allowed attributes, which makes the developed solution very flexible for using it in different contexts.

If a client should be authorized for access of data, such a policy can be sent to the gateway that contains all of the information that a client may request and the constraints under which a request will be approved. Examples for these attributes like the location of the peer and the time of the request can be found in section 2.4.1.

5.3.1 Proof of authorization through stateless grants

In section 4.2.2.1, the X.509 authorization certificate format was presented. It defines exactly the kind of authorization proofs that is necessary for a stateless proof of authorization. In the described dynamic system there were some drawbacks due to the high complexity and low flexibility though. While preserving the general mechanism of a signed document that contains authorization information, a more lightweight custom solution is used in this protocol and will be described in this section.

The client may request this document from the gateway in order to have a verifiable document that it can present in further requests. This document will from now on be called authorization grant. The format of this authorization grant is described in detail in section 5.3.1.3. The process and the messages that are used to obtain this authorization grant are based on XACML and described in section 5.5.2.

5.3.1.1 Obtaining an authorization grant

The XACML standard defines a format for *authorization requests*. Here, the attributes of the request like client attributes or current time at the moment of the request can be defined. The access control component then compares this information to the previously stored policies, in order to decide if sufficient access rights could be proven, and if the request will be permitted or denied.

The outcome of this decision process needs to be made available to the peers in later requests. Since the gateway should not need to hold any state information, requirement AC.3 demanded a form of stateless authorization grant that a client can use for such proofs without such authorization states. This document should then be possible to be verified and contain all information so that an entity can assess the legitimacy of the request that the authorization grant was presented in.

The access control component therefore provides an *authorization decision* as defined in the XACML standard. This document provides all of the information that is necessary to allow privacy-preserving access control (AC.1). It clearly identifies the client that was granted access and the attributes and therefore groups that he was granted access for. Also other information can be included here, such as recommendations or directives to components that enforce the decision.

A central measure of providing transparency to peers consists of providing them information about the source of this authorization as demanded in requirement AC.1. In order to verify this information, the decision document contains the identifier of the authorization policy that the decision was based on. After the access control component signs this decision, it can be verified by any other party. This therefore allows this signed document to be used as a stateless authorization proof.

5.3.1.2 Attributes defining the scope of a grant

The authorization decision can be made based on various factors, which is necessary due to requirement AC.2). While the format was designed in a way that is extensible, a set of common attributes was defined within the thesis.

As an continued example from the listing 5.1, the following attributes are used by the exemplary solution:

- Room and floor number (peer attributes)
- Sensor data (resource attribute)
- Clearance of client (client attribute)
- Current time (environmental attribute)

Since authorization is defined by a list of attribute-value pairs, only peers that constitute a conjunction of all of these attributes are covered by such an authorization grant. Since a peer can define several values for each of their attributes, it is still possible to define various groups with this approach. Therefore, a process for disjunctive attribute groups is not necessary in the authorization grant and not provided in this context (defining such access policies is possible though).

Additional attributes can be defined in accordance with the XACML standard, which is entirely supported by the access control component.

5.3.1.3 Authorization grants format

Listing 5.2 shows the format of the authorization grant, while the different contained fields are described below. The document is embedded in a JWS object as described in section 5.4.1, which provides the certificate information of the gateway and a verifiable signature.

```
{
  "request_status": "successful",
  "valid_until": "2018-02-17T10:46:43.599008+00:00",
  "authorization_grant_xacml": {
    "Response": {
      "Result": {
        "Decision": "Permit",
        "Attributes": [
          {
            "@Category": "client_identifier",
            "Attribute": {
```

```
          "@AttributeId": "client_identifier",
          "@IncludeInResult": "true",
          "AttributeValue": {
            "@DataType": "http://www.w3.org/2001/XMLSchema#string",
            "\$": "92429d82a41e930486c6de5ebda9602d55c39986"
          }
        }
      },
      {
        "@Category": "sensor_type",
        "Attribute": {..., "AttributeValue": {
            ..., "\$": "temperature" }}
      },
      {
        "@Category": "room_name",
        "Attribute": {..., "AttributeValue": {
            ..., "\$": "meeting_room" }}
      },
      {
        "@Category": "floor_number",
        "Attribute": {..., "AttributeValue": {
            ..., "\$": "3rd_floor" }}
      },
      {
        "@Category": "time",
        "Attribute": {..., "AttributeValue": {
            ..., "\$": "12:00:00" }}
      },
      {
        "@Category": "clearance",
        "Attribute": {..., "AttributeValue": {
            "@DataType": "http://www.w3.org/2001/XMLSchema#integer",
            "\$": "9" }}
      }
    ],
    "PolicyIdentifierList": {
      "PolicyIdReference": {
        "@Version": "1",
        "\$": "92429d82a41e930486c6de5ebda9602d55c39986"
      },
      "PolicySetIdReference": [
            {..., "\$": "1b6d20788dea279d8156c9dbe6bec46c96316e87"},
             ...,
            {..., "\$": "__root_policy_placeholder__"}
          ]
        }
      }
    }
  },
  "request_status_message": "Authorization successfully granted, the
      client may therefore request the data in scope of this
      authorization grant"
}
```

Listing 5.2: Authorization grant format

Note: In case of repetition of information, that was seen in other fields, it has been emitted.

The *request_status* fields shows the result of the authorization request, while some additional information can be provided in the *request_status_message* field. The *valid_until* field in the ISO 8601 format defines a validity period of this authorization grant, which can be freely configured.

The field *authorization_grant_xacml* provides the actual authorization information in the XACML format. While the exact contents and format depend on the specific XACML-implementation, it should at least contain the following information:

The *Decision* field contains the original decision message from the access control component. The *Attributes* field contains a list of the attributes that the client has requested access to as described in section 5.3.1. In each of the attribute dictionaries, the name of the attribute can be found under *@Category*, the specific value is found in the *Attribute-Value* field.

The *PolicyIdentifierList* shows the source of the authorization decision, which leads to the identifier of the message that originally announced these access rights (here shown by the exemplary stub signature *1b6d....6e87*). Using the message that can be obtained by this identifier, the peer can check who posted this authorization policy and if they trust this source to grant this authorization. The *PolicyIdReference* is used as the client identifier, which here has the value *9242....9986*.

5.4 Request processing component

This section will describe the transmission of messages between the gateway and the different entities of the protocol and how these design decisions fulfill the different requirements. Since security and privacy are in scope of the message exchange protocol, the format and mode of transmission of messages is especially influenced by the non-functional requirements that were discussed in chapter 3.1.

Due to it's lightweight structure and easy implementation in interface consumers, the information that is exchanged between the different entities is serialized as JSON in a manner that is based on the REST paradigms as described in section 4.3.4.2. Furthermore, the developed system uses the three standards of the Javascript Object Signing and Encryption (JOSE) series, that were discussed in section 4.3.5.1.

5.4.1 Transparent and non-repudiatiable requests

Most importantly, the protocol needs to provide a way of making transparent requests (PP.1) that support accountability (ISP.4) of the requester. After receiving an information request, the gateway first verifies all included information such as the signature, timestamp, authorization grant and client certificate.

In order to provide transparency (PP.1) and intervenability (PP.3), all information that helps the peer to understand the purpose and origin of the request should be provided. By including the original request into an authorized request message, the gateway provides all of this information, which the peer needs in order to understand the source and legitimacy of the request.

This transparency makes it possible to log request details and present them later-on as proof of requests that try to exceed the client's access rights as it was demanded in requirement ISP.4). Any request comprises the risk of disclosing confidential information. Integrity, authenticity and accountability (ISP.3, ISP.4, ISP.5) of the messages that result from this request translation should therefore be ensured in all protocol phases.

This can be achieved through cryptographic signatures over data such as requests or authorization information. Using the public key that belongs to the signing entity, any peer that receives this document can then verify the origin, countering the adversary models AM.4-AM.6. Storing these authorized requests makes it possible for any entity to later hold the original requester accountable.

This authorized request is sent to the client regardless of any other SMC protocols that are potentially used, since it provides the information that is used for the privacy-preserving property of the request translation.

In the present design, the JSON Web Signatures standard as discussed in section 4.3.5.1

is used. The corresponding key and certificate certificate can be identified by a key thumbprint in the "kid" field in the header of the JWS-object. In order to evaluate and verify the signatures, the certificates and keys can be obtained from the gateway as described in section 5.5.1.4.

5.4.2 Avoiding redundant transmission through unambiguous addressing of content

In every message of the protocol, static information that does not change between communication phases and protocol runs is necessary. An example for this are the certificates and keys that are necessary for verification or decryption of messages as described in sections 5.4.1 and 5.4.5.2. Another example are the stateless authorization grants that are described in section 5.3.1.

In order to provide an example that shows the scale of this problem, the information request shown in listing 5.13 has a size of approximately 300 Bytes excluding the authorization grant. The exemplary authorization grant shown in listing 5.2 has an approximate size of over 3000 Bytes. Due to the way that transparency is provided as described in section 5.4.1, this redundancy is necessary though.

This very frequent re-transmission of redundant data to a potentially large number of hosts would therefore severely impair the goal of the performance requirements in section 3.1.3. The scalability (DE.1) is affected, since the amount of bandwidth required for the message grows with the number of sensors. In case of failure recovery, complete re-transmission of those files would be inefficient and contradict the requirement RP.1.

It is therefore desirable to cache information locally at entities, since due to the high frequency of requests significant savings are to be expected. This raises the need to distinctly specify all of the contents in these data objects in their current version as it was described in section 4.3.2. It therefore should be possible to specifically instruct them about which data object they should use based on an object identifier.

Since the signatures of the data objects already are unique and un-forgable cryptographic hashes, they will be used for this purpose. Instead of sending the object itself, therefore only the identifier is included in communication. It can easily be verified by clients and files that contain false information can instantly be identified by checking the signature. At the same time, this mode allows them to verify the origin of the information that they base their authorization decisions on.

If an entity in the network now receives an identifier that it doesn’t yet have in it’s storage, it can request the specified object from the object storage as specified in section 5.5.1.4. Information can therefore be cached locally, while the entity does not have to trust the prevailing validity of the contents unless it advised to do so through the identifier in a message.

Due to the sheer size of the range of possible hashes, guessing a valid ID is practically impossible. As long as a brute force attack is not feasible, no additional security measures are necessary. If an adversary learns about a valid ID from an entity, this entity could also directly present them the data object. It therefore would make no difference, if additional authentication would be used.

5.4.3 Operability in dynamic environments

Furthermore, the request processing should take the dynamic setting into account. Possible implications of this setting are temporary connection disruptions and problems (RP.1), or failures of involved systems like peers or gateway components (RP.2).

Since the gateway may frequently lose connection to other entities (RP.1), it does not keep an connection open with the peers until they can provide the results. After forwarding the authorized requests to peers, it therefore is continuously accepting result messages from nodes in order to store them until retrieved by clients.

A connection disruption therefore does not cancel the entire request, the peer just has to keep on trying to provide the result to the gateway until it is successful. In case of failure or loss of data at the request processing component, the authorized request can be re-transmitted by the client and then be forwarded by the gateway.

Without further measures, allowing this behavior would make request flooding possible: if peers already has answered to a request before, a new SMC computation round could be highly expensive and possibly provide unwanted insights, if an attacker can request sensor data with a high temporal resolution.

The peer should therefore keep track of answered requests. Since the id of this authorized request stays the same, the peers can identify this duplication and re-transmit their former response. This approach therefore allows to save resources through an efficient retry mechanism.

These mechanisms try to guarantee eventual reception of the results. The collection of data from peers can take a non-specifiable amount of time though, since they are possibly deployed in a context with high-latency and slow connections. In case of permanent failure of a peer, the request will never be possible to be entirely completed.

The gateway therefore provides an interface for collection of results, that provides information about the current progress of the request such as successful peer responses, veto messages or failures. At the same time, it provides an intermediate result that is based on the responses that have been received so far. This procedure allows dynamic job execution as demanded in requirement RP.2.

5.4.4 Post-processing of received results

The gateway gradually receives results from peers as described in the previous section 5.4.3. When peers query the current status of their earlier information request, as demanded in requirement RP.2 they should receive information even if some peers are not done yet or have failed.

In order to understand the reliability and completeness of the provided result, clients should receive some statistics. The possible options for the status of a information requests to peer are the following:

- successful response
- the peer has accepted the request but the response is still pending
- veto
- failure, e.g. due to connection problems

Furthermore, when secure multiparty computation is used for calculating the responses of peers, they may all provide an identical response. When forwarding results to clients, those duplicates provide no additional value and should therefore be removed before.

The gateway therefore needs to apply some post-processing to the received responses, before forwarding them to clients.

5.4.5 Security considerations for information transmission

The adversary models have described in section 2.3.1.2, while the implication of each for the security (section 3.1.1) and privacy (section 3.1.2) of peers has been shown in section 2.3.2.

Therefore in order to ensure privacy, the communication between the gateway and the peers and clients has to be secured. Hereby the different attacker models require various precautions, which will be described below in conjunction with their purpose.

5.4.5.1 Authenticity of communicating entities

Section 4.2.3 has shown different approaches for a secure authentication solution, which is necessary due to the attacker models AM.4-AM.6. In order to not leak any information by communicating with false devices, authentication should always be required from both of the communication partners.

Certificate authentication as discussed in section 4.2.3.3 provides a large number of advantages and will therefore be used in all communication of the gateway solution, that is not covered by the connected protocol. The TLS protocol, of which the purpose is described in further detail in the subsection 5.4.5.2, supports certificate authentication for the server and the client and will therefore be the way of implementing X.509-authentication.

The X.509-certificates require a public-key infrastructure, since in the Dolev-Yao attacker model (AM.4) a trust-on-first-use approach is not feasible. This is due to the fact that the attacker could provide their own certificates to communication partners in an man-in-the-middle-attack in an attempt to wiretap their communication. Each of the communication entities is therefore equipped with a certificate authority certificate before deployment, which will be used to verify the authenticity of the certificates that any communication partners may present.

Since any other attribute may change over time, the certificate and it's public key of any entity serves as their only stable identifier. The advantage of such a solution is, that basing the peers identity on cryptographic primitives promises a high level of security as long as these primitives hold secure. The identifier therefore is as secure from being forged as the primitive is according to the current state of knowledge.

5.4.5.2 Confidentiality and integrity

Any of the attacker models AM.1-AM.6 constitute a threat to the confidentiality of transmitted data. In order to solve this problem, encryption can be used in order to keep information secret from unauthorized entities.

A special challenge in this context is the privileged position of the gateway in the protocol, that it needs in order to provide it's functionality. Therefore a differentiation in the cryptographic solutions needs to be made for the different attacker models in order to ensure privacy and security in all of them.

Transport security

If transmitted data can be intercepted from e.g. a wireless channel by an adversary of any of the models AM.1, AM.2, AM.4 or AM.5, they should not be able to find out the raw contents of the data. This means, that traffic has to be encrypted so that the contents are secret for any unauthorized entity even when transmission happens over an insecure network layer.

All communication over the interfaces described in this thesis happens above TCP as a reliable network communication protocol. Therefore the TLS protocol as it was described in section 4.3.5.2 is an ideal candidate for the purpose of providing a confidential and integrity-protected channel for communication with the gateway. Therefore peers and clients use HTTPS for communication with the gateway.

End-to-end encryption between clients and peers

The transport encryption only provides confidentiality for the two direct communication partners in a TCP connection. Since the purpose of the gateway is hiding peers and connected services, such as an authorization provider behind a central gateway, it is not desired that direct communication between clients and these other entities takes place. In order to cover the attacker models AM.3 and AM.6 therefore a special provision needs to be made.

This can be solved by using end-to-end encryption. Any information that should not be disclosed to the gateway can be encrypted with the public key of the communication partner before sending it to them through the gateway.

The solution used in this thesis is the JSON Web Encryption standard as discussed in section 4.3.5.1.

5.5 Protocol phases and content of messages

The different messages, that can be seen on an abstract level in figure 5.1, will now be defined in more detail. Due to the dynamic nature of the environment and in order to fulfill the requirement RP.1, the different phases of the protocol were designed to be independent from each other. The communication flow that is described below is therefore just exemplary, while the different phases can happen isolated from others, depending on the data that is already present at the different communication entities. Each of the phases described in the following sections is coherent in itself though.

The sections start with a figure that shows an overview of the messages of the respective phase. The messages and their contents are then described, while they are each referenced by their ID number as it can be seen in the figure that provides the overview of the protocol at the beginning of the description of each phase. Exemplary contents such as signatures and times are provided. In cases where a fixed set of possible values is pre-defined, all possible values are given.

5.5.1 Directory services

The gateway directory accepts, stores and provides information about the current state of peers. As a secondary function, it stores static information for retrieval in other requests and for logging.

5.5.1.1 Pairing and peer update

The process of registering new peers, updating them or deleting them as it was described in requirement DS.2 and section 5.2.2 can be seen in figure 5.2:

REST endpoint: /directory/pairing/<string:peer_identifier>

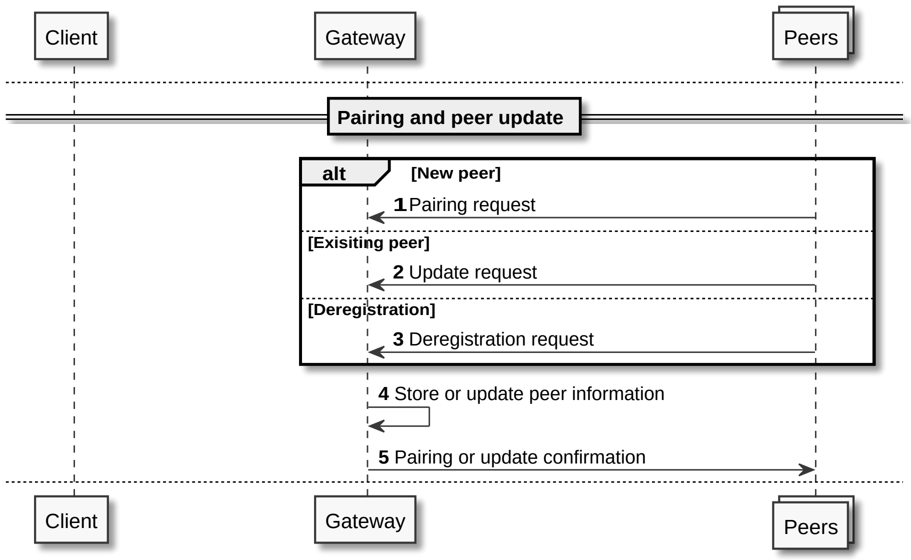

Figure 5.2: Peer pairing and update

1 Pairing: All information about peers comes from the pairing process, where peers announce their identification proofs and share information about their setup in a request.

HTTP request method: POST

2 Update: In order to update their information, a peer sends a message that equals the format of a pairing message with a "Update" type.

HTTP request method: PUT

3 De-registration: De-registration from the gateway can be achieved by sending an update request with the "deregistration" type, while it contains empty fields for the attributes, the capabilities and other metadata.

HTTP request method: DELETE

Since their contents are very similar, messages 1-3 share the same structure as described in listing 5.3. In case fields are not needed, such as the attributes during de-registration, they may contain empty values.

```
{
  "peer_identifier": "kndn....EhaG",
  "announcement_type": "pairing/update/deregistration",
  "announcement": {
    "available": "True/False",
    "timestamp": "2017-10-20T13:37:55.144Z",
    "current_address": "https://3buzkt6hd:30653",
    "peer_attributes": {
      "clearance": [9],
      "rooms": ["all", "meeting_room"],
      "floor_number": ["all", "3rd_floor"],
      "sensor_data": ["temperature", "light"],
      "...": ["...", "..."]
    }
  }
}
```

Listing 5.3: Pairing and peer update message

4 Store or update peer information: The gateway then processes the received information and subsequently stores it in the directory, updates an existing record or deletes it.

5 Pairing confirmation: Afterwards a confirmation as seen in listing 5.4 is sent to the peer.

```
{
  "peer_identifier": "kndn....EhaG",
  "originating_request_signature": "7820...7d5e", # Signature of
      # the pairing, update or de-registration message
  "request_status": "success/error",
  "announcement_status_message": "Free text"
}
```

Listing 5.4: Pairing and peer update confirmation message

5.5.1.2 Keepalive

Section 5.2.2 described a keepalive process, that lets peers communicate a subset of their current state such as the address and availability to the gateway as demanded in requirement DS.2. This process can be seen in figure 5.3:

REST endpoint: /directory/keepalive/<string:peer_identifier>

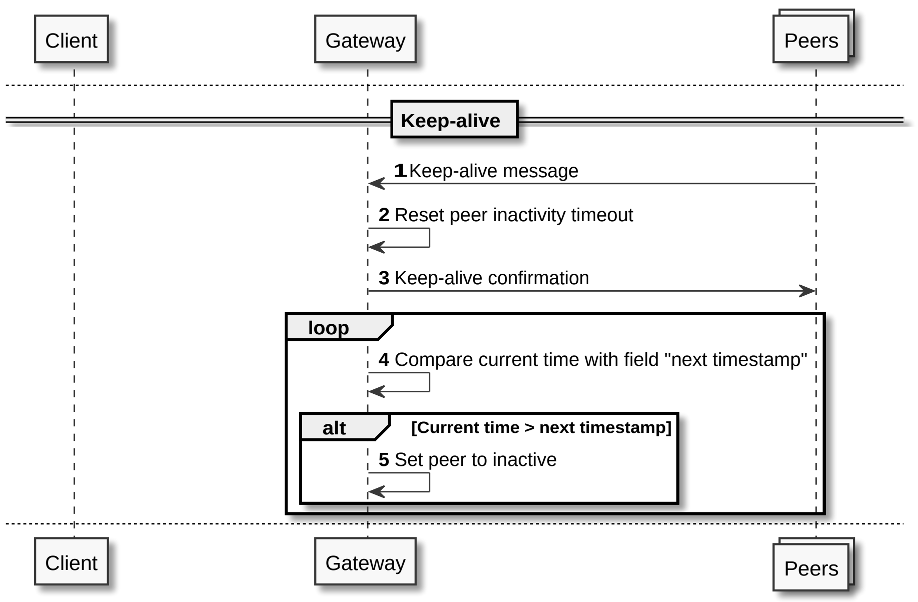
Client
Gateway
Peers
Keep-alive
1 Keep-alive message
2 Reset peer inactivity timeout
3 Keep-alive confirmation
loop
4 Compare current time with field "next timestamp"
alt
[Current time > next timestamp]
5 Set peer to inactive
Client
Gateway
Peers

Figure 5.3: Peer keepalive

1 keepalive message: The peer periodically sends a message containing information it about its availability to the gateway.

HTTP request method: PUT

```
{
  "peer_identifier": "kndn....EhaG",
  "available": "True/False",
  "current_address": "http://localhost:36824",
  "keepalive_timestamp": "2017-10-20T13:37:55.144Z",
  "next_keepalive": "2017-10-20T13:56:56.415Z"
}
```

Listing 5.5: Default keepalive message

2 Reset peer timeout: The gateway then updates it's local peer record with the values contained in the message as it can be seen in listing 5.5.

4 Compare current time with next timestamp field: The cleanup worker of the gateway periodically checks, if the time for the next keepalive that the peer has previously defined has already passed.

5 Set peer to inactive: If the gateway has not received a keepalive message until the time that the peer has previously announced to do so, the record of the peer is set to inactive.

5.5.1.3 Directory queries

Section 5.2.3 described an information query of clients, that they can use to learn about available data and peer groups. This data is requested and results are provided as seen in figure 5.4.

REST endpoint: /directory/peer_info

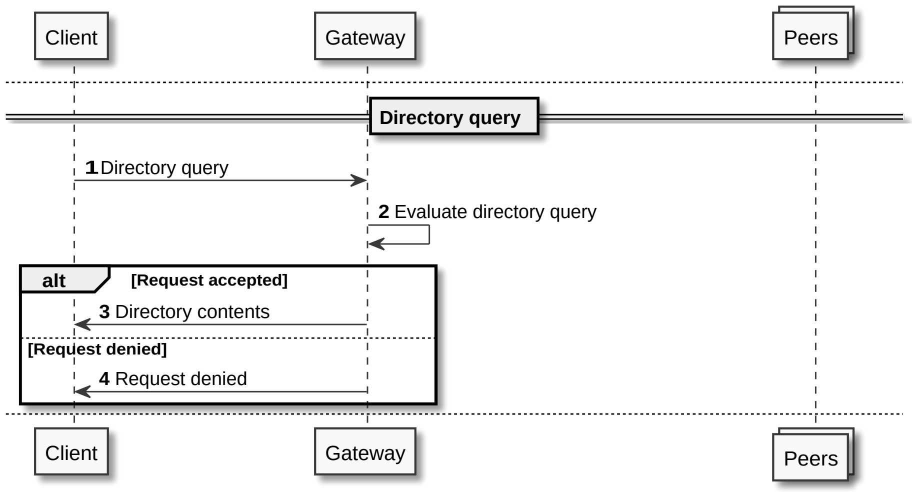

Figure 5.4: Directory query

1 Directory query: In the initial directory query, the client defines the information it desires to obtain with a message as seen in listing 5.6. The defined group comprises all peers in a building up to clearance level 9, which provide light as their sensor data.

HTTP request method: GET

```
{
  "client_identifier": "exlE....xD5x",
  "queried_information": {
    "clearance": [9],
    "rooms": ["all"],
    "floor_number": ["all"],
    "sensor_data": ["light"]
  }
}
```

Listing 5.6: Directory query

2 Evaluate query: The gateway then needs to evaluate the received request. Apart from obtaining the data from the database, it must be aggregated into the different groups so no peer-specific information can be derived.

3 Directory contents: The contents of the directory are then provided to the clients in the format that is specified in listing 5.7.

```

{
  "request_status": "successful/denied/failure",
  "query_results": {
    "clearance": {
      "3": 8,
      "5": 4,
      "9": 6,
    },
    "rooms": {
      "kitchen": 5,
      "bathroom": 3,
      "open_space": 3,
      "ceo_office": 2,
      "meeting_room": 5
    },
    "floor_number": {
      "1st_floor": 4,
      "2nd_floor": 6,
      "3rd_floor": 5,
      "4th_floor": 3
    },
    "sensor_data": {
      "light": 18
    }
  }
}
```

Listing 5.7: Directory contents

4 Request denied: In case the gateway decides to not present the requested information to the client, it sends a message with the same format as seen in listing 5.7. All fields are left blank except for the message which states the error or request denied code.

5.5.1.4 Request static data from gateway by ID

As described in section 5.4.2, messages may contain identifiers instead of actual data in cases such as certificates, authorization grants and the "original_request" field in an authorized request. If an entity receives a message that contains such an identifier and does not have the according object cached locally, it needs to request the information from the gateway storage that was described in section 5.2.4.

Figure 5.5 shows the process of obtaining a document by it's ID, which has been received within a different request.

REST endpoint: /directory/blobs/<string:blob_identifier>

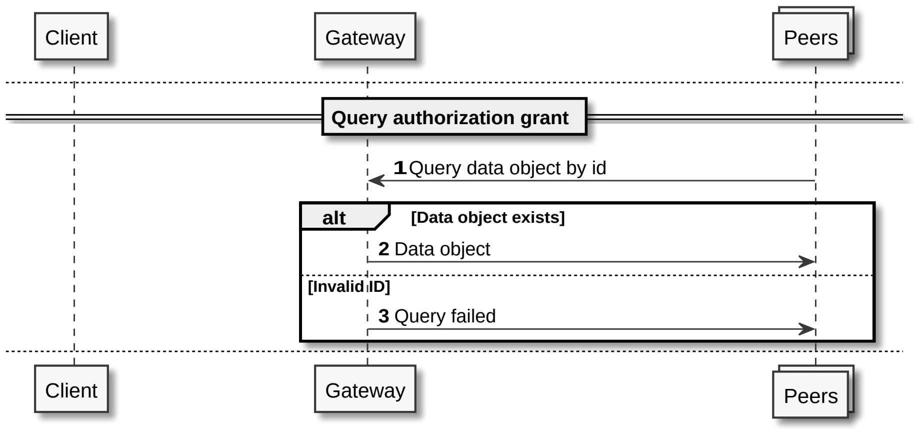

Figure 5.5: Requesting static data from gateway

1 Query data object: The message, that entities send to the gateway in order to request data objects, is described in listing 5.8.

HTTP request method: GET

```
{
  "blob_identifier": "dTO1....B2qc",
  "origin_request_identifier": "oWrv....mA1g",
  "blob_type": "certificate/original_request/authorization_grant"
}
```

Listing 5.8: Querying authorization grant by ID

2 Data object: In case of a valid identifier, the gateway provides the data object to the peer in a message as specified in listing 5.9.

```
{
  "request_status": "success/denied",
  "blob_identifier": "dT01....B2qc",
  "blob": {...}, # Either JWS or JWK container
  "blob_type": "certificate/original_request/authorization_grant"
}
```

Listing 5.9: Data object query response

3 Query failed: In case the query failed due to e.g. an invalid ID, the gateway sends a message as seen in listing 5.9, while the blob field is left empty.

5.5.2 Access control

The access control component provides an interface for posting authorization policies and requesting authorization grants. The logic and different attributes of both are described in further detail in section 5.3.1.2.

5.5.2.1 Post authorization policy

Figure 5.6 shows, how an XACML authorization policy can be posted to the gateway.

REST endpoint: /access_control/authorization_grants

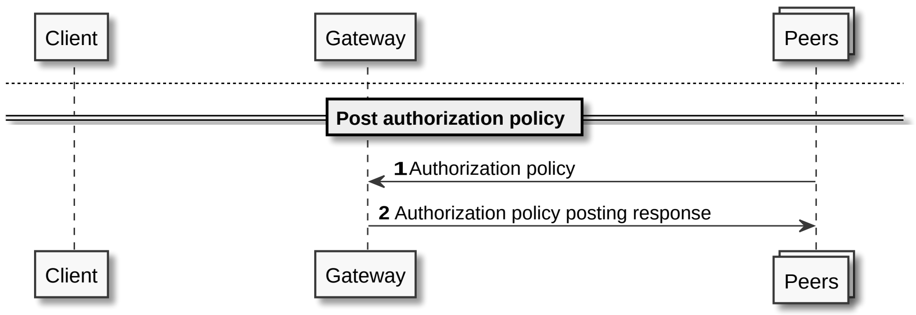
Client
Gateway
Peers
Post authorization policy
1 Authorization policy
2 Authorization policy posting response
Client
Gateway
Peers

Figure 5.6: Posting an authorization policy

1 Authorization policy: The peers, or a party that they trust, can post an authorization policy container message containing an XACML authorization policy, as it is seen in listing 5.10. The JWS container of the message contains the information about the original poster by providing the key ID.

HTTP request method: POST

```
{
  "PolicySet": {...}, # An example for possible attributes and data
     # types can be found in listing 5.2
  "timestamp": "2017-10-20T13:37:55.144Z"
}
```

Listing 5.10: Authorization policy container message

2 Authorization policy posting response: The gateway then returns a status for the authorization policy post.

5.5.2.2 Obtaining an authorization grant

Figure 5.7 shows the process of requesting an authorization grant from the gateway.

REST endpoint: /access_control/authorization_grants

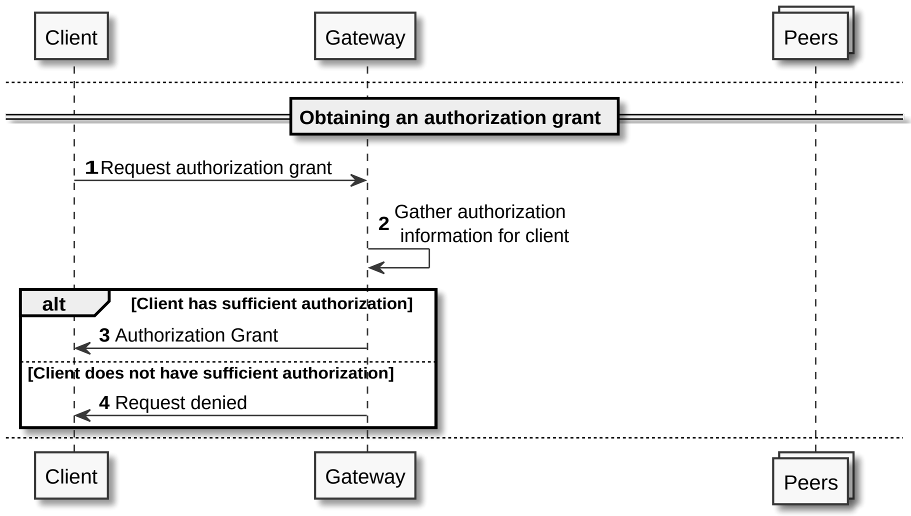
Client
Gateway
Peers
Obtaining an authorization grant
1 Request authorization grant
2 Gather authorization information for client
alt
[Client has sufficient authorization]
3 Authorization Grant
[Client does not have sufficient authorization]
4 Request denied
Client
Gateway
Peers

Figure 5.7: Obtaining an authorization grant

1 Request authorization grant: The client can send a request for an authorization grant as it is described in listing 5.11. The contents of the XACML part of the message follow the XACML standard and it's JSON profile.

HTTP request method: GET

```
{
  "client_identifier": "exlE....xD5x",
  "grant_request_xacml": {...} # An example for possible attributes
     # and data types can be seen in listing 5.2
}
```

Listing 5.11: Request to issue an authorization grant

2 Gather authorization information and check validity: The gateway now has to process the XACML access request and compare it to the stored XACML policies. Based on this information, it decides if the request is valid and should be processed further.

3 Authorization Grant: If the client can prove sufficient authorization, the gateway provides it with a message containing the authorization grant, as it can be seen in listing 5.12.

```
{
  "request_status": "successful/rejected",
  "request_status_message": "Free text",
  "authorization_grant_xacml": { # For an example see listing 5.2
    "...": "..."
  },
  "valid_until": "2018-03-20T13:56:56.415Z"
}
```

Listing 5.12: Authorization grant container message

4 Request denied: In case the client does not have sufficient authorization rights for the requested grant, the gateway sends a request denied message containing the XACML authorization decision It has the same format as the message in listing 5.12, while the valid_until field can stay empty.

5.5.3 Request translation

The following protocol phases provide the request translation service.

5.5.3.1 Information request

The central purpose of the gateway is allowing clients access to data that is present at the peers. Therefore requests are made at and executed through the gateway, which translates them to the integrated target protocol. The according process works as seen in figure 5.8.

REST endpoint: /request_translation/info_requests/<string:original_request_signature>

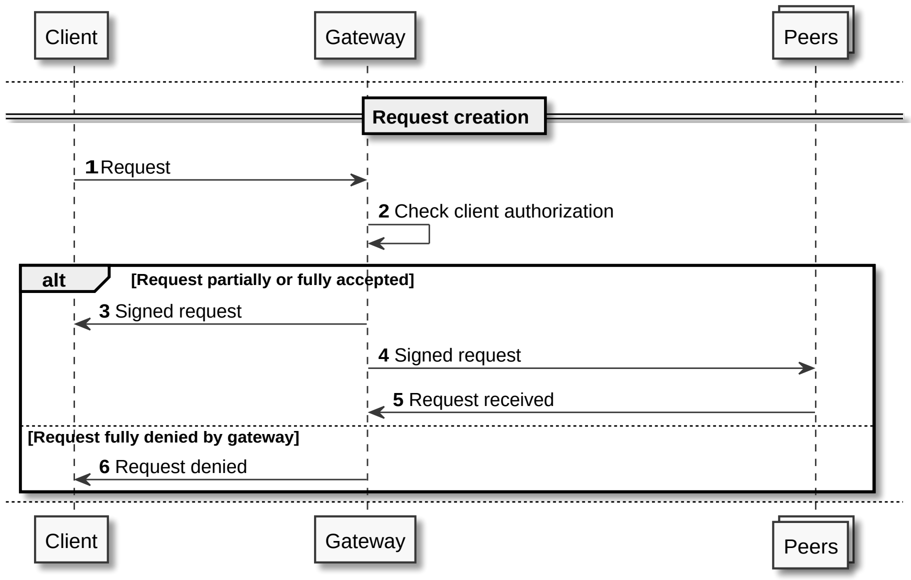

Figure 5.8: Information request

1 Initial request: The format of the original client request can be seen in listing 5.13. It specifies the attributes of the queried peer groups and the client and grant identifier so that peers can later verify the request after it has been forwarded.

HTTP request method: POST

```
{
  "requested_information": {
    "clearance": "9",
    "rooms": "meeting_room",
    "floor_number": "all",
    "sensor_data": "light"
  },
  "client_identifier": "exlE....xD5x",
  "authorization_grant_identifier": "mBm7....pbhx",
  "original_request_timestamp": "2017-10-20T13:37:55.144Z"
}
```

Listing 5.13: Initial information request

2 Check client authorization: Check, if the client was able to present a valid authorization grant, that is sufficient for the requested group-attributes.

3 Signed request for client: If the checks were successful, the gateway compiles all relevant information in a signed request as seen in listing 5.14 and sends it to the client. The purpose of this is, that the client may resend the signed request later-on. This is useful for efficient failure recovery, e.g. in case of data loss on side of the gateway.

4 Signed request for peers: A list of applicable peers is obtained from the database through the directory component based on the attributes that the client has provided in it's request. The gateway then forwards the signed requests to these peers as it can be seen in listing 5.14.

REST endpoint on peer side: /peer_data/info_requests/<string:request_id>

HTTP request method: POST

```
{
  "original_request": "c47b...7e0a", # JWS signature of the
     # original request seen in listing 5.13
  "gateway_timestamp": "2017-10-20T13:37:55.144Z",
  "request_status": "accepted/denied/failure"
}
```

Listing 5.14: Signed request

5 Request received message: The client then answers with a request received message as seen in listing 5.15.

```
{
  "request_status": "accepted/veto/failure"
}
```

Listing 5.15: Request received message from peer

6 Request denied: If the client could not provide sufficient authentication or specified invalid attributes, a message as seen in listing 5.14 can be sent with the according request_status field.

5.5.3.2 Collection of results from peers

Due to possible loss of connection between peers and the gateway, the collection of results from peers is a process separate from the initial information request that was described in section 5.5.3.1. The background to this decision is described in further detail in section 5.4.3. Figure 5.9 shows the collection process.

The support for end-to-end encryption of results was described in section 5.4.5.2.

REST endpoint: /request_translation/peer_response/<string:original_request_signature>

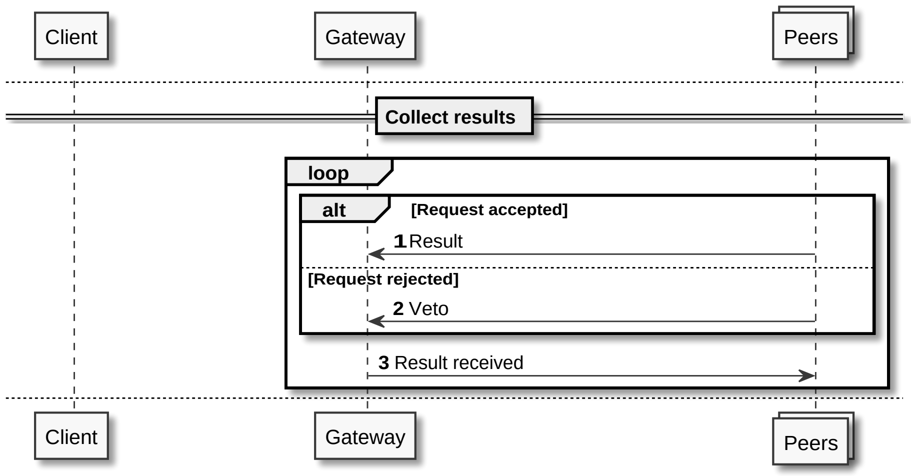

Figure 5.9: Collection of results from peers

1/2 Result or veto message: After the peer has processed the original request, it provides a message containing either the results or a veto to the gateway. The format of this message can be seen in listing 5.16.

It should be especially noted, that the actual results contained in the message (field "result_message") may contain the results in encrypted form in a JWE container.

HTTP request method: PUT

```
{
  "request_identifier": "oWrv....mA1g",
  "peer_identifier": "kndn....EhaG",
  "peer_response": "success/veto/error",
  "peer_computation_results": { # Dictionary may contain any
      # information. Encrypted information represented in a
      # JWE container is possible.
    "temperature": "30"
  },
  "timestamp_computation": "2017-10-20T13:37:55.144Z"
}
```

Listing 5.16: Response to information request of peers

In case of a veto, the peer can leave the field "peer_computation_results" empty.

3 Result received: After receiving the results, the gateway confirms this with a confirmation message as seen in listing 5.17.

```
{
  "request_status": "received/failure"
}
```

Listing 5.17: Peer response received message sent by gateway

5.5.3.3 Retrieval of results from gateway

After a client has made a successful information request as described in section 5.5.3.1, they can query the status of this request until the full results are available. The gateway should therefore provide information about the progress of the request and then present either intermediate or final results. Figure 5.10 shows the result retrieval process.

REST endpoint: /request_translation/info_requests/<string:original_request_signature>

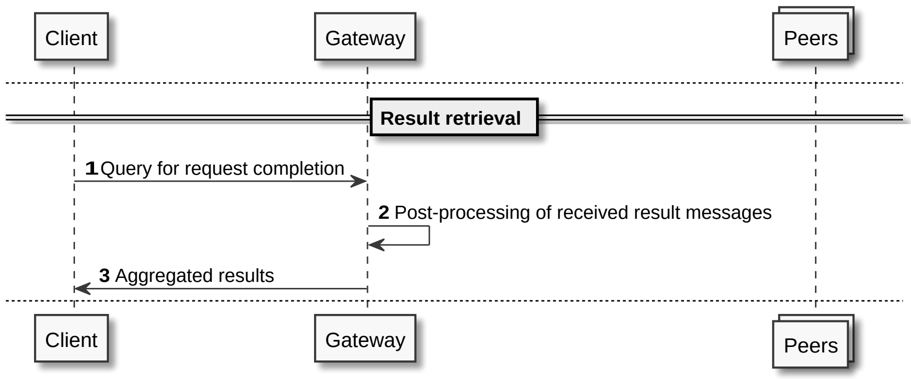
Client
Gateway
Peers
Result retrieval
1 Query for request completion
2 Post-processing of received result messages
3 Aggregated results
Client
Gateway
Peers

Figure 5.10: Retrieval of results from gateway

1 Query for request completion: The client can query about the status of the request as specified in listing 5.18

HTTP request method: GET

```
{
  "request_identifier": "oWrv....mA1g"
}
```

Listing 5.18: Request for results

2 Post-processing of results: As long as results have been received from some or all peers, the post-processing as described in section 5.4.4 can be applied.

3 Aggregated results: The gateway then provides the results to the client as specified in listing 5.19. The field *results* is a list, since in the use case of DecADe each peer could provide different results that need to be specified separately. In case of SMC, only one value should be contained in the list, since only one aggregate exists and the duplicates were removed during post-processing.

```
{
  "request_status": "successful/in_progress/veto/failure",
  "results": [{...}, {...}, ...], # List of dictionaries as
     # provided in field "peer_computation_results" from listing
         5.16
  "request_details": {
    "last_response_complete": "2017-10-20T13:37:55.144Z",
    "request_identifier": "oWrv....mA1g"
  },
  "request_statistics": {
    "success": 15,
    "accepted": 2,
    "veto": 2,
    "failure": 3
  }
}
```

Listing 5.19: Aggregated results

Chapter 6

Implementation

The description of the design of the system in chapter 5 has shown, how different technologies as described in chapter 4 can be used in order to solve the problem and the requirements that were described in chapters 2 and 3. The outcome of this was a precise description of an architecture and protocol that solves these problems.

This chapter will now use these results and show the feasibility of the proposed approach and protocol and how it can be implemented in order to provide a solution for privacy preserving data retrieval from distributed and dynamic systems in a real-world scenario. It describes some implementation decisions that were made in the exemplary solution that was developed in the course of this thesis.

First, section 6.1 will show the approach of modularization of the gateway and how the different components can be integrated with each other for simple and effective deployment. Then the technological solution and the final scope of the different components of the gateway are described in sections 6.2 to 6.4. Section 6.5 will finish the chapter by describing the exemplary mock implementations for the peer and client systems, which can be used to show the interaction of the system and network as a whole.

All software was developed in the programming language Python 3, while few bash scripts were used for deployment of components. For each of the components it is listed, which modules in the subfolder *components* of the implementation provide the described functionality. If external software was used in the form of docker images, they also will be listed. The interaction between the containers will be discussed in section 6.1.

6.1 Modularization of system components

Adhering to the architecture of the gateway derived in section 2.4, the gateway was developed as a set of components that are as independent of each other as possible. The upsides of this approach were described in section 5.1.3. In order to fully leverage these advantages, the different functional parts of each component were further modularized into sub-components that each provide one specific functionality. Each of the following subsections describes one of them.

The modularization is reflected in the file structure of the code, which is separated by the different components and their sub-components. Additionally, clear interfaces are defined through functions that can be called by the request processing or other sub-components.

Apart from a clean-up worker that is started in a separate thread on start up, at runtime the components developed within this thesis are not distinct though. The request processing component waits for requests and then takes care of the interaction between the different sub-modules.

In the developed system, existing solutions were used and integrated when possible. Therefore in some cases, components draw on external programs in order to provide functionality though. These are each run as a separate instance and waiting for communication that is initiated by the gateway. These implemented components and how their functionality is integrated in the gateway will be described in the section of the respective gateway component. These cases are marked with a note that specifies the container image of the software. Communication with any additional software, that was not developed in the course of this thesis, happens over TCP connections.

All of the software developed in this thesis can optionally be deployed using the docker platform.In this case, all external software and libraries don't have to be installed manually. While it is also possible to run the software directly on a system without such virtualization, it significantly reduces the setup effort. The docker-subdirectory of the code contains the information for the creation of the images, while the file docker-compose.yml specifies their setup and interaction between each other.

In summary, the gateway is modularized as shown in figure 6.1. For each component, the modules are grouped by their functional scope.

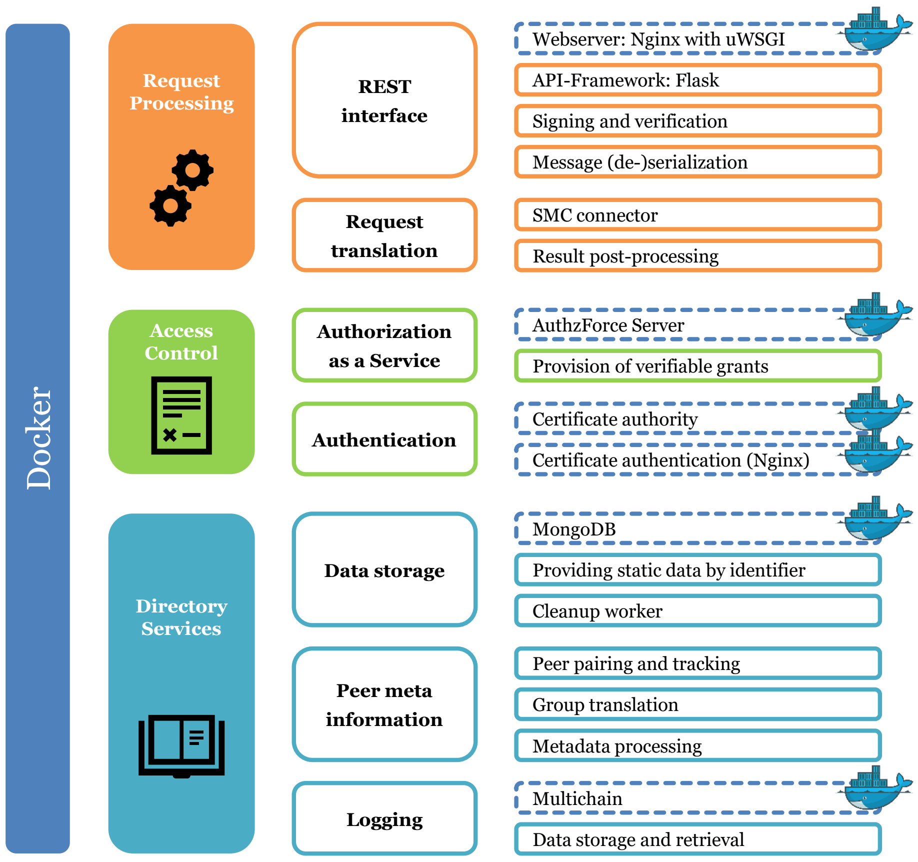

Figure 6.1: Overview of gateway components and their modules (software provided by external docker images marked with logo

It is important to stress, that not all of the described sub-components need to be operated by the same organization or on the same physical system. For various possible reasons, it could be decided to transfer their operation to a third party or one of the entities involved in the system other than the gateway provider. It will be discussed for some components, if such a form of outsourcing could be useful.

6.2 Access control

The design of the access control component was discussed in section 5.1.3.2.

Implemented in modules:
AuthForce/
smc_gw/access_control/

6.2.1 Authorization as a service

A number of different access control solutions support the XACML standard. Integration of them can be achieved by the gateway through translating requests that it receives to the specific software. Implementation, interoperability and exchangability of those solutions is facilitated by the XACML standard through providing a profile for such REST interfaces including JSON message formats [39].

In this thesis, the AuthZForce Server was chosen for this purpose. [40] It supports multiple tenants, which allows to use the same instance of the software for multiple separate systems. This could e.g. be used, if one gateway should be used for several separated buildings (not in the currently implemented scope).

For a simple setup and usage process, the following steps are necessary. [41] All of the message formats used for this are defined in the XACML standard [19].

- First an authorization domain needs to be set up, which comprises all information added later such as policies and general settings.

  Implemented in modules:
  AuthForce/authforce_initialize.py

- Adding policies:

  - Policies can then be posted to the Policy Administration Point (PAP). It just provides the storage and takes care of assistant functions such as versioning, doesn't fulfill any other purposes though.

  - Since posting policies alone is not sufficient for them to be used in later authorization decisions, they need to be made known by adding them to the root policy.

  Implemented in modules:
  smc_gw/access_control/authorization_grants.py

- As soon as appropriate authorization policies have been added, an authorization request can be sent. The server responds to this with an authorization decision message, which is used in the authorization grant.

  Implemented in modules:
  smc_gw/access_control/authorization_grants.py

Since just the authorization decisions by themselves don't provide access control, the gateway needs to assist in enforcing them. This means to deny a request of a client if the authorization decision was negative or if authorization grants are invalid due to age. Additionally it needs to check plausibility of the requests forwarded to the authorization provider. Examples for this are comparing the client identifiers in the XACML documents to the certificates that are actually provided during requests.

Due to the clear separation of functionality and due to the critical information stored in this component, it is an obvious candidate for provision by a more trustful third party or the owner of the peers. Since the posted authorization documents are secured by signatures, little trust is required from the authorization provider though, because the source of the authorization can be verified and this information is cryptographically secure.

Some software provided by docker image:
authzforce/server:release-8.0.0

Implemented in modules:
smc_gw/access_control/authorization_grants.py

6.2.2 Certificate authority

As a form of identity management, section 5.4.5.1 suggested certificates that are based on a public key infrastructure. The certificate authority takes care of maintaining this infrastructure by providing certificates and revocation services.

The certificate authority has the central position in the trust model of the designed system. It therefore most be ensured, that it only provides certificates with the correct roles to entities that are entitled to have the position that the certificate attests. It therefore presents a candidate for being operated by somebody other than the gateway provider.

Due to the hierarchical structure of X.509 certificates, it would also be an option to define different trusted certificate authorities that each only have validity for their scope, e.g. all of the offices of a company in a building. The peers could then trust only certificates that were issued by their trusted certificate authority. Otherwise the peers could also be equipped with a list of trusted certificates (e.g. of their owners). They then only trust authorization documents, that were signed with a key that belongs to one of these certificates.

Since key management was not in the central scope of this thesis, only a minimal certificate authority was implemented using the standard tool OpenSSL [42] and some scripts. It only possesses one root certificate and provides all entities involved in communication with a certificate and the according private key.

Some software provided by docker image:
paulczar/omgwtfssl

6.3 Request processing

Providing most of the functionality of the gateway as seen by the clients, the design of the request processing component was discussed in section 5.1.3.3.

Implemented in modules:
smc_gw/request_processing/

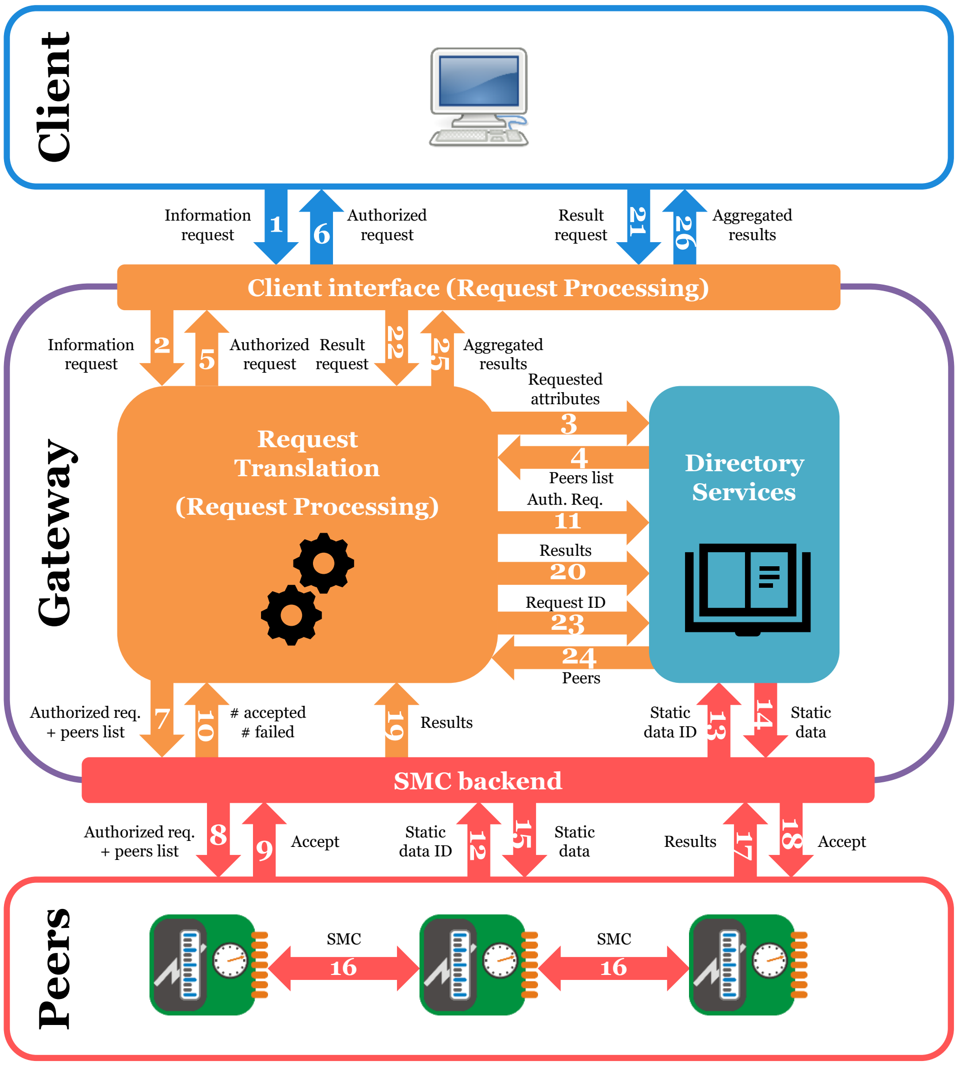

Figure 6.2: Overview of the information request phase

Figure 6.2 shows an overview of the implementation of the different phases of an information request, that are handled by the request processing component. The numbers will be referred to in the respective subsections.

6.3.1 Client and peer interface

The client interface (orange box in figure 6.2) provides the central API that then passes requests (#1-#2 in figure 6.2) to the request translation component of the gateway.

The client interface uses JSON messages transferred over HTTPS for communication, as it is typical in interfaces that follow the REST paradigms. The JSON messages themselves are in the JSON Web Signatures format in order to provide solutions to some of the privacy and information security requirements as discussed in section 5.4.1. The implemented interface endpoints are defined in section 5.5.

The Python framework Flask [43] was used in conjunction with the extension Flask-RESTful [44]. The developed web service must be delivered by a web server, that supports server side scripts and HTTPS with client-side authentication. In this thesis, Nginx[1] was used together with uWSGI[2]. JOSN Web Signature files are generated using the JW Crypto library[3].

Implemented in modules:
smc_gw/api/smc_api.py
smc_gw/api/smc_client_side.py
smc_gw/api/smc_peer_side.py

6.3.2 Request translation and SMC back end

The request translation component handles the communication with peers and forwards requests in a form that ensures transparency as described in section 5.4.1. If a different SMC back end should be used, section 6.3.3 provides information about the implementation through the request translation component.

Each request translation process contains the following phases:

Initial information request: If an initial information request as described in section 5.5.3.1 is received, the request translation component forwards the request using the following steps:

1. The client's authorization grant is checked for validity. It therefore must still be within the time specified by the valid_until field.

[1]https://nginx.org/
[2]https://uwsgi-docs.readthedocs.io/en/latest/
[3]https://github.com/latchset/jwcrypto

2. The attributes defined in the authorization grant need to match the attributes of the requested peer groups.

3. Then the authorized request is compiled, which most importantly contains the identifier of the original request.

4. The groups defined by the attributes in the request are translated to a set of peers as described in section 6.4.2 (#3-#4 in figure 6.2).

5. The authorized request is then forwarded to the client (#5-#6 in figure 6.2) and to all the peers (#7-#10 in figure 6.2) that were found in the query.

6. A record is created in the database, which uses the signature of the original request as the unique identifier as described in section 6.4.1.2 (#11 in figure 6.2). It contains the authorized request and the identifiers of the peers that the request has been forwarded to. The initial result is added for each of them, which is either "accepted" in case of success or "failure" in case the request could not be forwarded to them.

Implemented in module:
smc_gw/request_processing/client_information_requests.py

Collection of results: The results are then collected from the peers by waiting for their results as described in section 5.5.3.2. Upon reception of a response (#17-#19 in figure 6.2), the results are stored (#20 in figure 6.2) in the record that was previously created in the data base.

Implemented in module:
smc_gw/request_processing/peer_incoming_communication.py

Retrieval of results: The client can check the state of completion of the request (#21-#22 in figure 6.2) and receive either intermediate or final results (#25-#26 in figure 6.2) as described in section 5.5.3.3. The gateway compiles the information by applying the post-processing as it was described in section 5.4.4 (#23-#24 in figure 6.2).

Implemented in module:
smc_gw/request_processing/client_information_requests.py

6.3.3 Implementation interface for SMC back ends

The gateway provides an interface for implementing SMC back ends, which can be implemented as follows:

6.3.3.1 Forwarding a request

When the client interface receives a new request, it actively calls the request processing component in order to trigger communication with peers. All functions that are used for this purpose are contained in one file. By replacing the functional parts therein, the request can be forwarded using a different protocol.

The function *add()* specifies how to compile the message that is forwarded. The class *Peer_Request_Thread* is then used by this function and defines how the request is forwarded to each applicable peer.

In the exemplary implementation, the request is forwarded as a simple HTTP POST request containing the authorized request as described in message 4 of the protocol phase described in section 5.5.3.1. The forwarding itself was handled in a different thread for each peer, in order to ensure the number of peers has a smaller impact on the time that it takes to forward the authorized request to all peers.

Implemented in module:
smc_gw/request_processing/client_information_requests.py

6.3.3.2 Receiving a response

On the other hand, some functions can be triggered by the SMC back end in order to store results from a peer.

The functions needed for handling incoming peer-communication are contained in one file. Another form of receiving peer responses can be can be integrated by triggering the process of adding the response to the request processing component of the gateway. When the SMC back end receives a response from a peer, it therefore needs to call the function *inforeq_peer_response()* while providing all contents defined in listing 5.16 as a dictionary.

The function then stores the results of each peer in the database in the record that represents the information request.

The formats and process used in the exemplary solution are described in section 5.5.3.2.

Implemented in module:
smc_gw/request_processing/peer_incoming_communication.py

6.3.3.3 Pairing and keepalive mechanism

The process for pairing and updating peers or handling keepalive messages is contained in a dedicated file.

If a different pairing and peer information update process is desired, the functions *add()*, *update_details()* and *delete_peer()* can be called by the SMC back end. They expect a dictionary containing the contents of the messages as they are defined in the exemplary solution. The process and said messages can be found in section 5.5.1.1.

A different keepalive mechanism can be implemented by calling the function *keep_alive()*, which expects the contents found in listing 5.5.1.2. It then resets the keepalive timer for the peer, therefore keeping the peer marked as active until the specified time.

The process and formats used in the exemplary solution can be found in section 5.5.1.2.

Implemented in module:
smc_gw/directory_services/peers.py

6.4 Directory services

As described in section 5.1.3.1, the directory service component mainly has three tasks:

- Persisting static data that will be used in the protocol, as described in section 5.4.2 and storing peer metadata.
- Providing information that is derived based on this data as described in section 5.2.3.
- Logging requests in order to provide accountability as discussed in section 5.2.5.

Implemented in modules:
smc_gw/directory_services/

6.4.1 Storage and retrieval of unprocessed directory information

For simple storage of the information as it was described in section 5.2.4, the gateway can use a document-oriented database as described in section 4.4.2. In the context of this thesis this approach is especially useful, since any information is addressed by a single identifier as described in section 5.4.2 and data is present in the JSON format anyway. Each of the key-value pairs represents one object and the attributes that describe it, such as a request or a peer. Due to the simple structure and hierarchy of information contained in the design, solutions such as LDAP that was described in section 4.4.1 would incur significant complexity that is not needed.

In the implementation of this thesis, MongoDB was used. The following collections in the database smc_gw are created and used to store information.

Some software provided by docker image:
mongo:3

Implemented in modules:
smc_gw/directory_services/blob.py
smc_gw/directory_services/peers.py

6.4.1.1 Peer records

The database is used to store the metadata of peers discussed in section 5.2.1. The format of the records is equivalent to the JSON message used for pairing and updates without any further container, as it can be seen in listing 5.3. The peer_identifier is used as a unique identifier for the record.

Their state is not static but frequently changes which is why the record needs to be updated. In case of an update message, that defines a new set of attributes, the record is overwritten with the updated object. In case of a keepalive message, the according fields provided by the keepalive message are updated while leaving the rest of the record untouched.

Each record in the collection therefore represents one peer and contains all information that describes it.

Database collection: peers

6.4.1.2 Aggregation of metadata for directory queries

The request translation described in section 5.4.3 waits for peer responses to an information request and gradually receives them. They then need to be stored in the database as described in section 6.3.2 so that they can later be forwarded to the client.

The record that is used to store the information can be seen in listing 6.1.

```
{
  "_id": "ATU7....DS78",
  "authorized_request": {   # Identical to the message seen in
                            # listing 5.14 without a JWS container
    "original_request": "c47b...7e0a",
    "gateway_timestamp": "2018-01-16T10:46:45.161577+00:00",
    "request_status": "accepted"
  },
  "queried_peers": { # Each of the keys represents the identifier of
                     # one peer, that the request has been forwarded to
    "UVmg....mKDM": {....},   # Each of the values contains a dictionary
    "X68f....E3-R": {....},   # of the peer response as seen in
    ....                      # listing 5.16 without a JWS container
    }
}
```

Listing 6.1: Information request database record

Storing the response is a simple matter of updating the record by setting the contents of the response as an update to the key representing the peer.

Database collection: inforeqs

6.4.1.3 Quick retrieval of static information

Some components store static data in the gateway for quick retrieval. Entities request this data, when it was referenced in a different message from the gateway as described in section 5.5.1.4 (#12 in figure 6.2). This quick retrieval is in contrast to logging as it was described in section 6.4.4.

MongoDB simply returns the object queried by it's ID (#13-#14 in figure 6.2). The information is contained in their JWS container and passed on to the requesting entity without any further modifications (#15 in figure 6.2).

Database collection: blobs

6.4.2 Translation groups defined by attributes to peer

A group in this thesis is defined by a set of peers that contain the same attributes. No separate data structure is created in order to gather information about group membership. Therefore the attributes need to be translated to a set of peers when making an information request.

The format of the attributes field of each peer record is defined in section 5.2.1, while the format of a possible query is defined in listing 5.6. Since the number and contents of attributes defining a peer are not fixed, a general approach is needed. This can be done by iterating over every key-value pair of the dictionary of requested attributes, while adding each key as a "field" and each value as a "value" in a MongoDB query.

In case of special attributes that don't require exact accordance of values but more difficult comparisons, these need to specifically specified on the code side. As an example for this, the clearance field in the thesis specifies a maximum value that should contain all of the sensors of lower clearance levels as well, which is why a less then or equals comparison is used.

The referenced exemplary directory query would therefore result in a MongoDB query as seen in listing 6.2.

```
{
  'peer_record.announcement.peer_attributes.rooms': ['all'],
  'peer_record.announcement.peer_attributes.sensor_data': ['light'],
  'peer_record.announcement.peer_attributes.clearance': {'\$lte': 9},
  'peer_record.announcement.peer_attributes.floor_number': ['all']
}
```

Listing 6.2: MongoDB query for translation of attributes to a set of peers

In case of very large numbers of peers, which is not in scope of the use case of this thesis, an index can be created over the attributes to increase efficiency of queries.

6.4.3 Querying and processing of metadata

In this thesis, not all information that the database stores should be returned in it's raw format but rather in processed form as described in section 5.2.3. While the raw metadata of available peers can be queried from the database as described in section 6.4.2, it is aggregated as follows:

The attributes of all peers are stored in lists separated by their attribute identifier. Then the number of occurrences of each value per list is counted.

The dictionary returned to the client does not contain any information about the peers any more but only the attributes that describe their group.

Implemented in modules:
smc_gw/directory_services/content_queries.py

6.4.4 Auditable and unforgable logging

Section 5.2.5 has shown the advantages of logging requests in a blockchain. The solution developed in this thesis uses that approach by posting all of the log data in a private blockchain. It does not need to be operated by the gateway provider either and it is even possible to operate it on a number of peers that use the proof of work consensus algorithm to validate the contents of the blockchain.

This thesis uses the software Multichain, which provides a private blockchain and is compatible to the bitcoin protocols and formats [45]. In order to provide a medium that can be used for storage of information, a so-called stream "storeage_stream" is created in the blockchain during setup[4].

[4]https://www.multichain.com/developers/data-streams/

Communication with the daemon happens with commands transmitted over JSON-RPC[5] using a regular HTTP connection, as defined by the Bitcoin blockchain[6] with some extensions provided by MultiChain that support simple information storage[7].

Items can then be published to this stream by issuing a remote procedure call, while each item represents one data record. Each item is a key-value pair. The proposed solution uses the signatures of the JWS containers of messages as keys. The content is the JWS container itself.

The records can then be accessed by making a request to the stream, while specifying the desired object by its unique identifier.

Some software provided by docker image:
tilkal/multichain

Implemented in modules:
smc_gw/directory_services/request_logging.py
smc_gw/directory_services/multichain_storage.py

6.5 Exemplary implementation of peers and a client

In the course of the thesis, simple implementations for the functionality of the peer and the clients as they were described in section 5.1 were developed. They serve as an implementation reference of the protocol and can be used for testing and evaluation as described in section 8.

The peers don't provide any actual functionality or SMC protocol, but much rather just register to the gateway, accept requests and answer with a static response after a specified amount of time. JOSN Web Encryption support for the end-to-end encryption is provided using the JW Crypto library [8].

[5]http://www.jsonrpc.org/specification
[6]https://en.bitcoin.it/wiki/API_reference_%28JSON-RPC%29
[7]https://www.multichain.com/developers/json-rpc-api/
[8]https://github.com/latchset/jwcrypto

Chapter 7

Requirements evaluation

Chapters 5 and 6 have suggested a theoretical and a practical solution for the requirements that were described in chapters 2 and 3. This section will summarize for each of the categories defined in section 3, how the requirements were solved.

First, the non-function requirements are discussed on an abstract level in section 7.1. Then it is discussed in section 7.2, how the components of the gateway fulfill their specific functional requirements.

7.1 Non-functional requirements

The description of the three groups of non-functional requirements can be found in section 3.1.

7.1.1 Information security protection

ISP.1) Confidentiality: Confidentiality is provided by several means:

- *Communication* is secured through TLS between the gateway and other entities as described in section 5.4.5.2 and between clients as peers through end-to-end encryption as described in section 5.4.5.2.
- The confidentiality during *computation* of values is given by the used SMC back end.
- The confidentiality of data when responding to information requests is solved through privacy-preserving access control, as discussed further in section 7.2.1.

ISP.2) Integrity: The integrity is ensured on two levels. Communication between the gateway and all participants is secured through the message authentication codes of TLS as described in section 5.4.5.2. Due to the transparent request translation, reference to the authorization policies and authorization grants are included in some messages. The signatures contained in the objects requested from the gateway make end-to-end verification possible as described in section 5.4.1.

ISP.3) Authenticity: Authentication for all entities is handled through certificates as described in section 5.4.5.1. In communication between the gateway and the clients and peers, TLS uses them for client-side certificate authentication of connections. In the end-to-end verification, they are used to verify the authenticity of the JSON Web Keys that are needed to verify the JSON Web Signatures.

ISP.4) Accountability: Accountability can be ensured through checking the validity of all requests and logging them in a secure medium as discussed in section 5.2.5. All messages that are sent and received by the gateway are stored for later forensic purposes. Communication can be traced back to the participating entities, since the JWS containers of the messages provide the key identifier of the party that has sent them.

ISP.5) Non-repudiation: Signatures as they were explained in section 5.4.1 provide non-repudiation, since without knowledge of private keys of an entity they can't be generated. It therefore can be proven, that an entity in possession of the keys has sent a message containing exactly the contents that are stored in the log. Since all relevant messages contain a timestamp, it is not possible to claim a message has been replayed. Logging in the auditable append-only medium blockchain provides additional non-repudiation and makes logged messages safe from modification as discussed in section 5.2.5.

7.1.2 Privacy protection

PP.1) Transparency: Transparency of the requests that the processing component handles is provided by including all information necessary for understanding the source and purpose of a request as described in section 5.4.1. This information consists of references to the original request and the authorization grant, which in turn contains a reference to the verifiable authorization policy that it is based on.

PP.2) Unlinkability: Several measures ensure unlinkability of the information generated during communication:

- The client only receives a processed results rather than raw information.
- The connected SMC back end ensures that the data of single peers is not

disclosed.

- Result messages, authorized requests and grants contain no information about the specific peers that are part of a group, and therefore which peers have participated in a SMC round.

Unlinkability of information that can be queried is given through defining appropriate XACML policies as described in section 4.2.2.2. E.g. the current electricity usage should not be possible to be queried during certain times of the day.

PP.3) Intervenability: Peers are granted the option to deny and veto against requests as described in section 5.5.3.2. Since they are provided all relevant information as described in PP.1 of this section, they may send a veto message without making the results of the entire information request unusable.

7.1.3 Performance in dynamic environments

DE.1) Scalability: Scalability of the design is supported by avoiding redundant transmission of data where possible, as it is described in section 5.4.2. From the implementation side, the modularization allows scalability in case of performance limits on the hardware side 6.1.

DE.2) Extensibility: The possibility to add new kinds of devices and reflect their properties in a data structure is possible through the free definition of peer attributes as key value-list pairs as described in 5.2.3. The access control component supports this flexible definition of attributes as described in section 5.3.1.3.

7.2 Functional requirements

The functional requirements were grouped by the components of the gateway and can be found in section 3.2.

7.2.1 Access control

AC.1) Privacy-preserving access control: The different protection goals, that together make up the requirements needed for privacy-preserving access control, are solved by the authorization as a service component discussed in section 5.3. The access control component has a central role in the privacy model of this thesis. If a special solution is put in place apart from the system-wide solutions described in section 7.1, it is described for the specific context here.

ISP.1-ISP.5 are solved by the system-wide mechanisms described in section 7.1.1.

PP.1: Even though the authorization decision is first enforced by the gateway during request processing as described in section 5.5.3.1, the transparency makes it possible for peers to also enforce the decision themselves.

Transparency in access control is provided the attributes that authorization has been requested for. In case of a successful authorization decision, the source of the authorization policy is included. Accountability and non-repudiation for the contained information is then ensured by the general approach of verifying and logging messages. The exact process for this is described in section 5.4.1.

PP.2: Unlinkability is provided by including only a minimal necessary amount of information in authorization decisions. Specifically as stated earlier, they contain no information about the specific peers that are part of a group.

PP.3: Any authorization decision can be reviewed by data sources and potentially rejected as described in section 5.5.3.2.

AC.2) Dynamic authorization decisions: Section 5.3 describes, how attribute based access control following the XACML standard makes authorization decisions based on a variety of attributes possible. Various attributes can be included in the authorization policies and the decision process. The set of attributes is not fixed but can be dynamically defined.

AC.3) Stateless authorization grants: The stateless authorization grants are implemented by signing XACML authorization decisions as it was described in section 5.3.1. It contains all information that is necessary in order to validate it and trust the grant without any further communication.

7.2.2 Request processing

RP.1) Failure recovery and efficient retry: The central approach for providing resilience and failure recovery is the statelessness of the protocol. Section 5.4.3 discusses, how temporary connection interruptions are handled and information requests can be completed despite them.

Re-sending references to data objects with unique identifiers rather than re-transmitting them provides efficient retry as described in section 5.4.2.

RP.2) Dynamic job building and execution: The target systems of requests are not statically defined but dynamically built from attribute groups as they were explained in section 5.2.3.

Execution and provision of partial results in case of failure or veto decisions is possible as described in section 5.4.3. Peers which are not providing results therefore will not affect the information request as a whole.

7.2.3 Directory Service

DS.1) Extensible peer information storage: Section 5.2.3 shows how information about peers can be stored with a general approach that supports the the later definition of new attributes. This allows to extend the peer information storage in case devices with new types of attributes are added later.

DS.2) Pairing and tracking of peers: Peers can be tracked with a simple pairing and keepalive mechanism as described in section 5.1.3.1. It lets them announce their metadata and attributes and update them later-on.

This process can be replaced by an SMC back end in order to support other methods of discovery of services.

Chapter 8

Performance evaluation

The implementation that was described in chapter 6 is used for evaluation of the performance of the developed solution. The purpose of this empirical approach is to show the practical feasibility of the solution in addition to the theoretical discussion of requirements that were discussed in chapter 7. Such an evaluation is especially called for, since some of the requirements discussed in section 3.1.3 specifically stated performance goals that should be met in dynamic environments. In this context it will be discussed, how the different aspects demanded by the requirements are measured.

First the methodology that was used for evaluation is explained together with the measured metrics in section 8.1. Then the environment which was used for testing is described in section 8.2. The results that were obtained by carrying out those tests will then be discussed in section 8.3.

The performance evaluation should stay close to the initial purpose of the solution and provide an example that demonstrates the purpose of the used approaches in a way that is easy to grasp. Therefore the example of the smart building that was explained in section 2.1.1 will be used.

8.1 Methodology and metrics

The purpose of the developed system is to provide access to information to clients over a simple interface within a period as short as possible, under the boundary condition of ensuring privacy for peers. Acceptance of the system can only be achieved if it achieves the performance and simplicity goals visibly to potential users. From the point of view of clients, the most important metric is performance. Therefore the majority of the measurements happens using a blackbox approach and end to end tests.

Some measurements require a lower-level approach in order to investigate the source of the systems behavior. In those cases, more detailed results are obtained by making some measurements directly in the sub-components of the system. These measurements require modification of the applications code and may distort the test results. Therefore close attention was paid to keep the overhead of those measurements as low as possible.

Most of the communication described in section 5.5 consists of the communication between two partners and only has some database accesses on the gateway side as a result. The exception to this is the peer information request. Here the client sends a request to the gateway, which then uses a database query to find applicable peers and forward the request to them. They then perform their calculations and send the results to the gateway. All messages are logged in the blockchain by the request translation component.

Since this process contains all of the components and technologies that were developed in the course of the thesis, it will be the communication phase that is used for all measurements.

8.1.1 Test metrics

A lot of aspects of a system can be measured, while they have a varying degree of importance for the discussed use cases. It is therefore important to focus on aspects that are critical for potential users and evaluate if the system behavior doesn't restrict applicability in constrained environments that are given from the context that the system was developed for. The following metrics are therefore derived from the discussed use cases and scenarios.

Total response time: The responsiveness of a system is a central measure of usability, since for users it is directly visible during operation. For all requests it was therefore measured, how much time passes between initiation of request sending and the end of the reception of results.

Memory usage and computation effort: The collection of data in the specified use cases happens from sensor platforms. Since in comparison to full-scale computing systems those are designed for low cost and low energy consumption, the resources that they have access to are constrained. Therefore an interesting aspect is the consumption of those resources and how it develops under different circumstances that are tested.

Maximum request load handled: The number of application fields for smart devices and therefore their deployed number are quickly increasing, while the data they provide becomes easier to process. In order to assess applicability of the developed solution in other use cases and contexts, conclusions about it's performance boundaries have to be made. It will therefore be shown, which is the limit that the developed system and protocol can handle.

8.1.2 Parameterization of tests

Different aspects can influence the performance of a system. For example, a low number of peers and information requests in a reliable and fast network constitute good conditions for testing and promise desirable results for the different metrics. In the present evaluation, the influence of the following parameters on the previously discussed metrics were investigated.

Number of peers: The cost of each request increases together with the number of peers, since data is requested from a bigger number of sources. This number is therefore varied for each of the previously described parameters "frequency of requests" and round-trip delay time.

Due to shortcomings of the used test platform used in the setup, the peers had to be set up manually, which made it difficult to test an arbitrary number of peers. The number therefore was scaled up only to a number of entities that can be expected in the use case discussed in section 2.1.1, which was 30 peers. This number also constitutes a sufficient buffer considering the cost of the SMC protocols, which increases together with the number of peers. The thesis that was at the same chair tested the orchestation of up to 11 SMC peers [1].

Round-trip delay time: Sensor platforms may be deployed in environments where reliable connections are not given, for example wireless connections with high numbers of collisions or other indicators of signal quality. The round-trip delay time is used to simulate such connections by increasing the amount of time it takes to receive any kind of response to a signal.

The round-trip delay time is set on the operating system level, while pair-wise symmetric delays are used for every connection between two physical systems. The value is varied, while only one request per ten seconds is made in order to ensure the absence of correlations between the different requests.

Frequency of requests: This parameter simulates the influence of a higher number of information queries. Due to the physical limitations of a system, it has direct influence on all of the described test metrics.

This parameter can easily be set for each run of the client stub.

8.2 Testing environment

Meaningful results are achieved though deploying the solution on a hardware and network infrastructure that allows to simulate the behavior of scenarios that could occur in real use-cases. In the use cases that the system would be used comprise highly complex environments with very heterogeneous systems as seen in figure 5.1. Staying too close to such a setup would over-complicate testing though and lead to results that are hard to reproduce and interpret.

Such reproduction of results should be possible for later comparisons though, e.g. in regression tests that are used after modifications were made to the system. Additionally, not every aspect of such a complex network is relevant for the evaluation of the system.

The system is therefore deployed on a setup that constitutes a simplified model of the expected deployment infrastructures. It will therefore be elaborated, how such a setup can be emulated in a controlled testbed environment. For later reproduction of results on a similar system, this section will specifically describe the test setup.

8.2.1 Network setup

Simplifying the network setup of the testbed environment and reducing the number of physical machines that host the entities in the protocol's communication makes some assumptions about the performance implications of different possible options necessary. The decisions that were made are based on the following considerations.

Location of gateway components: The different components and modules of the gateway can be distributed over various machines in order to scale the solution. The communication between those and the management overhead incurs delays, that would distort the measurements though.

Therefore, all of the components of the gateway will be deployed on a single physical system.

Location of client stub: Most of the communication that happens in the proposed protocol is triggered by the clients in simple requests with a single response between the two communication partners client and gateway. While it is important to measure the total response time as described in section 8.1.1, measuring the cost of the communication between the client stub and the gateway would distort the results. This part of communication is not of major interest for the evaluation, since it is similar to classical systems and does not allow any conclusions about the performance of the protocol that does the actual collection of data. Furthermore, no computationally expensive tasks are handled by the client anyway, that could distort the results on the gateway side.

Due to these reasons, the client stub will be run on the same physical system as the gateway in order for them to be as close together other as possible.

Location of peers: As described at the beginning of the section, the main communication in the protocol happens when an information request is forwarded to the peers by the gateway since the amount of messages is amplified by the number of peers. It is therefore desirable to observe and modify the parameters of the network interface between the peers and the gateway.

The computational load on the side of each of the peers is very small though, since only a stub is implemented that returns a message that is almost always the same. Also, no communication between the peers is implemented.

The peers are therefore deployed on a single physical system, that is separate from the system that the gateway system is running on.

8.2.2 Hardware and software details

The described network setup leaves us with two physical systems, that will be used to simulate the participants. The application is containerized in order to define a system with easily reproducible state of software dependencies. Containerization incurs a slight distortion of network performance [46]. Due to the computational intensity of the developed system due to e.g. cryptographic actions, the effect of raw network latency in the range of only microseconds is does not significantly distort the results though.

Running an actual SMC protocol with high computational intensity on the peer-side would distort the results of the measurements of the gateway performance. The peers are therefore represented only by a stub that doesn't run any computation of results but provides fixed values. For the purpose of the tests, expensive cryptographic functions were disabled. For example the end-to-end encryption between the peers and the clients were disabled, since it has no impact on the performance of the gateway but would incur a load for the peers.

Therefore in each request they have to do very little amounts of computation in comparison to the gateway side. It therefore is assumed that the peers have a by far lower load than the gateway system.

Therefore two servers can be used in a symmetrical setup. While one system will run all of the components of the gateway and the client stub, the peers will be run as replicated containers of the peer image on the other system. Both servers are equipped with the following hardware:

- Intel(R) Xeon(R) CPU E31230 @ 3.20GHz, 4 physical cores with hyper-threading
- 16 GiB DDR3 main memory @ 1333 MHz
- 1 GBit LAN network connection between hosts

The following software versions were installed on the systems during the tests:

- Operating system Linux, distribution Grml 2017.05 with Kernel 4.12.0

- Docker server version 18.01.0-ce
- Python version 3.6.4
- Application server Nginx 1.13.7 delivering content from uWSGI 2.0.15
- MongoDB version 3.6.2

8.3 Analysis of results

Building on the parameters defined in section 8.1.2, reasonable ranges for their values were deduced in an iterative approach used during test development. For example the frequency of requests was increased until the gateway could not answer in a timely fashion any more.

Also the use case of SMC in a smart building was considered. For example, SMC can only be run between a limited number of peers and the effect of round trip delay times is also critical due to the high number of messages exchanged between all peers. Considering these limitations, a buffer was added to the limitations and included into the values tested for the gateway.

This resulted in the parameter values and their combinations as they can be seen in table 8.1.

Run ID	**Number of peers**	**Round-trip delay time**	**Requests per second**
1	10	0, 200, 400, 600, 800, 1000	0.1
2	30	0, 200, 400, 600, 800, 1000	0.1
3	10	0	1, 2, 3, 4, 5, 6, 7, 8
4	30	0	1, 2, 3, 4, 5, 6, 7, 8
5	10	0	6, 8, 10, 11, 12, 13, 14, 15

Table 8.1: Combination of parameters used in tests

It was then observed how the metrics were affected in each of the cases. The following sections lay out the conclusions that were made about the impact on the performance of the system.

The boxplots show the lower and the upper quartiles, while the whiskers show the maximum and minimum values observed in the measurements. [47]

8.3.1 Number of peers

The number of peers was varied between the two values 10 and 30 in the test cases described in listing 8.1. The measurements were made for each of the metrics made in the different test runs, where all parameters apart from the number of peers were left the same. The following observations were made using this approach.

8.3.1.1 Total response time

The influence on the response time for the initial posting of a request is significant for low latencies, as it can be seen in figure 8.1. As the figure shows, this effect gets even stronger if the number of requests per second are getting close to the maximum load that the gateway can handle (the value of 6 requests per second will be discussed in section 8.3.3.2). In both cases, the time that a client waits for the response to the initial request is more than doubled but still on a very low level of under 300 milliseconds.

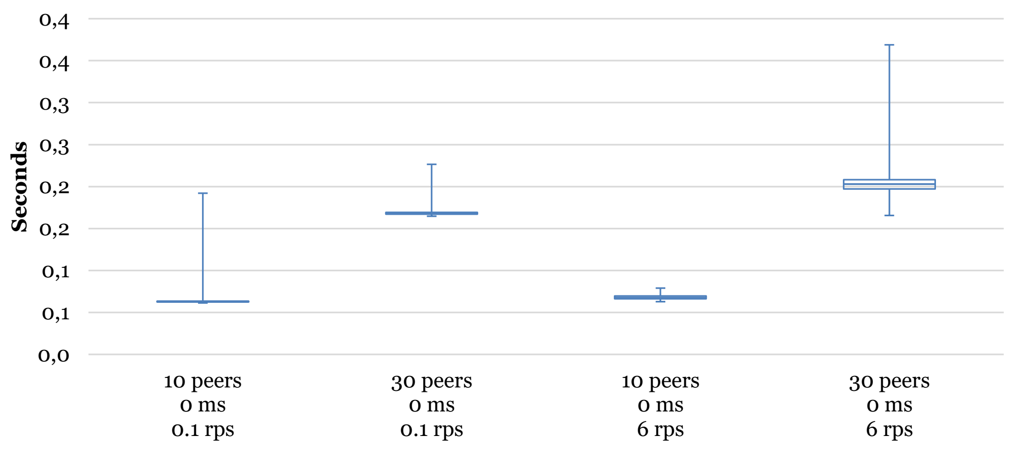

Figure 8.1: Impact of number of peers on response time

This is even though it was tried to keep the effect relatively small by forwarding requests using a thread for each peer as discussed in section 6.3.3.1. Therefore the reason for this significant impact was investigated.

The gateway waits for a response of all peers in order to provide a correct status message for the initial acceptance or failure of a request in the response to the client. As it will be shown in section 8.3.2.2, the initial response of the gateway depends largely on the initial response of the peers. The gateway therefore only adds a significant overhead per peer, if the responses from peers are received extremly fast. Figure 8.2 shows, that in case of higher latencies, the relative overhead provided by the gateway when forwarding a larger number of requests is relatively small.

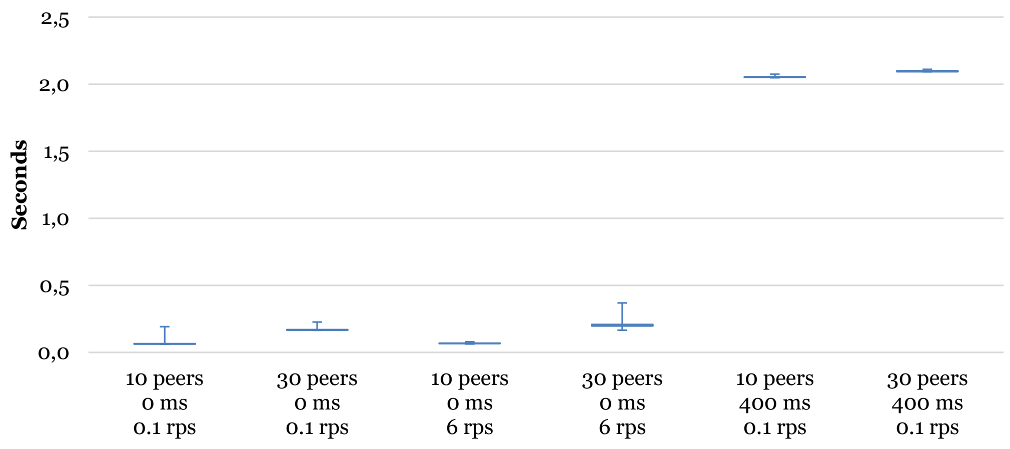

Figure 8.2: Impact of number of peers on the total amount of time it takes to forward a request to all applicable peers

This shows, that as long as the gateway has sufficient resources to handle the forwarding of messages, the number of peers should not affect the response time strongly. In case the initial posting of a request does not require a status message, the server could return a success or failure message to the client immediately in order to directly be able to respond to a client request.

Interpretation: The overhead added by the request handling executed by the gateway is not affected strongly by the number of peers in networks with higher latency. In case of low latencies, the relative impact of the gateway overhead increases. The low absolute values show though, that this increase is not significant for the response times as perceived by humans.

8.3.1.2 Memory usage and CPU utilization

For each received request, the gateway has to forward it multiplied by the number of applicable peers. Each of them may in turn request static data from the gateway afterwards. In the test setup this requested data object was the original request, normally the key of the client may also be necessary if the results should be encrypted. The effect of the number of peers therefore has a direct effect on the computational effort, as can be seen in figure 8.3. It shows the CPU utilization in dependency of the requests per second and the number of peers.

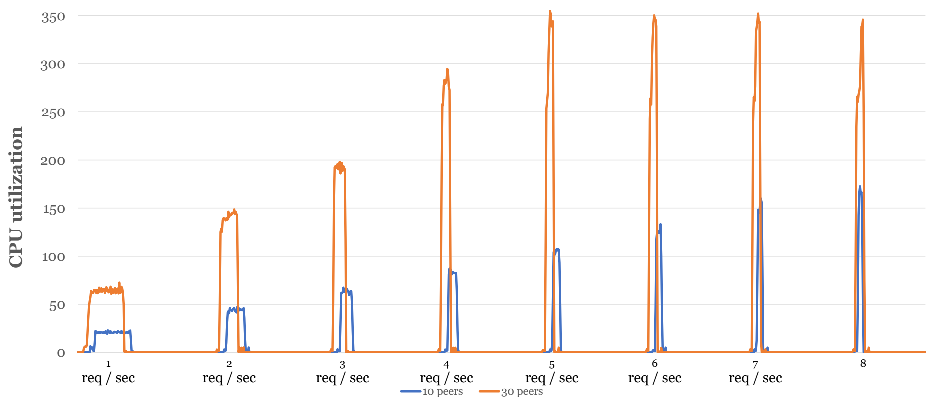

Figure 8.3: Impact of number of peers on gateway CPU utilization

The data is signed and due to TLS the communication is encrypted with a different key for each of the peers. The steep incline can therefore be explained by the cryptography used during information request forwarding.

Figure 8.4 shows the memory usage in dependence of the requests per second and the number of peers.

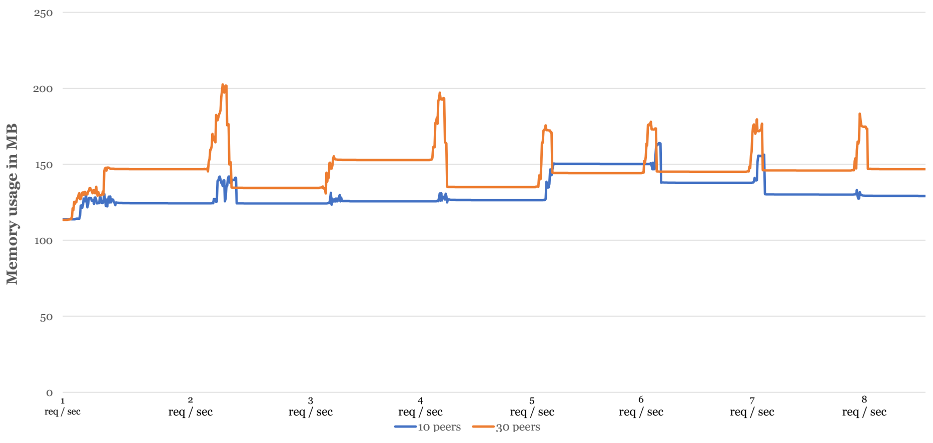

Figure 8.4: Impact of number of peers on gateway memory usage

Higher numbers of peers lead to increased memory usage, whereas the spikes in the graph can be seen especially during periods where requests are being received.

Interpretation: Since the number of peers is limited by the SMC protocol, this limit should not be a problem in the use cases considered in this thesis and applicability is given. This is due to the fact, that SMC will incur a higher load for peers due to the amount of messages exchanged between them in comparison with the request that is only forwarded once by the gateway.

8.3.2 Round-trip delay time

The added round-trip delay time was varied between 0ms and 1000ms in steps of 200ms in the test cases as described in listing 8.1. Following observations were made about the influence of the round trip delay times:

8.3.2.1 Total response time

As figure 8.5 shows, the resulting delay for a response is therefore strongly influenced by the added latency.

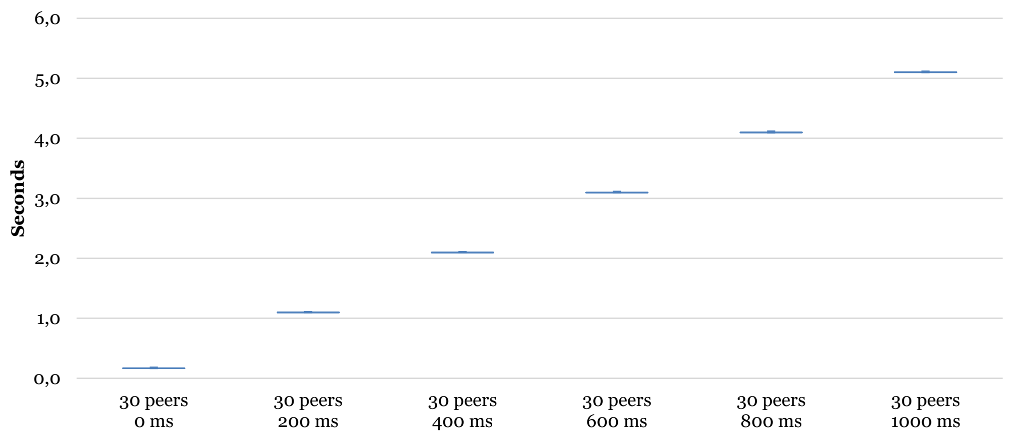

Figure 8.5: Impact of round trip delay time on total response time

For each 200 ms of round trip delay time added, the time that the gateway needs for a response increases by one second. This is due to the fact that for each request, the gateway forwards the authorized request to the peers. TLS increases the amount of messages exchanged with a peer before communication, since a handshake has to be executed before actual information can be exchanged.

As shown in section 8.3.1.1, the number of peers has no strong influence in higher latency settings.

Interpretation: The client would have to wait for peer responses also if making a request directly to them which would also result in increased round-trip delay times of no gateway was used. If a similar protocol that uses TLS and the same mechanisms would be run directly between the clients and the peers, the only additional overhead for response times is incurred by the communication between the client itself and the gateway.

8.3.2.2 Time until initial response from all peers for an information request

As an additional metric for the round-trip delay time test case, the time until the peers have initially responded to the request were measured separately. Figure 8.6 shows the total response time to an initial information request in comparison to the total amount of time it takes to forward a request to all peers. As can be seen, this amount of time it takes to forward a request to all peers makes up the largest part of the overall response time of the gateway to a client.

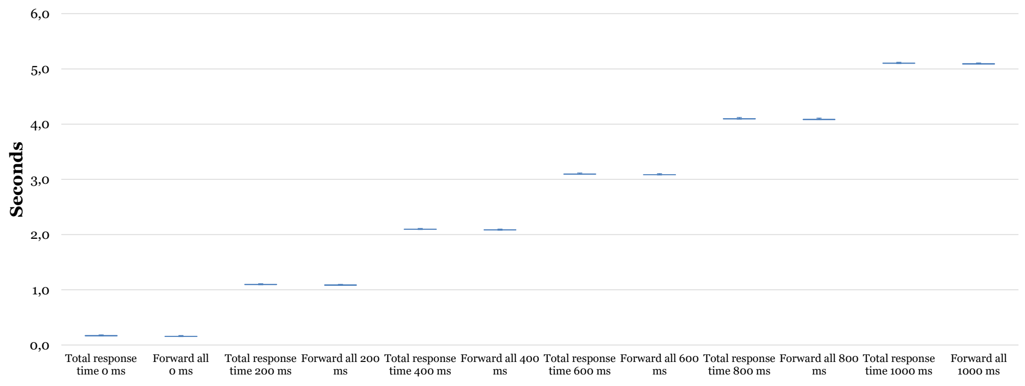

Figure 8.6: Impact of round trip delay time on time until gateway has received responses from all 30 peers

Interpretation: The initial answer time by peers is strongly affecting the total time it takes the gateway to respond to a request. This leads to longer times that clients need to wait. The effect of the overhead presented by the gateway on response times is therefore only minimal.

8.3.2.3 Memory usage and computation effort

The effects of the round-trip delay time on the gateway's CPU utilization can be seen in figure 8.7. While the median doesn't differ significantly, for low delay times the CPU utilization is less evenly distributed than for high delays.

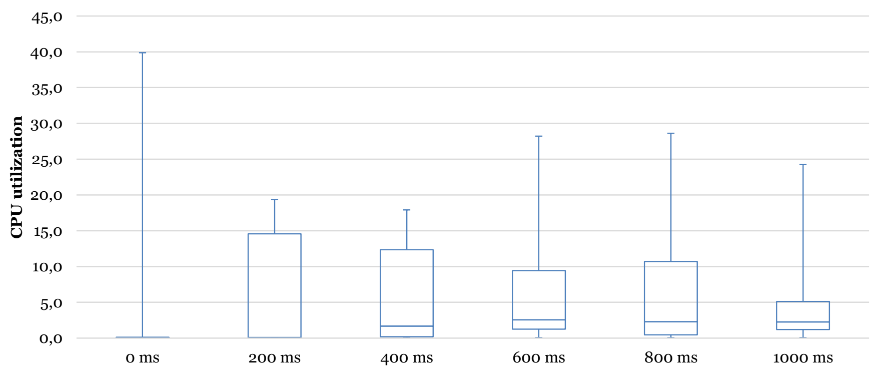

Figure 8.7: Impact of round trip delay time on CPU utilization of gateway (30 peers)

In settings with low delays, all requests can be forwarded very quickly which explains the strong outliers in comparison to the high delay settings where the forwarding of the request is stretched out over a longer time. The results with the strong outliers could be confirmed in an alternative run with less peers, while the outliers were not as strong since the messages were forwarded to less peers.

Figure 8.8 shows the effect of delays on the memory usage of the gateway.

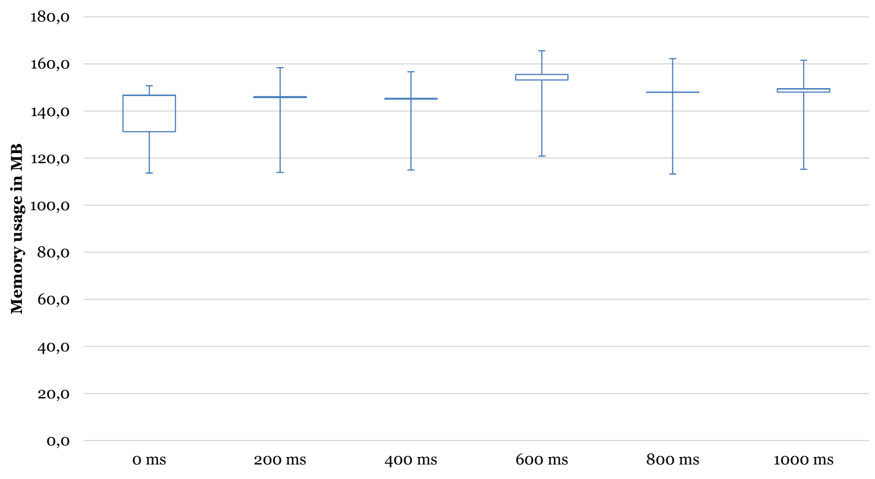

Figure 8.8: Impact of round trip delay time on memory usage of gateway

No clear effect on the memory usage could be observed.

Interpretation: The impact of delays on gateways resources is minor since the same computations are executed as usually, just slower. This spreads out the load, that the gateway has to handle over a longer period of time.

8.3.3 Frequency of requests

The frequency of requests was varied between 6 and 15 requests per second for a setting with 10 peers, while smaller steps between the higher values were chosen as can be seen in listing 8.1. The goal was to observe how the request limit of the gateway slowly is being reached. Following observations were made about the influence of the round trip delay times:

8.3.3.1 Memory usage and computation effort

The request frequency has a strong effect on the CPU utilization of the gateway as can be seen in figure 8.9.

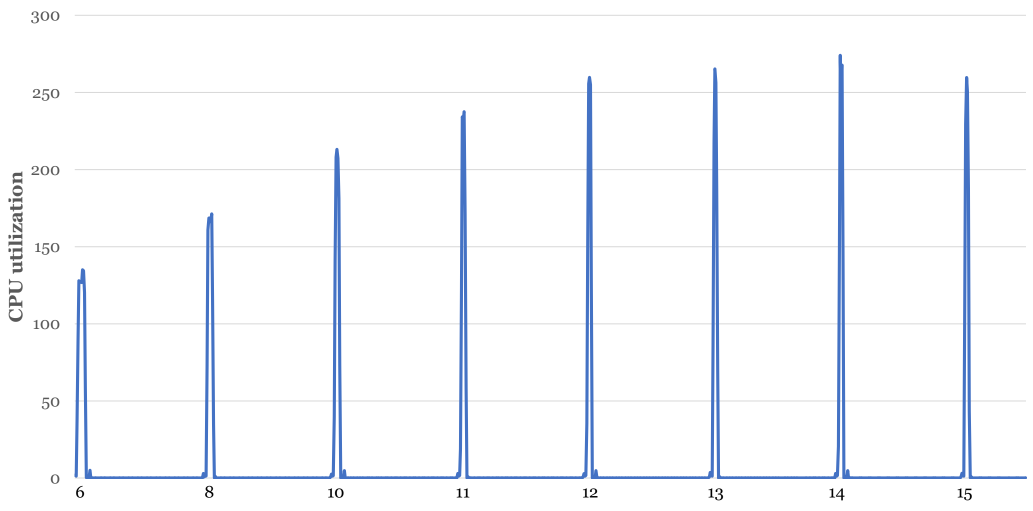

Figure 8.9: Impact of request frequency on gateway CPU utilization

This is due to the cryptography, that has to be executed for each of the peers that the messages are forwarded to. At a certain amount of requests per second, the maximum CPU utilization is reached which influences the maximum handled load and will be discussed in section 8.3.3.2.

The used amount of main memory can be seen in figure 8.10.

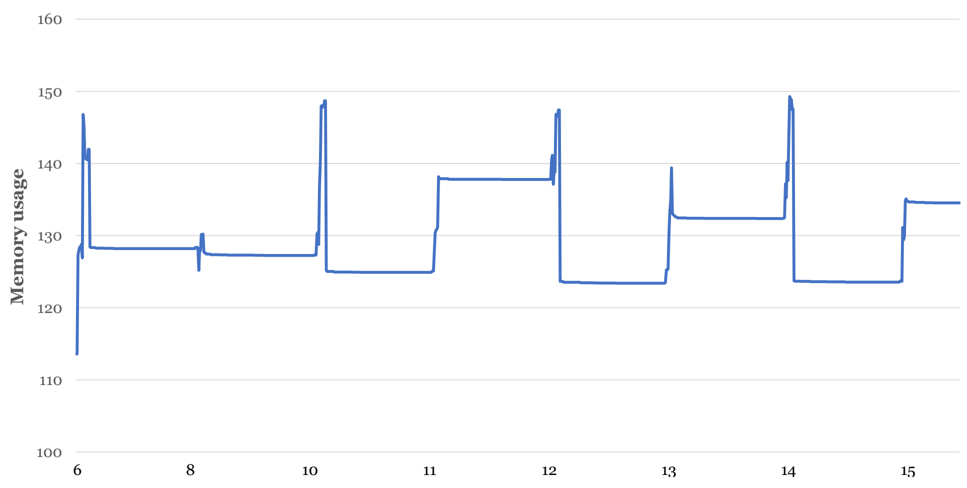

Figure 8.10: Impact of request frequency on gateway memory usage

The amount of used memory jumps between different levels, while no trend can be derived from the graph. This effect is likely due to the application server used to deliver the gateway interface, which automatically adds and removes process instances depending on the amount of requests currently being handled. Once such an instance is spawned, the gateway logic itself only has minor impact on the memory usage.

Interpretation: The resource load handled by the gateway due to the cryptographic operations mainly uses the CPU while the memory is underutilized. The gateway should not provide a bottleneck in actual SMC runs though, since the protocols run between peers are even more arithmetically intensive.

8.3.3.2 Maximum handled load

As seen in figure 8.9, the maximum load of requests handled by the gateway is limited by the CPU. If more requests are made per second than can be handled by the gateway, it will slowly become overloaded. During the execution of test cases, the frequency was therefore increased until the response time was higher than the period between each request. It could then be seen, starting at which request frequency the gateway took longer to respond than the period between requests.

With a number of 10 peers, the limit was seen at around 13 requests per second. With a number of 30 peers, the limit was seen at around 5 requests per second. This limit therefore describes the maximum continuous load which the gateway can handle. The peak limit is depends on the platform used for delivery and how well it can queue unfinished requests.

Interpretation: Considering the use case of SMC, the number of requests that can be accepted per second are sufficient since they also cause long-running and expensive computations for peers. Furthermore such a high continuous load should not be expected in normal office buildings with the described use cases anyway.

8.3.3.3 Total response time

Since the the application server showed the behavior of blocking connections when becoming overloaded for longer periods of time, only sequential requests were tested. The increase of total response time with higher request frequencies can be seen in figure 8.11.

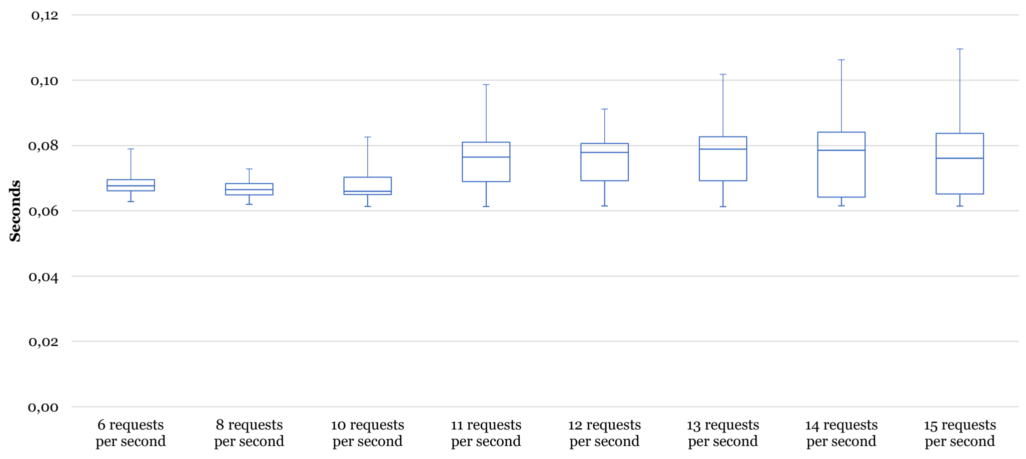

Figure 8.11: Impact of request frequency on total response time

As long as the gateway does not come close to it's performance limit, no increase is seen. As soon as the resources are fully utilized as described in section 8.3.3.1, the response times increase though.

Interpretation: The increasing response times are a sign for the fact that the gateway is becoming overloaded. If a load is so high for a longer period of time, the gateway would after a short time not be able to handle the requests any more anyway. The slight increase in response times therefore is a smaller problem, that should not occur in case of a load that is within the limits of what the gateway can handle anyway.

8.3.4 Summary of performance evaluation

The analysis has made several insights about the gateway performance possible. For one, the overhead added by the gateway is slightly noticeable by clients in case of sufficient resources of the gateway. Second, the protocol doesn't delay the communication with peers noticeably in case of low latencies.

These boundary conditions are influenced by various factors though. Most importantly, the utilization of gateway resources is strongly influenced by the number of peers. The second relevant factor is the frequency of requests made to the gateway.

High latencies slow down the response times, which are largely dependent on the request time of peers. This effect is independent of the usage of a gateway though, since also in direct communication between peers and clients the delay would affect the response times.

The gateway therefore only adds a small overhead to the communication under normal circumstances. The observed performance boundaries are more than sufficient for the use cases set out at the beginning of the thesis. Applicability is therefore given and could be further optimized as will be discussed in the conclusion.

Chapter 9

Related work

The shortcomings of the traditional trust-based approach for privacy have been shown in section 4.1. In contrast to this it has been suggested to design systems in a way that by itself ensures privacy [48]. This eliminates the necessity for entrusting a third party with personal information while not being able to enforce their privacy guarantees or observing the actual usage of information.

The secret multiparty computation and decentralized anomaly detection systems that defined the scope of this thesis in section 1.2 are an example for this approach. What makes them special is the fact, that information is supposed to stay at the data sources and give them full control over this information. It is then still possible to derive aggregated information from the data that is stored only locally.

The authorization decisions that lead to such computations are made on the basis of previously defined policies. This thesis has proposed a solution, where the policy that an authorization decision is made is specifically stated in conjunction with the request that was made. Furthermore the source of both the policy and the request can be verified.

It is possible to verify the following privacy-critical aspects of authorization decisions. A second layer of access control can ensure that the policy was enforced correctly and in accordance with the access rights and constraints specified in the policy. Using the original policy and request, verifying their signatures can ensure integrity of the policy storage, since any modifications would be immediately noticed. While therefore removing the need for trust, a central entity is still responsible for all authorization decisions that are made.

This chapter describes some alternative approaches that have been suggested in order to provide privacy-preserving access control.

9.1 Secret Sharing based signatures

The setting of this thesis described use of the protocol in embedded devices in dynamic environments. They therefore possess limited resources and quality of their network connection may potentially vary significantly. This limits the amount of data that can be transmitted, processed and handled with cryptographic operations.

Since in the present setting trust does not need to be removed entirely, it would be enough if a set of entities trusted by a peer would confirm the validity of an authorization decision. In that case, said entities could be operated in a controlled environment and with sufficient resources. When an authorization decision is made, they can receive and review it. If they agree on the validity of the information that they are provided, they can issue a verification message that is then forwarded to the peers.

Secret sharing based signatures suggest a way how participation of a set of entities, called players, is required in order to access a secret such as a key [49]. Coming back to the example, a verification message could then be granted only if all players participating in the verification process agree on the result.

This process could therefore be used as an alternative to the approach of forwarding verifiable authorization grants to all involved entities.

In comparison to the solution presented in this thesis, the advantage is increased performance and lowered complexity in the side of the data sources. As a disadvantage, it increases the overall effort and complexity of the system and just shifts it to other parties, the players of the secret sharing scheme. The need for trusted third parties in turn is in contrast to the central benefit developed in this thesis, where full transparency is provided to the data sources.

9.2 Decentralized authorization decisions using blockchain smart contracts

Since the contents of the policy and the request are disclosed to peers anyway, the authorization decisions could be made on a distributed set of nodes as described in [50]. In such a scenario, access policies are stored in a blockchain. When access to a resource is needed, the according request is made to this network. The resulting access decision can then be enforced by the requested resources.

One possible alternative approach for ensuring correct evaluation of policies is using a distributed approach based on smart contracts. In smart contracts, instructions are stored in a blockchain distributed on a set of peers that participate [51]. In the present context, policies could be stored in a distributed manner and contain the access control demands of peers. The according set of instructions is then carried out by the network by a trigger as defined in the contract, for example on request. An example for this approach is Slock.it, which makes access decisions for physical devices based on smart contracts [52].

The advantage of such distributed consensus mechanisms is, that they remove the need for a single, central entity. The authorization decision can therefore be trusted and doesn't have to be re-evaluated at the second layer of access control.

In comparison to the solution developed in this thesis, only one layer of access control is needed in contrast to the proposed two-tier model where authorization decisions are made centrally and then re-evaluated during second layer verification checks. On the other hand, the distributed nature of policy storage and evaluation requires ways of ensuring privacy for the authorization information in the network executing the smart contracts.

9.3 Enigma

Enigma promises to provide distributed data storage and computations while ensuring privacy [53]. It is a combination of secret sharing based secure multiparty computation, smart contracts and a distributed hash table (DHT) that allows to find referenced information. Through this combination it tries to achieve scalability and efficiency for the provided services.

Information is split in the categories public and private. The public part needs to be stored on a blockchain as a record that provides correctness e.g. for access control and identity information and is not privacy critical since it only links to data with non-disclosed contents. The private information is then stored in encrypted form in a separate network and can be obtained using a modified protocol that is based on a DHT in order to use it for computation.

In order so that no single entity may gain access to information, the different parts of a computation process are split up. The hierarchical SMC scheme that is used offers scalability for complex computations and redundant computations are avoided.

A possible application in the described use case would be a set of peers transferring their data to the network while providing authorization information in the blockchain. When a client requests access, the networks evaluates the query and can then provide data in aggregated form. The requests are logged and therefore accountability is given.

In comparison to the solution developed in the thesis, a central similarity is the use of SMC for computation of results. An addition is the distributed authorization process as described in section 9.2. Centralization such as with the gateway solution is completely avoided, which brings advantages but also incurs complexity of queries.

Out of the central privacy requirements, transparency (PP.1) is given. Unlinkability may not be given, since peers directly provide their information to the enigma network and therefore have to trust that it will not use the derived information such as frequency or size of measurements to draw conclusions about the peers. Since data is transferred to the enigma network and computed there based on the decision made based on the access control information present in the blockchain, intervention (PP.3) is not possible for peers after a request has already been made.

9.4 Sticky policies

A common problem when providing data to someone is that they may share it with a third-party. An example for this would be an online shop that passes payment information to a service provider that takes care of the invoice.

Since the data may be passed on several times, there is no relation to the original purpose and statement of agreement of the data source. Also, anybody in possession of the data can use it for effectively anything they want without any technological constraints.

Sticky policies are an abstract concept for bundling data with access policies [54]. The idea therefore is to include the policies about the circumstances of allowed data processing operations when providing it to another party. This is supposed to ensure usage for only the specified purposes, while the data source would have to newly agree to any other intended uses.

An exemplary implementation to this could be encrypting personal data while attaching a document containing the policies to it. [55] It furthermore contains information about a trusted party that is in possession of information that allows decryption of the information and therefore access to it's processable representation. The goal is to be able to pass on a file between multiple parties without accessing the contents in clear text form.

When a service provider finally requires access to the data, it contacts the trusted party party with information about the intended usage of the information. The trusted party then decides if the requester should be granted access or not and in case of a positive result provides the key for decryption. Removing the policy part would make the file useless, since access to the raw information can't be gained through the third party any more.

In comparison to the solution developed in this thesis, a third party is introduced again. Furthermore once access to the raw form of information is given, it may be replicated and used for other purposes later on. Therefore privacy as defined in this thesis and especially transparency and intervenability are not given for peers.

9.5 Purpose based access control

Contemporary access control principles such as RBAC have been described in section 4.2.1. Policies are based on attributes of the different entities involved in the access control process. In the authorization decision phase they are then evaluated against the attributes stated in a request. The decision can therefore be based on attributes of artifacts, while no statement is made about the purpose of the request.

The purpose based access control approach tries to close this gap [56]. The allowed purposes that information can be used for are stored together with it. During an information request, its purpose has to be stated. Those two values are then checked against each other and the contents are only provided if a match occurs.

An approach for purpose definitions and deductions has been suggested in [57]. Purposes and requester roles can be defined in a hierarchical structure. In comparison with the earlier solution, data may be present in complex data structures while XML is used as an example. This would lower the complexity of systems as the one that was proposed here, where a separate access control components needs to make the authorization decisions. Using such purpose based access control could have the advantage of bundling all aspects of a request in a single component, such as a data base system that deduces access rights by itself.

A central similarity of this approach with the thesis is including information in an authorization request, that helps to understand it. The weakness of the proposed mode is, that the purpose only constitutes another attribute of the information that can be used when making an access decision. A correct value can be stated in order to gain access, while no control or observation over the further usage is possible. Data sources therefore again have no control over the data in a form that was demanded in the requirements.

Chapter 10

Conclusion and future work

The present thesis proposed a solution for a general approach of querying data from distributed sources while introducing two central novelties. It uses a recently developing understanding of privacy and suggests how to preserve it according to this definition. For data sinks it drastically simplifies the process of querying data over distributed protocols by concealing them behind a central interface.

The solution therefore offers the advantages of two architectures: the privacy preserving properties of secure multiparty computation protocols and the ease of use of cloud solutions. The conceptual result is the present thesis, that describes the concepts that were used to achieve these goals. The practical results of the thesis are the designed protocol and the implemented prototype, that show the feasibility of the proposed concepts.

The question that motivated the thesis about how privacy can be ensured when analyzing increasing amounts of sensor data was described in chapter 1. It was then analyzed in chapter 2 in order to provide an abstract definition of the scope of this thesis, while access control, directory and request processing were identified as the necessary components of a practical solution. This lead to a set of verifiable non-functional information security, privacy and performance requirements and functional requirements for the components of the developed solution that was described in chapter 3.

Different technologies were considered and described as a basis for the solution of the components in chapter 4 and chosen as solution candidates for the system. Since none of those could solve the requirements in their entirety, a new architecture of a system and a protocol that provides it was specified in chapter 5. The implementation of the modularized solution in Python and the use of external software such as for example an authorization provider was then described in chapter 6.

In chapter 7, the results were discussed on a theoretical basis using the initial set of requirements and it was shown, that the developed solution can fulfill the goals as they were previously specified. Using the exemplary implementation, the feasibility of application of the results of this thesis was shown in a model of a real world scenario in chapter 8.

Alternative approaches to the same or similar problems were finally presented in chapter 9.

While the proposed solution gives a possible solution to all of the requirements defined in the beginning, several starting points for further research were uncovered in the process. The following topics should be considered and evaluated as possible improvements or extensions:

Improvements though peer to peer communication: The gateway solution introduces a central point of contact for clients and peers. In the current form, many tasks such as distribution of data are directly handled by the gateway itself. This brings several limitations that are inherent to centralized systems, such as for example providing a bottleneck due to the maximum network load. While the design lays a foundation for scaling the gateway both vertically and horizontally, due to cost and the limited resources in embedded networks, ways of decreasing the load of central components and the network itself should be investigated.

The interaction between the different components of the gateway is already designed in a way that eliminates the need for trust to a high degree. Therefore information doesn't have to be obtained from the gateway directly but could come from any other entity and then be verified using signatures. It is therefore interesting to investigate the advantages that peer to peer communication could provide in this context.

For example, data such as keys and authorization grants are addressed by hashes and provided by the gateway on demand as described in section 5.4.2. During an information request, it is highly likely that the same information is requested by peers that are in spatial proximity of each other. Therefore the static objects could be requested from the gateway only once, while the peers then take care of distributing them among each other. Since they are close to each other, not only the network load of the gateway but even the overall network load decreases.

The approach of addressing content by a secure and verifiable identifier used in the thesis is very similar to peer to peer file distribution protocols. Therefore look-up could be solved by a protocol such as Kademlia [58] in order to obtain static data from a close source.

It should therefore be investigated, how peer to peer communication can improve security, scalability and reliability of different phases of the protocol.

Decoupling gateway components: The use of cryptographic primitives has eliminated the trust that needs to be provided to the gateway to a high degree. Integrity of all information that is forwarded by the gateway such as information requests and access control information is completely ensured through signatures. Confidentiality of results is provided by using end to end encryption.

Section 5.1.3 has described how the modular design of the gateway can be used to distribute the different components in order to decrease the dependence on it. Even when using this method, almost all information still passes it in an unencrypted form though, contradicting the confidentiality requirement. While the gateway solution leverages the advantages of SMC protocols, it therefore introduces a point of concentration of private information.

The largest part of the information that is contained in every request is only of interest for the specific component providing the functionality and not the request processing component or the rest of the gateway. For example, only the access control component processes authorization policies and requests, while only the directory component processes the groups specified in a directory queries. This information could therefore be passed on in an encrypted form without any limitations of functionality.

Further research could therefore show, how gateway components can be decoupled to a higher degree while only components that process an information artifact have access to the contents of it.

Bibliography

[1] S. Smarzly, “Flexible and robust orchestration of secure multi-party computation for privacy-preserving services in dynamic environments,” https://www.net.in.tum.de/fileadmin/bibtex/publications/theses/smarzly2017flexsmc.pdf, 03 2017.

[2] M. Mimoso, “No firewalls, no problem for google,” https://threatpost.com/no-firewalls-no-problem-for-google/123748/, 02 2017, (Accessed on 10/03/2017).

[3] Z. Benenson, E.-O. Blaß, and F. C. Freiling, “Attacker models for wireless sensor networks,” *it-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik*, vol. 52, no. 6, pp. 320–324, 2010.

[4] D. Dolev and A. Yao, “On the security of public key protocols,” *IEEE Transactions on information theory*, vol. 29, no. 2, pp. 198–208, 1983.

[5] M. Brenner, N. Felde, W. Hommel, S. Metzger, H. Reiser, and T. Schaaf, *Praxisbuch ISO/IEC 27001: Management der Informationssicherheit und Vorbereitung auf die Zertifizierung.* Carl Hanser Verlag GmbH & Company KG, 2017.

[6] M. Rost and K. Bock, “Privacy by design and the new protection goals,” *Datenschutz und Datensicherheit - DuD*, vol. 01, 2011.

[7] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, “Shining light in dark places: Understanding the tor network,” in *International Symposium on Privacy Enhancing Technologies Symposium.* Springer, 2008, pp. 63–76.

[8] B. Zhou, J. Pei, and W. Luk, “A brief survey on anonymization techniques for privacy preserving publishing of social network data,” *ACM Sigkdd Explorations Newsletter*, vol. 10, no. 2, pp. 12–22, 2008.

[9] S. John Walker, “Big data: A revolution that will transform how we live, work, and think,” 2014.

[10] N. D. Lane, J. Xie, T. Moscibroda, and F. Zhao, “On the feasibility of user de-anonymization from shared mobile sensor data,” in *Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones.* ACM, 2012, p. 3.

[11] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on.* IEEE, 2008, pp. 111–125.

[12] C. C. Aggarwal and S. Y. Philip, "A general survey of privacy-preserving data mining models and algorithms," in *Privacy-preserving data mining.* Springer, 2008, pp. 11–52.

[13] A. Cavoukian, "Privacy by design [leading edge]," *IEEE Technology and Society Magazine*, vol. 31, no. 4, pp. 18–19, 2012.

[14] J.-H. Hoepman, "Privacy design strategies," in *IFIP International Information Security Conference.* Springer, 2014, pp. 446–459.

[15] National Institute of Standards and Technology, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations," http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf, 01 2014.

[16] American National Standards Institute Inc., "American National Standard for Information Technology – Role Based Access Control," http://profsandhu.com/journals/tissec/ANSI+INCITS+359-2004.pdf, 02 2004.

[17] S. Turner, S. Farrell, and R. Housley, "An Internet Attribute Certificate Profile for Authorization," RFC 5755, Jan. 2010. [Online]. Available: https://rfc-editor.org/rfc/rfc5755.txt

[18] OASIS Open, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0," https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf, 03 2005.

[19] OASIS Open, "eXtensible Access Control Markup Language (XACML) Version 3.0," http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf, 01 2013.

[20] P. Zheng, "Tradeoffs in certificate revocation schemes," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 103–112, 2003.

[21] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, Oct. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6749.txt

[22] N. Sakimura, et al., "OpenID Connect Core 1.0 incorporating errata set 1," http://openid.net/specs/openid-connect-core-1_0.html, 11 2014.

[23] P. J. Leach, R. Salz, and M. H. Mealling, "A Universally Unique IDentifier (UUID) URN Namespace," RFC 4122, Jul. 2005. [Online]. Available: https://rfc-editor.org/rfc/rfc4122.txt

[24] J. Benet, "IPFS - content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[25] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[26] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon, "Soap version 1.2 part 1: Messaging framework (second edition)," W3C, W3C Recommendation 7515, April 2007.

[27] R. T. Fielding and R. N. Taylor, *Architectural styles and the design of network-based software architectures.* University of California, Irvine Doctoral dissertation, 2000.

[28] T. Bray, "The javascript object notation (json) data interchange format," 2017.

[29] R. Barnes, "Use Cases and Requirements for JSON Object Signing and Encryption (JOSE)," RFC 7165, Apr. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7165.txt

[30] M. Jones, "JSON Web Algorithms (JWA)," RFC 7518, May 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7518.txt

[31] M. Jones, J. Bradley, and N. Sakimura, "Json web signature (jws)," RFC, IETF, RFC 7515, May 2015. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7515.txt

[32] M. Jones and J. Hildebrand, "JSON Web Encryption (JWE)," RFC 7516, May 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7516.txt

[33] M. Jones, "JSON Web Key (JWK)," RFC 7517, May 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7517.txt

[34] M. Jones and N. Sakimura, "JSON Web Key (JWK) Thumbprint," RFC 7638, Sep. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7638.txt

[35] T. Dierks and E. Rescorla, "The transport layer security (tls) protocol version 1.2," RFC, IETF, RFC 5246, August 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5246.txt

[36] J. Sermersheim, "Lightweight Directory Access Protocol (LDAP): The Protocol," RFC 4511, Jun. 2006. [Online]. Available: https://rfc-editor.org/rfc/rfc4511.txt

[37] K. Zeilenga, "Named Subordinate References in Lightweight Directory Access Protocol (LDAP) Directories," RFC 3296, Jul. 2002. [Online]. Available: https://rfc-editor.org/rfc/rfc3296.txt

[38] MongoDB, Inc., "A MongoDB Architecture Guide," 2015.

[39] OASIS Open, "REST Profile of XACML v3.0 Version 1.0," https://docs.oasis-open.org/xacml/xacml-rest/v1.0/xacml-rest-v1.0.pdf, 10 2017.

[40] OW2, "AuthZForce (Community Edition)," https://authzforce.ow2.org/, (Accessed on 02/02/2018).

[41] OW2, "5. user and programmers guide — authzforce ce 5.4.1e documentation," http://authzforce-ce-fiware.readthedocs.io/en/release-5.4.1e/UserAndProgrammersGuide.html, (Accessed on 02/02/2018).

[42] OpenSSL Software Foundation, "OpenSSL Cryptography and SSL/TLS Toolkit," https://www.openssl.org/, (Accessed on 02/02/2018).

[43] A. Ronacher, "Flask (a python microframework)," http://flask.pocoo.org/, (Accessed on 02/02/2018).

[44] K. Burke, K. Conroy, R. Horn, F. Stratton, and G. Binet, "Flask-RESTful," http://flask-restful.readthedocs.io/en/latest/, (Accessed on 02/02/2018).

[45] G. Greenspan, "MultiChain Private Blockchain — White Paper," 2015.

[46] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," in *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On.* IEEE, 2015, pp. 171–172.

[47] D. F. Williamson, R. A. Parker, and J. S. Kendrick, "The box plot: a simple visual method to interpret data," *Annals of internal medicine*, vol. 110, no. 11, pp. 916–921, 1989.

[48] M. Langheinrich, "Privacy by design—principles of privacy-aware ubiquitous systems," in *Ubicomp 2001: Ubiquitous Computing.* Springer, 2001, pp. 273–291.

[49] M. Stadler, "Publicly verifiable secret sharing," in *Eurocrypt*, vol. 96. Springer, 1996, pp. 190–199.

[50] G. Zyskind, O. Nathan *et al.*, "Decentralizing privacy: Using blockchain to protect personal data," in *Security and Privacy Workshops (SPW), 2015 IEEE.* IEEE, 2015, pp. 180–184.

[51] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, 2014.

[52] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[53] G. Zyskind, O. Nathan, and A. Pentland, "Enigma: Decentralized computation platform with guaranteed privacy," *arXiv preprint arXiv:1506.03471*, 2015.

[54] M. C. Mont, S. Pearson, and P. Bramhall, "Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services," in *Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on.* IEEE, 2003, pp. 377–382.

[55] S. Pearson and M. Casassa-Mont, “Sticky policies: an approach for managing privacy across multiple parties,” *Computer*, vol. 44, no. 9, pp. 60–68, 2011.

[56] J.-W. Byun and N. Li, “Purpose based access control for privacy protection in relational database systems,” *The VLDB Journal*, vol. 17, no. 4, pp. 603–619, 2008.

[57] J.-W. Byun, E. Bertino, and N. Li, “Purpose based access control of complex data for privacy protection,” in *Proceedings of the tenth ACM symposium on Access control models and technologies.* ACM, 2005, pp. 102–110.

[58] P. Maymounkov and D. Mazieres, “Kademlia: A peer-to-peer information system based on the xor metric,” in *International Workshop on Peer-to-Peer Systems.* Springer, 2002, pp. 53–65.